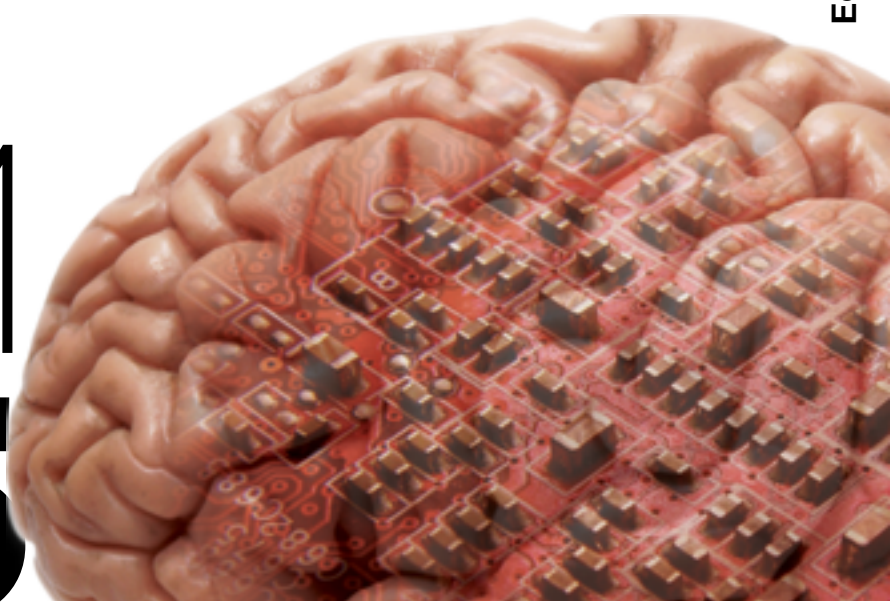
university of groningen

Editors Niels Taatgen, Marieke van Vugt, Jelmer Borst, & Katja Mehlhorn

Proceedings of

ICCM 2015



13th International Conference on Cognitive Modeling

April 9-11, Groningen, The Netherlands

university of groningen

Proceedings of ICCM 2015

13th International Conference on Cognitive Modeling

April 9-11, Groningen, The Netherlands

Edited by

Niels A. Taatgen
Marieke K. van Vugt
Jelmer P. Borst
Katja Mehlhorn

ISBN digital version: 978-90-367-7763-6

Table of Contents

Introduction

The International Conference on Cognitive Modeling (ICCM) is the premier conference for research on computational models and computation-based theories of human cognition. ICCM is a forum for presenting and discussing the complete spectrum of cognitive modeling approaches, including connectionism, symbolic modeling, dynamical systems, Bayesian modeling, and cognitive architectures. Research topics can range from low-level perception to high-level reasoning. In 2015 we specifically welcomed contributions that use computational models to better understand neuroscientific data. The 13th ICCM was held at the University of Groningen in Groningen, the Netherlands, on April 9-11, 2015.

All papers and abstracts in the ICCM 2015 proceedings may be cited as follows:

Author, A., & Author, B. (2015). This is the title of the paper. In N. A. Taatgen, M. K. van Vugt, J. P. Borst, & K. Mehlhorn (Eds.), *Proceedings of the 13th International Conference on Cognitive Modeling* (pp. 1-6). Groningen, the Netherlands: University of Groningen.

Sponsors

KONINKLIJKE NEDERLANDSE
AKADEMIE VAN WETENSCHAPPEN

www.knaw.nl

NWO
Netherlands Organisation for Scientific Research

www.nwo.nl

THE GRONINGEN UNIVERSITY FUND

www.rug.nl/guf

Committees

Organizing Committee

Conference chairs: Niels A. Taatgen
Marieke K. van Vugt
Jelmer P. Borst
Katja Mehlhorn

External chair: Dario D. Salvucci

Program Committee

Cengiz Acarturk
Eduardo Alonso
Erik Altmann
Thomas Barkowsky
Martin Baumann
Roman Belavkin
Thierry Bellet
Marijke Beulen
Mike Byrne
Fokie Cnossen
Rick Cooper
Fabio Del Missier
Adele Diederich
Uwe Drewitz
Wai-Tat Fu
Danilo Fum
Joachim Funke
Francesco Gagliardi
Kevin Gluck
Richard Young
Cleotilde Gonzalez
Wayne Gray
Andrew Howes
Christian Janssen
Mark Keane
David Kieras
Josef Krems
Bernd Kroeger
John Laird
Peter Lane
Christian Lebiere
Luís Macedo
Ralf Mayrhofer
Krishna Miyapuram
Shane Mueller
Josef Nerb
David Noelle
Enkhbold Nyamsuren
David Peebles
Antonino Raffone
Marco Ragni
Frank Ritter
Nele Rußwinkel
Ute Schmid
Mike Schoelles
Patrick Simen
Jennifer Spenader
Robert St. Amant
Christopher Stevens
Terrence Stewart
Andrea Stocco
Ron Sun
Julia Taylor
Manfred Thuering
Greg Trafton
Leon Urbas
Marcel van Gerven
Leendert van Maanen
Jacolien van Rij
Hedderik van Rijn
Boris Velichkovsky
Rineke Verbrugge
Sharon Wood
Richard Young
Iraide Zipitria

Newell Award Committee

Wayne Gray
Glenn Gunzelmann
Terrence Stewart

Keynotes

Combining Human Judgments in General Knowledge and Forecasting Tasks

Mark Steyvers – *University of California, Irvine*

When individuals retrieve facts from memory, or predict future events, how can we aggregate the judgments to maximize accuracy? We analyze the performance of individuals in a number of ranking tasks, such as reconstructing the order of historic events from memory (e.g. the order of US presidents) as well as forecasting tasks where individuals judge the likelihood of geopolitical events. Participants either independently provide judgments or share information in iterated learning environments where each individual in a chain combines their own independent judgment with the judgments from the previous individual in a chain. We propose that a successful aggregation approach requires a cognitive modeling framework that consider a number of psychological factors, including individual differences in skill and expertise, systematic distortions in human judgments, and the role of information sharing. We develop Bayesian cognitive models that assume that each individual's judgment is based on random samples from distributions centered on a latent ground truth and that each individual is associated with a latent level of knowledge of the domain. The models demonstrate a wisdom of crowds effect, where the aggregated judgments are closer to the true answer than the majority of individual judgments. The models also demonstrate that we can recover the degree of knowledge of each individual, in the absence of any explicit feedback or access to ground truth, and suggest ways in which limited information sharing can improve performance.

Interactions between attention and reward for the guidance of plasticity, learning and memory

Pieter R. Roelfsema – *Netherlands Institute for Neuroscience*

Many forms of learning are guided by rewards and punishments. Humans and animals learn complex tasks that consist of multiple epochs by the appropriate choice of reward contingencies. It is not well understood how association cortices learn to link sensory stimuli and memory representations to motor programs when we learn to map stimuli onto responses. I will present AuGMEnT (Attention Gated MEmory Tagging), a biologically plausible model that can train a neuronal network to perform a large variety of tasks while only stimuli and reward contingencies are varied.

The model's aim is to learn action values in a feedforward neuronal network. It is equipped with mechanisms to overcome the structural and the temporal credit assignment problem. The temporal credit assignment problem is solved by a form of Q-learning. The structural credit assignment problem is solved by 'attentional' feedback from motor cortex to association cortex that "tags" synapses that should change to improve behavior. The new learning rule can train a simple neural network in many tasks that are in use in neurophysiology, including

(1) delayed saccade tasks; (2) memory saccade tasks; (3) saccade-antisaccade tasks; (4) decision making tasks; and (5) classification tasks.

Interestingly, neurons at intermediate levels of the network acquire visual responses and memory responses during training that resemble the tuning of neurons in association areas of the cerebral cortex of animals that are trained in these same tasks. I will discuss why and how AuGMEnT can account for learning in so many tasks.

The Sequential Structure of Thought

John R. Anderson – *Carnegie Mellon University*

The goal of experimental psychology from its outset has been to take the wondrous acts of human cognition and decompose them into their basic steps and understand how these steps are put together. For much of its history, psychology has been limited to observations of what went into the mind and what came out. Theoretical endeavors over the generations have built ever more complex models to account for the input-output relationships. Brain imaging offers the possibility of gathering signs from the processes that intervene between input and output. I will describe how we have combined complex models like ACT-R with such signs (tire tracks in the case of fMRI, a breadcrumb trail in the case of EEG) to understand the sequential structure of thought.

A Connectionist Semantic Network Modeling the Influence of Category Member Distance on Induction Strength

Michael Vinos (michael.vinos@gmail.com)
Graduate Program in Basic and Applied Cognitive Science, University of Athens
Ano Ilissia University Campus, GR-15771 Zografos, Greece

Efthymios Tsilionis (eftsilio@gmail.com)
Graduate Program in Basic and Applied Cognitive Science, University of Athens
Ano Ilissia University Campus, GR-15771 Zografos, Greece

Athanassios Protopapas (aprotopapas@phs.uoa.gr)
Department of Philosophy and History of Science, University of Athens
Ano Ilissia University Campus, GR-15771 Zografos, Greece

Abstract

We present a model for inductive inference when both the premises and the conclusion are categorical. The phenomenon under investigation is that less similar categories in the premises lead to stronger conclusions. The model is based on the Rumelhart semantic connectionist network (Rogers & McClelland, 2004, 2008). Simulations addressed the main phenomenon and nine additional non-trivial phenomena of categorical induction (from Osherson, Smith, Wilkie, Lopez, & Shafir, 1990), providing support to the majority of the hypotheses.

Keywords: category-based induction; connectionist model; semantic network; categorization

Inductive inference is a distinctive attribute of human cognition and is distinguished as an important field in cognitive science research. Induction refers to our ability of deriving conclusions through generalization of pre-existing knowledge into new circumstances (Hayes, Heit, & Swendsen, 2010). For instance, we can predict that the sun is going to rise tomorrow morning and also we know that each new to us species of fish can swim and that it bears a set of other fish features. A well-studied form of inductive inference is categorical induction. Category-based induction concerns transfer of the properties of a category or group of categories to some other category (Hayes et al., 2010; Heit, 1997). Transfer is facilitated when the categories under comparison belong to the same superordinate category and therefore share a lot of common characteristics and features (Hayes et al., 2010; Heit, 1997).

Osherson et al. (1990) studied category-based induction presenting to participants problems like those in Box 1. Propositions above the line are called premises and are assumed to be valid. The task was to make judgments about the strength of the conclusion below the line. Properties in these examples (blood sodium concentration) are called "blank" because participants are unlikely to have prior knowledge about them. Use of blank properties permits study of the net effect of the categories themselves upon the strength of the inference, to avoid effects of other related properties (Feeney & Heit, 2011; Osherson et al., 1990).

Box 1. *Problems used by Osherson et al. (1990).*

Hippopotamuses have a higher sodium concentration in their blood than humans.
Hamsters have a higher sodium concentration in their blood than humans.

All mammals have a higher sodium concentration in their blood than humans. (1)

Hippopotamuses have a higher sodium concentration in their blood than humans.
Rhinoceroses have a higher sodium concentration in their blood than humans.

All mammals have a higher sodium concentration in their blood than humans. (2)

Osherson et al. (1990) presented thirteen phenomena of category-based induction, which they argued should be part of a theory of inference strength, based on findings from two studies. They proposed the *similarity-coverage model of argument strength*, suggesting that the strength of the conclusion depends on the degree of similarity among the premises and the conclusion and among the premises and members of the superordinate category that includes them.

Here we focus on phenomena 2 and 6, which concern the *diversity* of the premises. Specifically, the less similar the categories of the premises are, the more they confirm the conclusion. Phenomenon 2 concerns the *general* case, in which premise categories are included in the conclusion category, while Phenomenon 6 concerns the *specific* case, in which a single category includes premises and conclusion.

Phenomenon 2 is exemplified in Box 1. Argument (1) leads to a stronger conclusion than (2) because of the diversity of the premises, even though hamsters are less typical members of the category mammals than rhinoceroses. An example of the Phenomenon 6 is shown in Box 2. Argument (3) is stronger than (4) because lions are more similar to tigers although giraffes are not more similar to rabbits than tigers are (Osherson et al., 1990).

Box 2. *Examples of Phenomenon 6.*

Lions use norepinephrine as a neurotransmitter.
Giraffes use norepinephrine as a neurotransmitter.

Rabbits use norepinephrine as a neurotransmitter. (3)

Lions use norepinephrine as a neurotransmitter.
Tigers use norepinephrine as a neurotransmitter.

Rabbits use norepinephrine as a neurotransmitter. (4)

In addition to these two phenomena, we conducted a preliminary investigation of the remaining nine non-trivial phenomena in Osherson et al. (1990). All phenomena are based on the assumption that conceptual representations are hierarchically structured. A connectionist model that achieves such conceptual organization is the semantic network proposed by Rumelhart and Todd (1993; as cited in Rogers & McClelland, 2004, 2008), henceforth termed *Rumelhart network*. This network discovers similarity structures in its training environment, resulting in similar representations of items with common features, i.e., members of a category (Rogers & McClelland, 2004, 2008). Similarities are context-dependent, so two members with similar representations in one context may vary in another (Rogers & McClelland, 2004, 2008).

Our aim is to show that such a semantic network can exhibit inductive behavior with differentiated strength depending on the premises. More specifically, we examine whether the network will produce stronger conclusions when premise categories are more distant.

Modeling Framework and Method

The classic Rumelhart architecture, as described by Rogers and McClelland (2004, 2008), has been shown to form "coherent categories" of entities, simulating human conceptual organization and acquisition. It also exhibits "inductive projection" of new properties, acquired after initial training (Thibodeau, Flusberg, Glick, & Sternberg, 2013). Our implementation is a feedforward network trained with error back-propagation, displayed in Figure 1.

There are two groups of input nodes, namely *Item*, with 21 nodes, and *Relation*, representing "context constraints", with 4 nodes. There are two hidden layers: *Representation,* with 15 nodes, receiving connections from Item, and *Hidden*, with 28 nodes, receiving connections from Relation and Representation and sending connections to the output layer. There are 4 groups of output nodes, each one corresponding to one context, namely ISA, Is, Can, and Has, with 28, 8, 6 and 12 nodes, respectively. An additional output node was used for the blank property (termed *Queem* following Rogers and McClelland). The numbers of nodes in the hidden layers were found to be sufficient for category learning in a reasonable number of training epochs and with stable outcomes, that is, converging to the same category structure in most training trials.

The Queem entity plays the role of substitute/placeholder for every blank property in Osherson et al. (1990), for example, "use the neurotransmitter Dihedron", "require titanium for normal muscle development", "have a higher sodium concentration in their blood than humans", etc. As noted by Rogers and McClelland do (2004, 2008), node labels play no functional role but are merely descriptive tags in the simulation. What is important for the simulation are the structural relations.

An alternative architecture, lacking the Relation input layer and with only one hidden layer, was used during our initial simulations but even though it was found to form appropriate category structures, it failed to exhibit the phenomenon under investigation with the blank property. It seems that this particular structure, including the Representation, is required for the network to exhibit the desired richness of behavior beyond simple categorization.

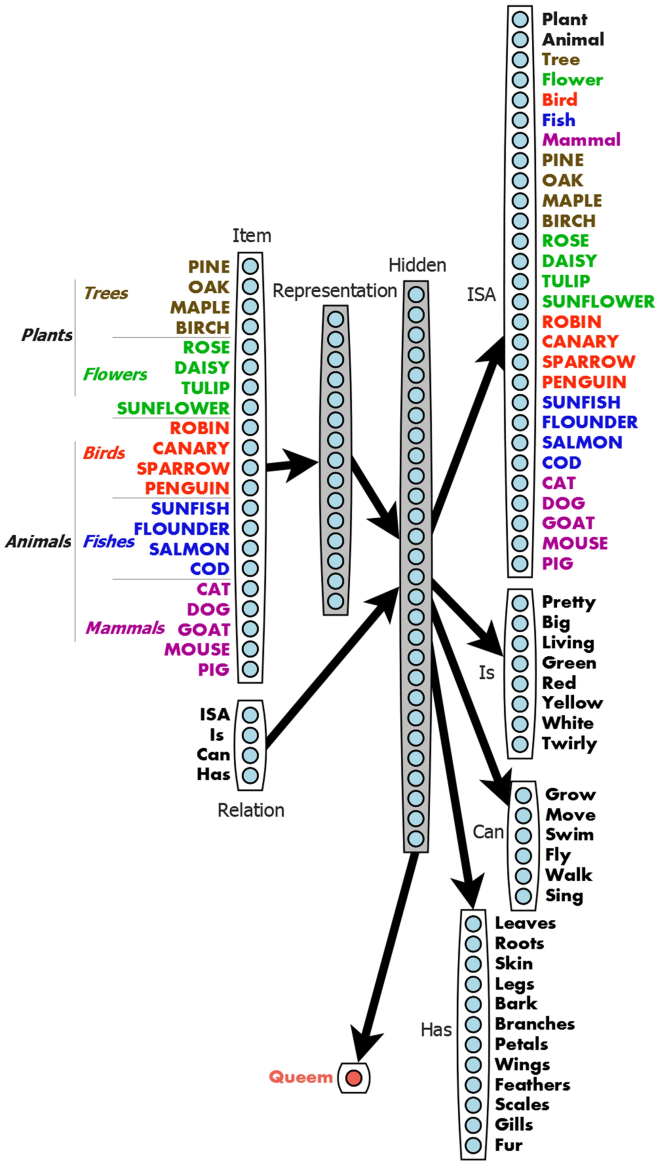


Figure 1: Network architecture used in the simulations.

We used the extended version of the Rogers and McClelland model (2004), with 21 input entities and one addition to the ISA output group. Specifically, we added the property (ISA)Mammal, activated for the training patterns of mammal species, similarly to the other classes. The input and output representations of the model are localist. During initial training, the Queem property has zero activation for all items in all four contexts.

There were 84 training patterns (21 items in 4 contexts). Here is a coding example for *Penguin* in the *Can* context:
Input: Item=(0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0) |PENGUIN; Relation=(0,0,1,0)|Can
Output: ISA=(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0); Is=(0,0,0,0,0,0,0,0); Can=(1,1,1,0,1,0)|Grow; Move, Swim, Walk; Has=(0,0,0,0,0,0,0,0,0,0,0,0), Queem=(0)

Simulations adopted the strategy of Rogers & McClelland (2004, 2008), with 2,500 epochs of initial training, leading to coherent categorization and modest generalization of the new property in the different contexts after additional training. Learning rate was set to 0.3 and momentum to zero. After the initial training, the learned categories were revealed in cluster analysis of the Representation layer.

Subsequently, pairs for the comparisons were chosen. Each comparison involved the generalization of the blank property Queem when the network learned that two members of a category, similar versus dissimilar to each other, possess the property. The comparison is meaningful when it concerns the learning of the property in the same context, Is, Can, or Has. The ISA context was excluded because it is only relevant to item classification within the hierarchy. The pairs were chosen in the following fashion: A computation of euclidean distances between all Hidden layer activation vectors was carried out for each context (having in mind that the clustering of representations on the Hidden layer differs in several respects from that on the Representation layer, which is context independent.) Based on that, the two most similar species were identified, for instance Goat–Dog from the category Mammals, to form the similarity condition. For the second pair, that is, the diversity condition, we keep one of the two chosen entities and computed the euclidean distances between it and all other entities in the superordinate category. Keeping to the above example, we sought the species most distant from Goat among all animals, which turned out to be Canary. So the pair for the diversity condition was Goat–Canary.

Subsequently, for each of the two pairs, the network was trained for additional epochs past the initial 2,500 to learn that the two species possess the Queem property in the particular context but not in the other three contexts (a total of eight training patterns). An example training pattern is:
Input: Item=(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0) |GOAT; Relation=(0,0,0,1) |Has.
Output: ISA=(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0); Is=(0,0,0,0,0,0,0,0); Can=(0,0,0,0,0,0); Has=(0,0,1,1,0,0,0,0,0,0,0,1)| Skin, Legs, Fur; Queem=(1)| Queem

A criterion was established to terminate additional training when the new property has been acquired. An activation threshold of the Queem node for the relevant inputs might be appropriate but this would demand a large number of training epochs. Instead, after some experimenting a mean error criterion of 0.002 was adopted, which ensured activations for the members of both pairs greater than .9 and usually greater than .95 for at least one of them. With this termination criterion, 345–977 (M=579, SD=176) additional epochs of training were required.

After the completion of the additional training cycles we followed Rogers and McClelland (2004) and modified the network weights, retaining only those from the Hidden layer to the Queem node, and replacing the rest from the initial training (2,500 epochs). Thus the remainder of the originally trained network was unaffected by training the Queem node. This network was submitted to a test with 21 input lines containing all species/items in the current context, to obtain activation values of the Queem node. Separate values for some species, or mean values for a category (at any level in the hierarch) or for all living things, as appropriate, were compared to test the experimental hypothesis. These activation values were used to quantify the strength of the argument and the corresponding "confirmation score" (comparing to Osherson et al., 1990). In this manner, the strength of the conclusion (and hence of the whole argument) concerning a certain species corresponds to the activation of the Queem node for this species in this context. When the conclusion concerns a category (e.g., fish), its strength corresponds to the mean activation for all species belonging to the category.

Results

Results are divided in two parts: The first part corresponds to the main phenomenon, that is, Premise Diversity. The second part concerns the other nine phenomena described by Osherson et al. (1990). Due to space limitations, we present results only for the Has context/relation (except when examining the context effect) but the results were similar for the other contexts. In addition, the patterns of results do not alter even with different premises, given that the latter obey the selection rules.

Premise Diversity Simulations

We have put together graphs in Figure 2 depicting the results for all the types of simulations for the two forms of the phenomenon, general and specific. Accompanying each bar chart is the respective argument formed under the general paradigm proposed by Osherson et al. (1990). Indicative colors are used for the bars and their corresponding arguments (either diverse or similar). In general for all simulations the results supported the initial hypothesis that the arguments with distant premises are stronger than the arguments with close premises. Note that, in the graph depicting the results for the specific version of the phenomenon, the three species, which correspond to the

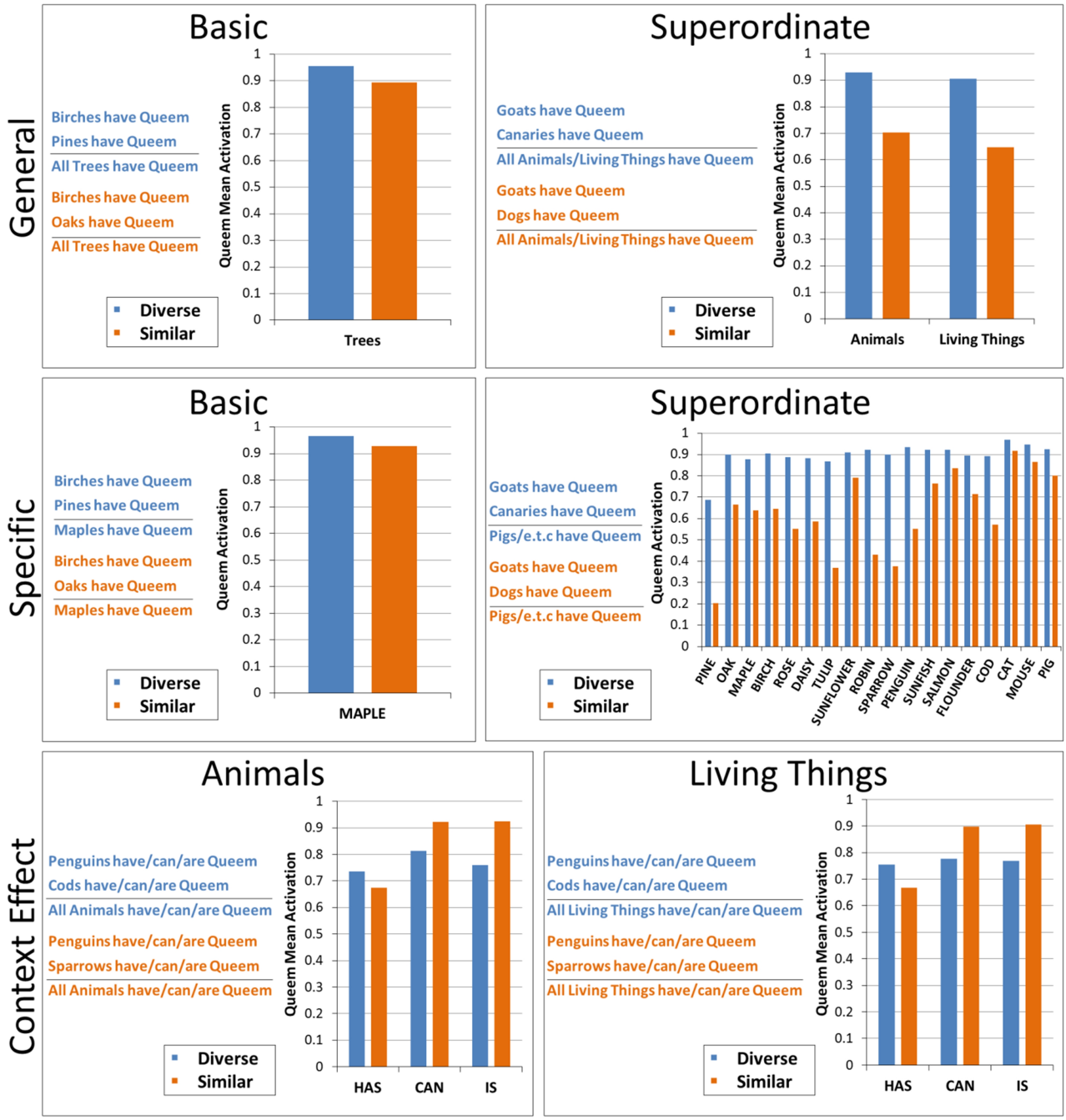


Figure 2: Activation of the Queem node for the different simulations of the phenomenon of premise diversity in each condition (diverse and similar). Inside each panel we give an example of the arguments in each condition. Above the line we present the premises that were used for the additional training and under the line the conclusion that was used for testing. The top two rows of panels refer to the two versions of diversity (general and specific) and the bottom row to the simulation with the different contexts. The columns of panels refer to which category the simulation was performed.

premises used, are not included.

Two kinds of premises were used in the diverse condition: one where the entities/items both belong to the same (basic) class as the conclusion (Trees) and another where the items belong to the same superordinate category but not to the same class (Mammals and Birds), as shown in Figure 2. This has to do with the limited range of distances between the activation vectors of the Hidden layer for a single class. The differences in activation values for the Queem node were larger for the *superordinate* simulations.

We finally conducted a series of simulations to examine the differentiating impact of context on the generalization of blank properties. Simulations revealed the paradoxical effect of *reversal* of the phenomenon. That is, the arguments of the diversity condition that exhibited greater strength than the ones of the similarity condition in one context may display a

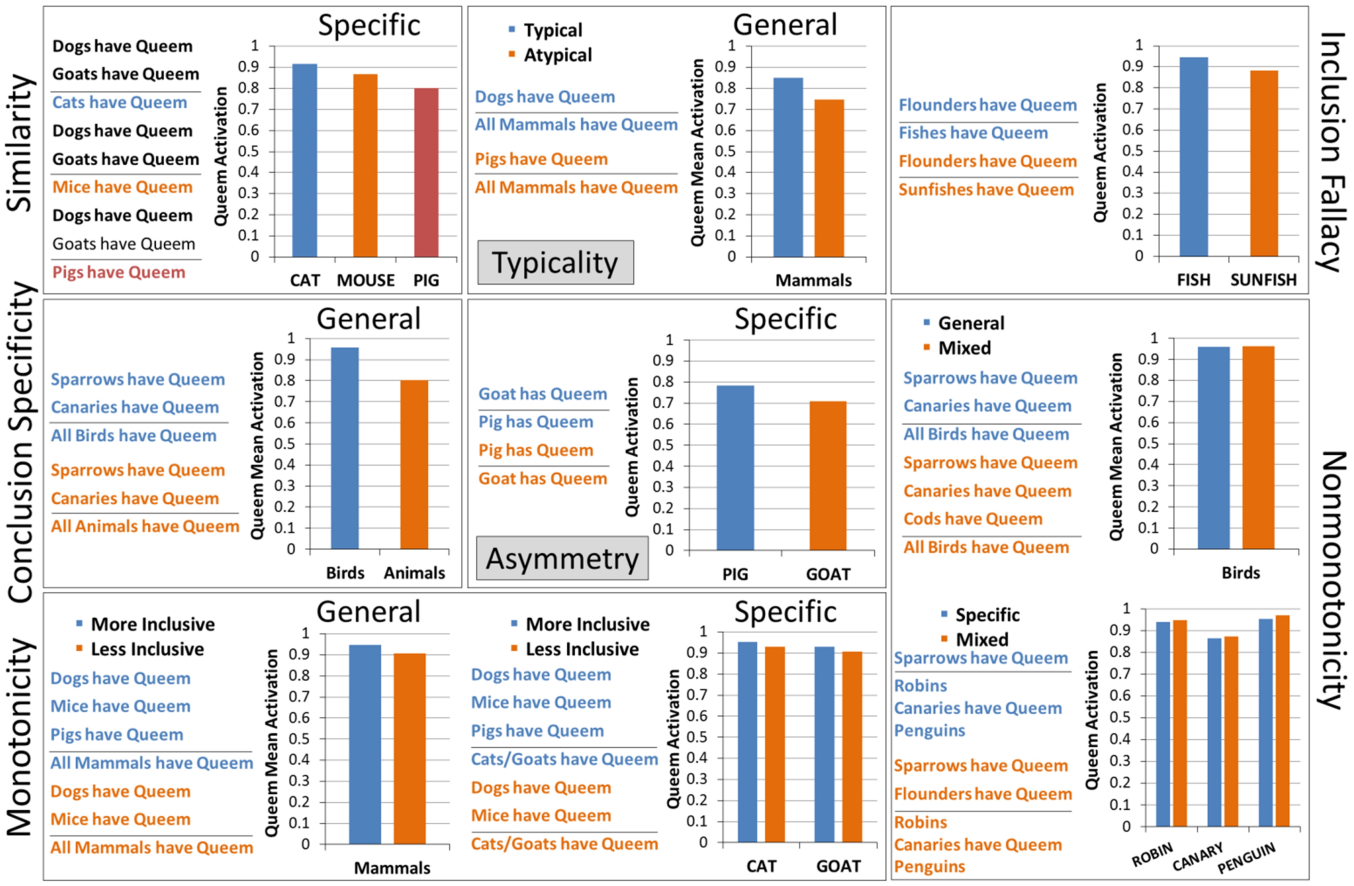


Figure 3: Activation of the Queem node for the remaining nine phenomena. As in Figure 2, inside each panel we give an example of the arguments in each condition. On the left side of each panel the name of the specific phenomenon is presented. The two columns of panels refer to the two types of arguments (general and specific).

reversed strength pattern when examined in another context: the corresponding arguments of the similarity condition are now stronger than the ones of the diversity condition.

Simulations of the remaining Phenomena

For the rest of the phenomena, the results are displayed in Figure 3, accompanied by the respective arguments and using indicative colors for their types. Although there are in total eleven phenomena, two of them are trivial (the premise-conclusion identity and the one in which the conclusion category is included in the premise category) and hence require no investigation or validation. The majority of hypotheses posed from these phenomena have been consistently supported in our simulations although the differences in activation values were not large. The only hypotheses not supported were the two referring to non-monotonicity. Brief descriptions derived from Osherson et al. (1990) and clarifications are given below.

Premise Typicality: "the more typical the premises are of the conclusion, the more they confirm it". We selected species for the premises from one class computing the mean value of the activation vectors for all class members over the Hidden layer and then choosing the one closest to it as the typical member and the most distant one as less typical.

Conclusion Specificity: "the more specific is the conclusion, the more it is confirmed by the premises"

Premise Monotonicity (general and specific versions): "more inclusive sets of premises yield more strength than less inclusive sets"

Premise-Conclusion Similarity: "the more similar the premises are to the conclusion, the more they confirm it"

Premise-Conclusion Asymmetry: "single-premise arguments are not symmetric, in the sense that premise/conclusion may not have the same strength as conclusion/premise"

Non-Monotonicity (general and specific versions): "arguments can be made weaker by adding a premise that converts them into mixed arguments". Note, that these two phenomena involve mixed arguments in the sense that they are neither general nor specific.

Inclusion Fallacy: "a specific argument can be made stronger by increasing the generality of its conclusion". To investigate the general conclusion for fish, we did not compute the mean value for fish but instead created a new item (Fish), for which nodes 12–15 of the Item layer had a value of .25, and in the other layers activation values were the same as for the other fish.

Discussion

The simulations supported both the general and specific versions of the diversity phenomenon. The strength of the arguments in the diversity condition was higher than in the similarity condition, and this was observed with many different initial parameters. The cause of this effect lies in the distance of the representations. The greater the distance between two items the more the blank property is generalized across the remaining items. Thus, the distance determines the degree of generalization. Another important conclusion is that the distances between items in the representational space change according to the context; as a result, the diversity effect can be inverted. The power of the context is that it transforms dissimilar objects into similar ones.

Although our main goal was the investigation of diversity we also performed a preliminary exploration of the other phenomena described by Osherson et al. (1990). Our results supported all phenomena except nonmonotonicity. Even though the differences in Queem activation were small, they were consistent. Undoubtedly, there is space for more extensive investigation of these phenomena, including diversity, to discover other aspects of network function.

A number of concerns emerged during our investigation. One issue was the small number of items and subsequently the small size of the categories. The consequence of this, and the fact that members of a category share many common properties, was that the euclidean distances between members of the same class (e.g., Mammals) were not big enough for the diversity phenomenon to appear to a large extent. For this reason we chose to use in our simulations mainly items from the same superordinate category where the number of items is greater and as a result the representational space increases. A possible improvement might be to construct basic categories with more objects by introducing more and appropriately selected properties in the network. This will also lead to an increase in the number of nodes.

Another issue concerns the method of replacing the weights of connections to the Queem node after the additional training. To minimize interactions with the existing knowledge of the network, an alternative method would be to freeze all connection weights during the additional training except the weights to the Queem node. Interesting questions that arise are whether learning could be facilitated with the proposed method (fewer epochs of training) and if the connection weights would be similar across the two methods. The implementation of this method would be a subject of future research.

As Rogers and McClelland (2004) indicate, the semantic network captures all the different kinds of human developmental phenomena about inductive reasoning supported by empirical findings (specificity, coalescence, differentiation etc.). Thus, an interesting new route of investigation could relate to these particular aspects of the categorical phenomena examined in this study.

Finally, in our simulations the different contexts regard the activation of specific properties for the different items. The network discovers the similarity relations between the items and creates the proper representations. However, items in the real world also exhibit causal relations. Causal relations are used very often in inductive inference and they are even preferred over similarity relations (Feeney & Heit, 2011; Hayes et al., 2010). Hence, if the premises and the conclusion share a causal link the argument is judged stronger and the diversity effect almost disappears (Feeney & Heit, 2011). Connectionist models are capable of discovering other forms of relations in the training data besides similarity structures. Recently, a network using the Rumelhart architecture displayed analogical inference (Thibodeau et al., 2013). This network generalized relations to untrained data. Therefore, a potential improvement of our model would be to examine whether learning causal relations would lead to stronger induction for a blank property in comparison to similarity relations.

Acknowledgments

This research has been co-financed by the European Union (European Social Fund – ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Programs: THALIS–UOA–375737, ARISTEIA–COGNIMUSE–GSRT–1770 and ARISTEIA–BabyAffect–GSRT–3610

References

Feeney, A., & Heit, E. (2011). Properties of the diversity effect in category-based inductive reasoning. *Thinking & Reasoning*, *17*(2), 156–181.

Hayes, B. K., Heit, E., & Swendsen, H. (2010). Inductive reasoning. *Wiley Interdisciplinary Reviews: Cognitive Science*, *1*(2), 278–292.

Heit, E. (1997). Features of similarity and category-based induction. In *An Interdisciplinary Workshop On Similarity And Categorisation (SimCat)*. Edinburgh, UK. Retrieved from http://faculty.ucmerced.edu.sci-hub.org/sites/default/files/eheit/files/simcat.pdf

Osherson, D. N., Smith, E. E., Wilkie, O., Lopez, A., & Shafir, E. (1990). Category-based induction. *Psychological Review*, *97*(2), 185.

Rogers, T. T., & McClelland, J. L. (2004). *Semantic cognition: A parallel distributed processing approach*. London, England: MIT press.

Rogers, T. T., & McClelland, J. L. (2008). Précis of semantic cognition: A parallel distributed processing approach. *Behavioral and Brain Sciences*, *31*(06), 689–714.

Thibodeau, P. H., Flusberg, S. J., Glick, J. J., & Sternberg, D. A. (2013). An emergent approach to analogical inference. *Connection Science*, *25*(1), 27–53.

Explorations in Distributed Recurrent Biological Parsing

Terrence C. Stewart (tcstewar@uwaterloo.ca)
Peter Blouw (pblouw@uwaterloo.ca)
Chris Eliasmith (celiasmith@uwaterloo.ca)
Centre for Theoretical Neuroscience
University of Waterloo, Waterloo, ON, Canada N2L 3G1

Abstract

Our ongoing investigations into biologically plausible syntactic and semantic parsing have identified a novel methodology for processing complex structured information. This approach combines Vector Symbolic Architectures (a method for representing sentence structures as distributed vectors), the Neural Engineering Framework (a method for organizing biologically realistic neurons to approximate algorithms), and constraint-based parsing (a method for creating dynamic systems that converge to correct parsings). Here, we present some of our initial findings that show the promise of this approach for explaining the complex, flexible, and scalable parsing abilities found in humans.

Keywords: Neural engineering framework; parsing; localist representation; distributed representation; vector symbolic architectures; holographic reduced representation

Parsing with Traditional Localist Networks

Neural networks have often been explored as mechanisms for parsing language. In many of these approaches (e.g. Cottrell, 1985), a single connectionist node is created not only for each term in the language, but also for each possible usage of that term, leading to a combinatorial explosion of nodes (Figure 1). While these sorts of models provide accurate parsing and show some performance characteristics similar to humans (e.g. Waltz & Pollack, 1985), this exponential growth of components means that they would require more nodes than there are neurons in the human brain. This makes it difficult to see how such an algorithm could be instantiated within the brain.

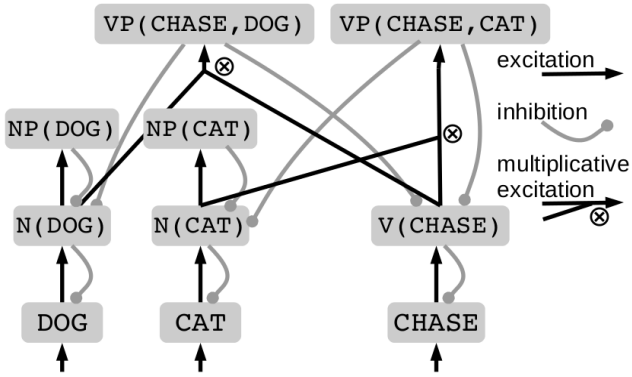


Figure 1: An example of a localist parsing network. A node exists for each possible combination of terms, leading to an exponential growth in resources required. The nodes for S(DOG,CHASE,CAT) and the three other possible sentences are omitted for clarity.

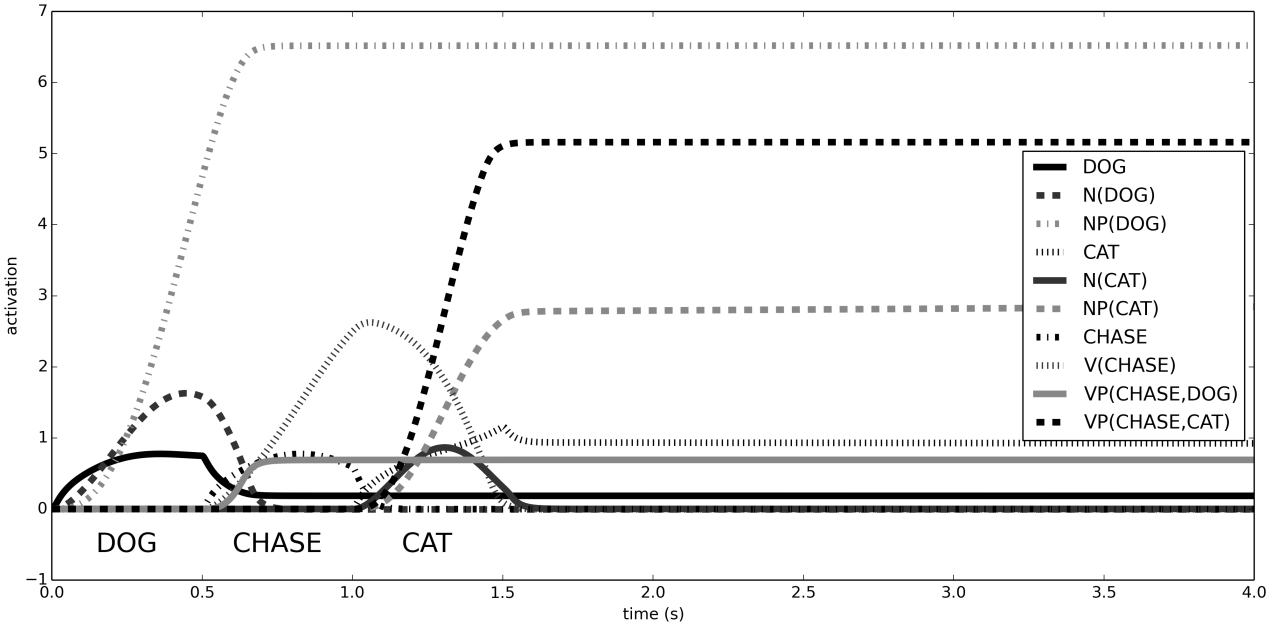


Figure 2: Activity level of each node in Figure 1 as the network is presented with the input DOG for the first 0.5s, CHASE for the next 0.5s, and then CAT for the next 0.5s. The result is the activation of NP(DOG) and VP(CHASE, CAT). The network can also successfully parse "DOG CHASE DOG", "CAT CHASE DOG", and "CAT CHASE CAT".

The network in Figure 1 is capable of parsing sentences from the following toy grammar:

S → NP, VP
NP → N VP → V N
N → DOG or CAT V → CHASE

Excitatory and inhibitory connections increase and decrease (respectively) the activity in the target node proportional to the activity in the source node. The multiplicative excitation connection increases activity based on the product of the activities in the two source nodes. The four nodes for the four possible sentences (S(DOG, CHASE, CAT), S(DOG, CHASE, DOG), and so on) are not shown, but are implemented similarly to the VP nodes, with connections to the corresponding NP and VP nodes.

Distributed Representation

As an alternative to localist representation, other approaches make use of a distributed representation. The idea is to represent content not as the activity of a single node, but rather each node has an equivalent set of numbers (a vector). For example, instead of a single node being active to represent DOG, this might be represented as the vector [0.1, -0.4, 0.7, 0.3, 0.2]. CAT would be another vector, and the presence of both terms would be represented as the sum of those vectors. The particular vectors used for each term might be chosen via some learning process that imposes similarity between vectors, such as DOG being more similar to CAT than it is to CAR. However, for the purposes of this paper we follow the standard process of randomly choosing these vectors.

With this approach, it is possible to re-describe any traditional localist model in a distributed manner. For example, we can take the nodes in Figure 1 and replace each one with a vector. To get the overall state of the system, we add together the vectors for each node, weighted by the activity level of that node.

To implement the connections, instead of using the activity of the source node, we must compute the *similarity* between the overall state vector and the ideal state vector for the source of the connection. Here, we use the dot product operation to compute similarity. So to implement an excitatory connection A → B, we take the state vector x and compute (x • A)B. The result is a vector indicating how much x should be changed. This can also be written mathematically as:

$$dx/dt = BA^T x$$

Importantly, the number of dimensions in the distributed representation's vector can be much less than the number of nodes in the localist representation, at the cost of a slight decrease in accuracy as the vectors slightly interfere with each other. This will work best when only a few of the nodes are active at any given time (i.e. when the distributed vector is formed by the combination of a few basic term vectors). Plate (2003) shows that with vectors with 1000 dimensions one can represent states with 8 nodes active out of a total of 50,000,000,000,000 nodes with 95% accuracy. The distributed representation thus avoids the problem of exponential growth.

Furthermore, Plate's approach and other similar Vector Symbolic Architectures (Gayler, 2003) supply a method for combining two vectors to generate a new vector. That is, instead of randomly generating a vector for N(DOG), it can be computed based on the vector for DOG and the vector for NOUN. For this, we follow Plate and use the mathematical operation of circular convolution (⊛). That is, we set N(DOG) = NOUN⊛DOG.

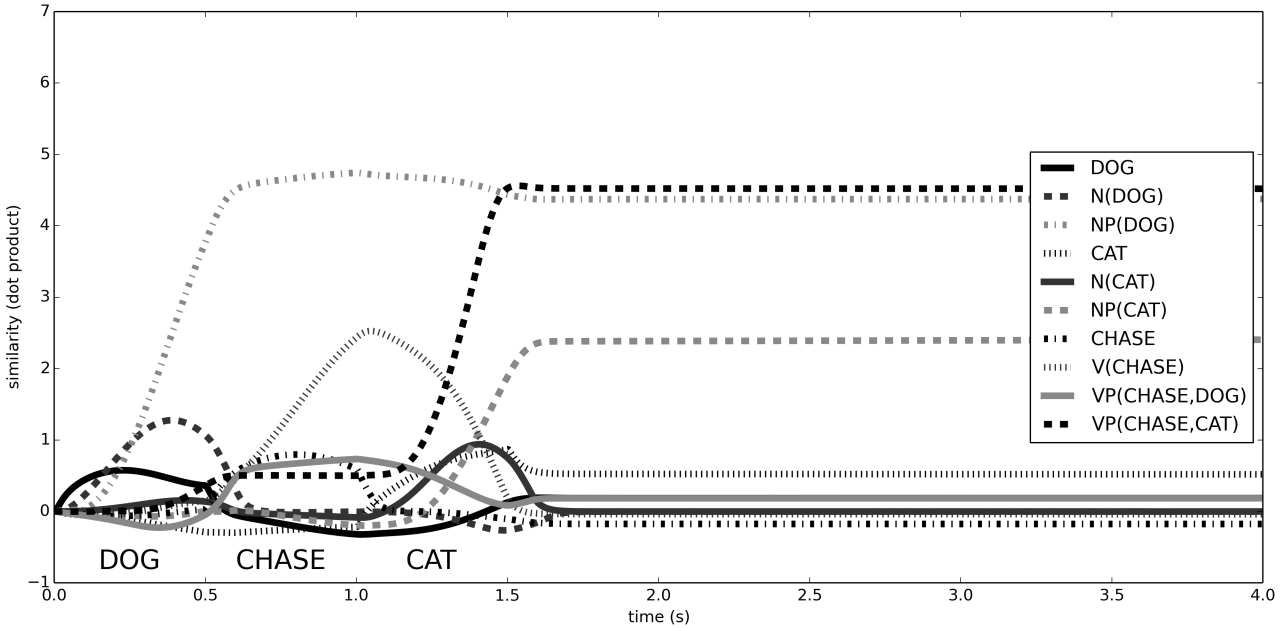


Figure 3: A distributed version of parsing. The state of the system is a single vector *x*. The lines are the dot product of x with the vectors for each node. The network behaves similarly to Figure 2 without requiring an exponential growth in node count.

Biological Recurrence

In the localist representation, the actual parsing process is done by changing the activity level of each node based on its connections and the activity levels of the nodes that connect into it. This can be thought of as a differential equation $dx/dt=f(x)$, where x is overall state vector (the activity level of each node) and f is a function that implements the effects of the connections.

This same idea is true for the distributed approach as well. We take each connection and replace it with a function. As discussed above, the connection causing the DOG node's activity to increase the activity of the N(DOG) node would be expressed as $dx/dt = (x \bullet \text{DOG})(\text{NOUN} \circledast \text{DOG})$. That is, we compute the similarity (dot product) of the current state x with DOG and multiply the result by NOUN⊛DOG. The result is the change in x caused by this connection. The change caused by all of these connections can be found by generating a single function that is the sum of each connection's function.

Now that we have expressed the parser as a differential equation, we can go one step farther and determine how biologically realistic neurons could implement that equation. That is, rather than dealing with the idealized connectionist nodes or simply computing the math of the distributed approach, we can create a model where each component is a spiking neuron, and the differential equation is approximated by the synaptic connections between neurons.

To do this, we use the Neural Engineering Framework (NEF; Eliasmith & Anderson, 2003). This provides a method whereby the activity in a group of neurons represents a vector, connections between groups of neurons implement functions on those vectors, and recurrent connections implement differential equations on those vectors. In each case, the neurons only *approximate* the desired function. Given a enough neurons, this approximation can be made arbitrarily close to the ideal function. However, given realistic biological constraints the resulting behaviour will not be identical to the mathematic version. This provides a natural competence/performance distinction.

To implement this parsing model using the NEF, we use a population of 4,000 LIF neurons. The activity of these neurons will represent a 128-dimensional vector. The neurons have randomly chosen biologically realistic properties in terms of their background current, sensitivity to input, and their "preferred" input stimulus (much like how neurons in visual cortex have particular visual stimuli to which they respond most strongly). This forms a distributed representation of our distributed vector.

Next, we transform the desired differential equation into a form that takes into account intrinsic neuron properties such as the post-synaptic time constant (the amount of time it takes neurotransmitters to be reabsorbed). For common recurrent connections in cortex, this is ~0.1s. The NEF can then be used to solve for an all-to-all recurrent connection weight matrix between all 4,000 neurons that will optimally approximate the given differential equation (Eliasmith & Anderson, 2003).

Importantly, this allows standard linear connection weights to approximate nonlinear functions. In this case, to implement multiplicative excitatory connections, we need $dx/dt = (x \bullet \text{V(CHASE)})(x \bullet \text{N(CAT)})(\text{VP(CHASE,CAT)})$ and other similar functions. These functions will be less accurately implemented than the ones for simple linear

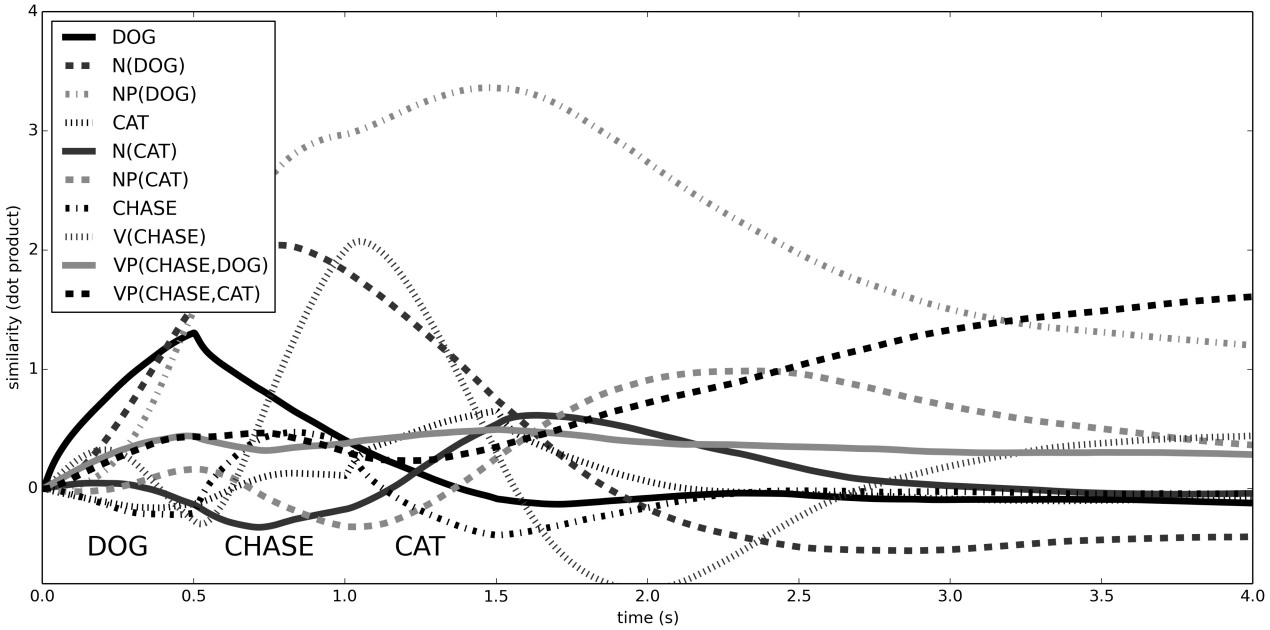


Figure 4: Parsing "DOGS CHASE CATS" in a network of 4000 leaky integrate-and-fire neurons, using distributed vectors of 128 dimensions. The input is the vector for DOG, then CHASE, and then CAT, for 0.5 seconds each. The graph shows the similarity of the represented vector x to the vectors for each indicated term. The network successfully parses the sentence.

connections. However, the ability to approximate this sort of complex connection allows for a wide range of new types of constraint models that we are only beginning to explore.

Semantic Pointer Architecture

It should be noted that the recurrent parsing networks described here use the same approach to representation as in our previous work with building neuron models to perform inductive reasoning, remember sequences, and exhibit cognitive control. This capability formed the core of Spaun (Eliasmith et al., 2012), the first large-scale brain simulation capable of performing multiple tasks. We call this general approach the Semantic Pointer Architecture (Eliasmith, 2013). The vectors are thought of as Semantic Pointers because they not only form a compressed representation of multiple pieces of information (so that the original information can be accessed by only having the compressed vector, much like a pointer in computer science), but also the vector itself contains similarity information, making it useful for making semantic decisions (so the vector value is not arbitrary, as in a computer science pointer).

In previous work, we have used a model of the cortex-basal ganglia-thalamus loop to control changes to these semantic pointers (vectors). In (Stewart, Choo, & Eliasmith, 2014), we showed that a left-corner parser could be implemented with this loop. However, the approach taken in this paper is to implement language processing with a dedicated recurrent network, rather than relying on the general-purpose (and less neurally efficient) basal ganglia. It is possible that this sort of system is involved in the dedicated language-oriented parts of the human brain.

Constraint Satisfaction in Recurrent Networks

Instead of implementing traditional rewrite rule grammars, as above, another way to think about parsing is that grammatical knowledge can be represented by a set of interacting constraints that favor and penalize the co-occurrence of certain structural features in the representation of a linguistic expression (Smolensky & Legendre, 2006). This approach can also be directly mapped into recurrent biologically plausible networks via the NEF.

To achieve this, we note that a constraint can be thought of as a bidirectional connection of the form seen in the distributed parsing model. That is, we can encode each constraint as a weighted outer product of two vectors (Smolensky et al., 2013). Then, we construct a transformation matrix that is sum of the outer products corresponding to entire collection of constraints under consideration. If, for example, there was just a positive constraint between two representations A and B, then the function the neural network needs to approximate would be:

$$dx/dt = (BA^T + AB^T)x$$

For more constraints, more outer products would be summed together. Each of these outer products can be thought of as a projection matrix that maps an input to a scaled version of a vector used to define the outer product. For instance, BA^T maps to B scaled by the dot product of the input x and A^T, by virtue of the linearity of matrix multiplication and the fact that the column-space of BA^T is all scalar multiples of B.

To build a biologically plausible implementation of this constraint satisfaction network, we use the NEF (Eliasmith & Anderson, 2003) in the same way as the previous example. 4000 neurons are configured to represent a 128 dimensional vector, and the NEF is used to find the optimal set of recurrent synaptic connection weights on all of those neurons that will best approximate this function.

The result of defining this mapping between the soft constraints and synaptic weights is that each pattern of neural activity in the population can be assigned a single scalar value (i.e. the value of a *harmony function*; Smolensky and Legendre, 2006) that reflects the degree to which the constraints in question are being satisfied. Over time, the state of the system will gravitate towards a position that maximizes this value and thereby involves a minimal degree of constraint violation. In the case that the constraints correspond to grammatical knowledge, one can think of this trajectory through the model's state space as a parallelized execution of the rules defining the grammar.

To test the scalability of this approach to performing constraint optimization in neural systems, we generate a vocabulary of 200 representations and generate random all-to-all constraints between them, yielding a transformation matrix encoding a total of 40,000 constraints. We then generate a neural population that computes the function described by the transformation matrix through its recurrent connections. Figure 5 depicts the similarity between the representational state encoded by this population and each of the 200 representations over time, as a fixed set of vocabulary items are presented as constant input. The stability achieved after approximately 300ms indicates that the system is able to rapidly compute a local solution to the problem of optimizing all 40,000 constraints.

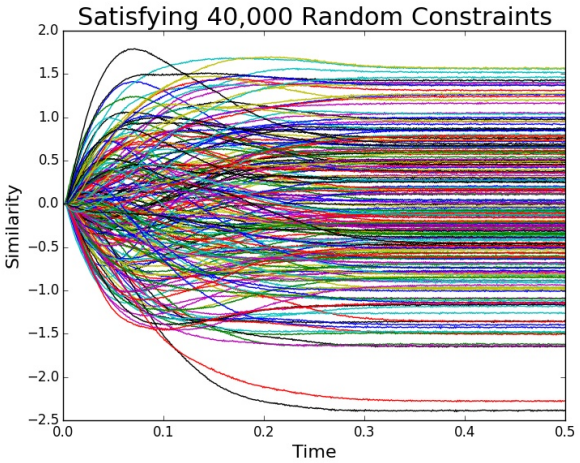


Figure 5: A recurrent network of 4000 neurons representing a 128-dimensional vector solving a randomly generated constraint satisfaction problem. The similarity of the resulting vector with the 200 basic vectors is shown. The network settles to a final result after ~300ms.

Completing Parse Trees

We can further use this approach to take partial parse trees and complete them. Given a set of rewrite rules for a grammar, we can identify grammatical constituents that can and cannot occur together. These form a set of constraints on the final vector representation. Traditionally, these constraints can be seen as connections between localist nodes. However, as noted in the previous section, these constraints can again be converted into differential equations that act on the distributed state vector.

To demonstrate this, consider the following toy grammar:

S → NP, VP	S → AUX, NP, VP
NP → DET, N	VP → V
VP → V	VP → V, NP

After building a network to implement these constraints, we can present partial tree to the system and it will generate the correct consistent components for the parse tree. For example, Figure 6 shows what happens when the input is the partial tree S(?,NP(DET,?),VP(?,V,?)). The network successfully identifies the missing components, resulting in S(AUX,NP(DET,N),VP(V)) as the final parse.

Stability Analysis of a Parser as a Dynamical System

Given that we use a recurrently connected neural ensemble to perform constraint satisfaction, we can treat the ensemble as a dynamical system and do a mathematical analysis of its behaviour. This behaviour is governed by the linear transformation matrix, T, where the computed function is

$$dx/dt = Tx$$

We can factor T into three matrices using eigen-decomposition:

$$T = S\Lambda S^{-1}$$

where S is an matrix whose columns are the eigenvectors of T, and Λ is a diagonal matrix containing the eigenvalues of T. Because the eigenvectors in S are orthogonal for symmetric matrices (and thus form a basis for the vector space), we can rewrite the starting state of the system as a linear combination of eigenvectors:

$$x_0 = c_1 v_1 + c_2 v_2 + ... + c_n v_n$$

where v_i is the i^{th} eigenvector, and c_i is a weight on this vector. This description of the initial condition of the system allows us to solve for the state of the system at arbitrary times given that the transform matrix simply scales each of its eigenvectors by a corresponding eigenvalue. If we treat the transform as a matrix differential equation (as is appropriate in the case that the constraint satisfaction process is implemented in neurons), we can then solve for the state of the system in the following manner:

$$x_t = c_1 e^{\lambda_1 t} v_1 + c_2 e^{\lambda_2 t} v_2 + ... + c_n e^{\lambda_n t} v_n$$

As time grows, eigenvectors with negative eigenvalues will disappear from x_t, while those eigenvectors with positive eigenvalues will exponentially increase; eigenvectors with an eigenvalue of zero will remain unchanged in terms of the contribution they make to x_t. Stable states are therefore acquired only when the transformation matrix has non-positive eigenvalues. Nonetheless, in the case that the eigenvalues are positive,

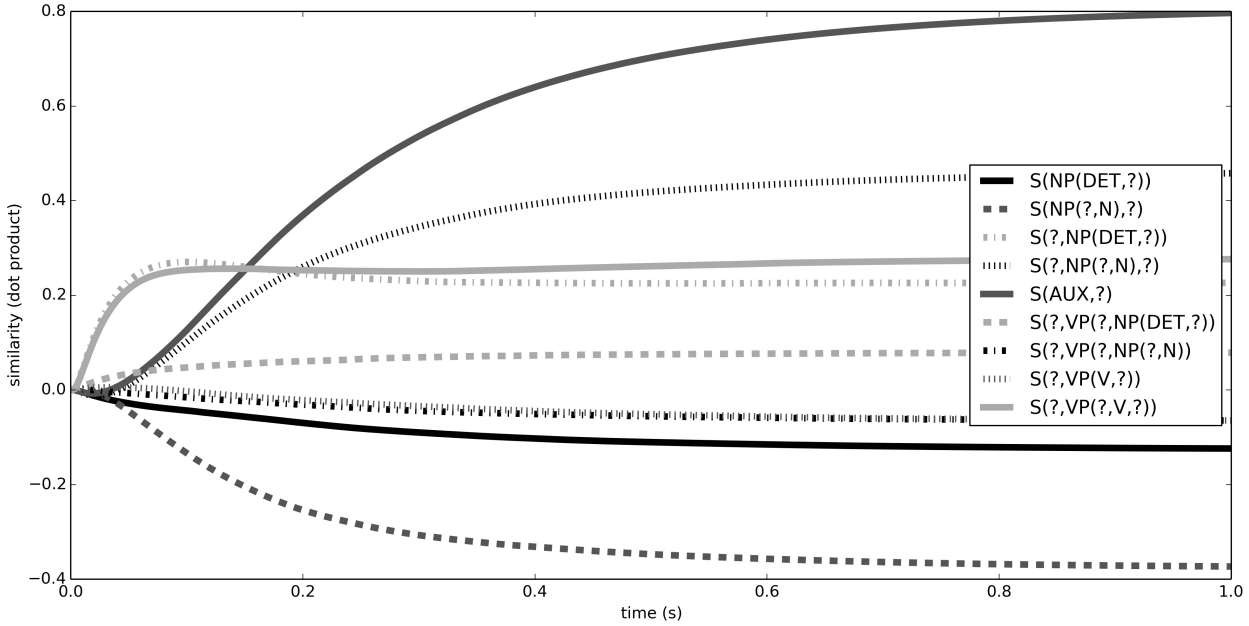


Figure 6: Completing a partial tree using grammar constraints encoded as recurrent connections in 4000 LIF neurons. Each possible grammar structure has its own vector, and the grammatical rules create positive and negative constraints. The input is the sum of the vectors for S(?,NP(DET,?),?) and S(?,VP(?,V,?)). After ~0.1 seconds, the two vectors for S(AUX,?) and S(?,NP(?,N),?) start to be represented, resulting in a stable parse of S(AUX,NP(DET,N),VP(V)).

the vector describing the system state converges on a particular direction in the vector space even as it grows without bound. Of course, when implemented using neurons with the NEF, this growth in numerical value will be balanced by the saturation behaviour that occurs when neurons reach their maximum firing limit.

The motivation for adopting this analysis is that it illustrates how our system might be harnessed to perform interesting computations with linguistic applications. In the case of parsing, the goal is to set the system to an initial state that encodes a set of words, and then have the dynamics of the system drive it towards a state that encodes an optimal parse of these words. The key insight offered by our analysis is that the representations over which our constraints are defined can all be represented as linear combinations of eigenvectors.

As such, we can predict which initial conditions will get mapped states that have particular degrees of similarity to the states that define particular representations. We could in theory choose eigenvalues that perform certain mappings of interest, and then reconstruct a transformation matrix with these eigenvalues. Overall, while more needs to be done to determine how sophisticated grammatical constraints can be encoded and processed using our recurrent network architecture, stability analysis of this sort offers a promising starting point.

Conclusions and Future Directions

Traditionally, grammatical knowledge has been thought of in terms of a set of fixed production rules that that are used to generate the sentences of a language. More recently, this knowledge has instead been characterized in terms of soft constraints on the well-formedness of linguistic expressions (Smolensky and Legendre, 2006). Our work suggests that it is possible to develop this latter approach to the study of grammatical knowledge in the context of detailed simulations of neural systems.

In order to extend our work to accommodate more sophisticated forms of language processing, a few outstanding problems need to be solved. First, it must demonstrated that the parsing capabilities of our simple dynamical systems can scale to more complex grammars. Second, a better understanding of the dynamic behavior associated with particular transformation matrices is required. For example, it might be useful to learn these matrices from examples of parse trees in much the same way that supervised learning techniques are used to train standard feed-forward networks. More generally, it would be very useful to be able to map desired properties of the system's behavior directly onto a set of constraints in a way that is consistent with what is known about how these constraints are likely organized.

Limitations aside, there are number of promising directions in which to extend this work. For example, many researchers are currently very interested in developing compositional distributional models of the meanings of arbitrary linguistic expressions (e.g. Socher et al., 2012). One possible approach to developing such models involves performing constraint optimization with recurrent networks over semantic features in addition to the syntactic features we are currently examining. Other interesting extensions involve incorporating hierarchical structure into the neural systems that perform constraint satisfaction, along with the incorporation of multiplicative interactions that modify the transformation matrix and allow the behavior of the recurrent network to be controlled by an external signal.

Overall, such extensions can help provide insight into the possible ways in which the sophisticated linguistic capabilities that are the hallmark of human intelligence are implemented in neural systems.

Acknowledgments

This work is funded thanks to the Social Sciences and Humanities Research Council of Canada and the U.S. Office of Naval Research.

References

Cottrell, G.W. (1985) Connectionist parsing. In *Proc. 7th Annu. Conf. Cogn. Sci. Soc.*, Erlbaum, 201–211.

Eliasmith, C. & Anderson, C. (2003). *Neural Engineering.* Cambridge: MIT Press.

Eliasmith, C., Stewart, T.C., Choo, X., Bekolay, T., DeWolf, T., Tang, Y., and Rasmussen, D. (2012). A large-scale model of the functioning brain. *Science, 338*:1202-1205.

Eliasmith, C. (2013). *How to build a brain.* Oxford University Press, New York, NY.

Gayler, R. (2003). Vector Symbolic Architectures Answer Jackendoff's Challenges for Cognitive Neuroscience, in Slezak, P. (ed). *Int. Conference on Cognitive Science*, Sydney: University of New South Wales, 133–138.

Plate, T. (2003). *Holographic Reduced Representations*, CSLI Publications, Stanford, CA.

Smolensky, Goldrick, and Mathis (2013). Optimization and quantization in gradient symbol systems: A framework for integrating the continuous and the discrete in cognition. *Cognitve Science, 38,*6, 1102-1138.

Smolensky, P. and Legendre, G. (2006). *The harmonic mind: From neural computation to optimality-theoretic grammar.* Volumes 1-2. Cambridge MA: MIT Press.

Socher, R., Huval, B., and Ng, A., and Manning, C. (2012). Semantic compositionality through recursive matrix-vector spaces. *Empirical Methods in Natural Language Processing.*

Stewart, T.C., Choo, X., and Eliasmith, C.. (2014). Sentence processing in spiking neurons: a biologically plausible left-corner parser. In *36th Annual Conference of the Cognitive Science Society*, 1533-1538.

Waltz, D. and Pollack, J. (1985). Massively parallel parsing: A strongly interactive model of natural language interpretation. *Cognitive Science 9,* 51–74.

Abstraction of analytical models from cognitive models of human control of robotic swarms

Katia Sycara[1], Christian Lebiere[1], Yulong Pei[1], Don Morrison[1], Yuqing Tang[1], and Michael Lewis[2]
[1]Carnegie Mellon University, 5000 Forbes Avenue
Pittsburgh, Pennsylvania, USA
{katia,cl,yulongp,dfm2,yuqing.tang}@cs.cmu.edu
[2]University of Pittsburgh, 4200 Fifth Avenue
Pittsburgh, Pennsylvania, USA
ml@sis.pitt.edu

Abstract

In order to formally validate cyber-physical systems, analytically tractable models of human control are desirable. While those models can be abstracted directly from human data, limitations on the amount and reliability of data can lead to overfitting and lack of generalization. We introduce a methodology for deriving formal models of human control of cyber-physical systems based on the use of cognitive models. Analytical models such as Markov models can be derived from an instance-based learning model of the task built using the ACT-R cognitive architecture. The approach is illustrated in the context of a robotic control task involving the choice of two options to control a robotic swarm. The cognitive model and various forms of the analytical model are validated against each other and against human performance data. The current limitations of the approach are discussed as well as its implications for the automated validation of cyber-physical systems.
Keywords: Cyber-physical systems; ACT-R cognitive models; Markov models; Robotic control

Introduction

As robotic platforms become more robust, teams of autonomously coordinating robots (robotic swarms) may be deployed for various tasks including environmental exploration, large-scale search and rescue, border protection, etc. One of the most important challenges in the design and deployment of such systems is making them amenable to effective human control. This requirement is complicated by the nonlinear dynamics of robotic swarm systems, the need to make realistic environmental assumptions, and the limitations and capabilities of human cognition. There has been much recent interest and research activity in control theory for formal system verification of safe operation of automation. In such work either the human has not been modeled at all, or the human has been modeled as a system disturbance. Modeling mixed human-autonomous systems where human cognition is taken into account is in its infancy. Formal and validated models of human-autonomous systems' safe operation, where the human element is modeled realistically, would be beneficial not only because these models would provide guarantees of performance, but also because they may uncover parts of the control space where human performance can deteriorate to unacceptable levels. Human cognitive limitations, the nonlinearity of the state-evolution dynamics of autonomously coordinating robots, and the high dimensionality of the joint state space of such systems preclude the possibility of a human maintaining or predicting the joint state of the whole system. Furthermore, the human may perform a broad spectrum of tasks ranging from reactive tasks, like manual control, to high-level deliberative tasks, like taking go/no-go decisions for a particular sub-mission. Cognitive modeling based on cognitive architectures such as ACT-R (Anderson & Lebiere, 1998; Anderson, 2007) has existed for many years. However, the resulting models are not in a mathematical form that is amenable to the techniques of formal verification. One way of meeting this challenge is creating an analytic model of human performance based on a cognitive model. Such an analytic model is cognitively compatible by construction, and because of its mathematical nature, is in the appropriate form for formal verification. In the case of a human operator controlling a robotic swarm, the analytic model can be integrated with a formal model that describes the swarm dynamics so that the overall mixed human-swarm system can be formally verified.

This paper presents the methodology of development of such an analytic model based on an ACT-R cognitive model. The task for which the cognitive model and the analytic model were constructed was the control of a robotic swarm simulation. The analytic model development process starts with data from human experiments. Human-in-the-loop experimentation supports the development and validation of descriptive cognitive models in two stages. Initial development and data collection from the simulation are used to bound expected performance and familiarize experimenters with the domain and its issues, as well as to constrain the task-independent control model to reflect general procedures. The data provides the experience needed to train the model using the Instance Based Learning (IBL) methodology (Gonzalez, Lerch, & Lebiere, 2003) in order to generate appropriate knowledge representations in memory in the form of control instances that guide decisions, as well as to tune general architectural parameters that modulate performance. Attentional routines can also be integrated to represent limitations in the speed and capacity of processing information in complex situations. Instances are grounded in specific situations, making them easy to learn through direct experience with the system in an automated process sometimes called chunking. Instances generalize dynamically to similar situations, providing predictions of performance in not previously experienced or partially experienced situations and resulting in situation-specific representations in short-term memory.

In building a (stochastic) state space model of a human the primary challenges are defining the relevant states and transforming the human constraints in the neuroscience and psychology literatures into state space constraints. We use the cognitive model as a proxy for the human operator and run simulations to produce the decisions made by the model as a function of operator cognitive state and cognitive limitations. The methodology and resulting model will be described in detail in the rest of this paper.

Experiment Task

The human-swarm system studied followed that described in (Bullo, Corts, & Martinez, 2009). Participants control a swarm of twenty simulated robots in a web interface. No control can be exerted over individual robots, only over the swarm as a whole, and only by the choice of one of two strategies controlling how the robots collectively move: Rendezvous or Deploy. The two strategies correspond to two different algorithms for the evolution of the robots' motions, Rendezvous causing the positions to largely converge, and Deploy to largely diverge. In addition to the robots themselves, the simulated environment also contains a set of fixed obstacles. Each of the sixty trials begins with a set of initial positions of the robots, and of the obstacles. These positions were sampled from bivariate Gaussian distributions, a different pair of distributions used at each trial. The means and variances of these distributions were themselves samples from a uniform distribution. While each participant saw roughly the same sets of positions, in the same order, a small amount of noise was introduced into each.

The interface presents the initial positions of the robots and the positions of the obstacles, and solicits a choice of Rendezvous or Deploy from the human. The robots then move according to that strategy, and leave a visual trail of where they have been (Figure 1). The interface also displays direct feedback in the form of a number representing the percentage of the environment's area that the ensemble of robots has covered. The human's goal in each trial is to select the strategy that can be expected to result in the larger coverage, for that set of initial robot positions and obstacles.

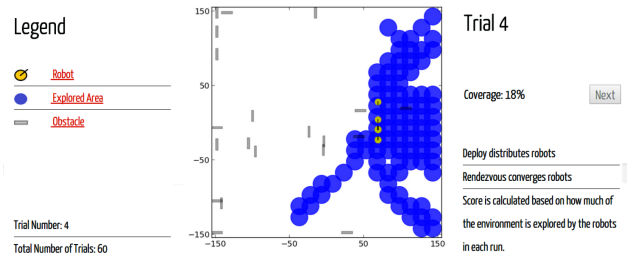


Figure 1: The interface to the simulated swarm experiment.

Fifty participants were recruited within Amazon Mechanical Turk[1], of whom forty-eight completed the experiment. Each of the forty-eight participants was presented with sixty trials. The first ten were training trials. In these training trials participants were asked to choose Rendezvous or Deploy and observe the resulting coverage. They were then asked to choose the forgone strategy and see its resulting coverage. After the training trials they were only able to select one of the strategies, and saw only the coverage result for the chosen strategy, with no feedback on the forgone strategy. Each of the first thirty post-training trials were unique, but the last twenty were alternately unique, and recapitulations of the training trials, modified slightly by the addition of noise. The participants were not told that there would be such recapitulated trials.

[1] https://www.mturk.com/mturk/welcome

Cognitive Model

The cognitive model is implemented in the neurally-inspired cognitive architecture ACT-R and follows the instance-based learning (IBL) methodology. In IBL decisions are made primarily based on experiences of a task. In our model these experiences are stored in the chunks of ACT-R's declarative memory, each such chunk corresponding to a relevant experience. Instance chunks typically contain a description of the context in which each decision is made, the decision itself, and the outcome of that decision. The mean initial position of the robots (eccentricity) and its variance (dispersion) were used to characterize the context of each trial. Other dimensions were considered, especially characterizing the distribution of obstacles, but were found upon further analysis both relatively inconsequential to the outcome and generally ignored by human participants. Both possible decisions were represented in each instance chunk, with the outcome in terms of coverage for each action stored in dedicated slots. Each instance chunk therefore contains four slots mapping the eccentricity and dispersion of the robot swarm to the coverage percentage for the Rendezvous and Deploy actions.

Before the model proper begins executing, chunks containing the ground truth values of both the Rendezvous and Deploy values for each of the ten training trials are added directly to declarative memory. For each of the fifty non-training trials of the experiment the model is presented with the eccentricity and dispersion values, and estimates, from the chunks stored in declarative memory, expected coverage fractions for Rendezvous and Deploy. These estimates are generated using ACT-R's blending mechanism (Lebiere, 1999), using partial matching of the chunks representing instances that are already in memory. This partial matching is done using a linear similarity function between the eccentricity and dispersion values, as well as the usual ACT-R declarative memory retrieval's activation computation, including recency and noise. The model selects as its decision whichever of the two actions produces the larger expected coverage percentage. The model then receives as feedback the ground truth coverage for the chosen Rendezvous or Deploy action, and registers it instead of its estimate in its representation of the current trial. The coverage value in this chunk for the forgone

option is, instead of the ground truth, left as the model's estimate. Upon completion of the trial the representation of the problem is added as a new chunk in declarative memory. The model thus starts out with ten instances, those from training, and builds up to sixty by the conclusion of the experiment as its experiences accumulate.

The ACT-R model is stochastic, and was run 1,000 times to generate stable estimates, each with a distinct random number seed. Most ACT-R parameters were left at their default values. The main deviation from standard values was to set the activation noise parameter to a relatively high value of 0.75 to reflect the high stochasticity of decisions made by the Mechanical Turk subjects. For each run declarative memory is reinitialized with just the ten training trials, and the full set of sixty instances is built up afresh, with potentially different values in them reflecting the stochasticity of the model's judgment at each step, and most specifically the fact that it receives feedback on only its chosen option and its potential implications for the dynamics of its behavior ((Lebiere, Gonzalez, & Martin, 2007)). The results are aggregated both for comparison to the human results and for constructing the Markov model.

Abstraction Procedure

The knowledge state in IBL models is characterized by the set of instance chunks and their activation. The evolution of the cognitive state as the model accumulates experiences can be thought of as a k-dimensional discrete-time signal, which is the time-trajectory of activation levels of the different memory chunks through various decision cycles, in response to particular inputs.

As stated before, an IBL memory chunk in ACT-R consists of a representation of the context and outcome of the control actions. In this setting, context involves the centrality and dispersion of the robots, while the outcome involves a representation of the percentage coverage achieved by the available decisions. Environment and system observations change the activation levels of the existing memory chunks as well as add new chunks to the model, thus reflecting the system state as observed by the operator. Abstracting that distributed state of knowledge contained in the cognitive model's declarative memory in an analytical model requires coarsening it into discrete states, such as the degree of preference toward one strategy or the other. To reflect the context-sensitive nature of the IBL decision process, distinct sets of states are created for each context neighborhood. The number and nature of the states is left to the modeler. The changes that each experience causes to the activations (and number) of instance chunks in memory are reflected in a probabilistic transition in the analytical model. The transitions in the analytical model are trained from Monte Carlo runs of the cognitive model.

For this specific model, our approach starts with an interface to the multi-robot system that explicitly represents two actions, "Deploy" and "Rendezvous", and the percentage coverage obtained by the action. In this setting, the decision context involves the centrality and dispersion of the robots. The memory chunk also contains a representation of the action taken as well as the action not taken, and the resulting and expected outcome, respectively. Environment and system observations change the activation levels of the memory chunks in the cognitive model, thus reflecting the system state as observed by the operator. Previously created chunks decay with time and are reinforced with retrieval, while new chunks are added to reflect recent observations. The time-trajectory of the activation levels of the memory chunks can be clustered to produce the Markov model. States of the Markov model are defined to correspond to a pattern of memory chunk activation levels. In this domain, they would correspond to a temporary preference for one action over the other. In general there can be k events that correspond to the consistent states of the system as observed by the operator.

Markov Model

Following the approach described in (Gray, 2002), we employ a Markov model as the analytic model for ACT-R cognitive processes of human control. Let D denote the selection of Deploy and R denote the selection of Rendezvous. Specialized for the human-swarm task, the overall Markov model of the cognitive processes is decomposed into two sub-Markov Models indicated by superscripts in the edges of the graphs that correspond to two basic outcomes of the action chosen (see right-hand side of Figure 2): a) Model U: the ground truth coverage is larger than the estimation of the ACT-R model; b) Model L: the ground truth coverage is less than or equal to the estimation of the ACT-R model. Model U is parameterized by four probabilistic action selection transitions: 1) $p^U_{D\to R}$, 2) $p^U_{D\to D}$ (where $p^U_{D\to R}+p^U_{D\to D}=1$); 3) $p^U_{R\to D}$, and 4) $p^U_{R\to R}$ (where $p^U_{R\to D}+p^U_{R\to R}=1$). Symmetrically, Model L is also parameterized by four probabilistic state transitions: 1) $p^L_{D\to R}$, 2) $p^L_{D\to D}$(where $p^L_{D\to R}+p^L_{D\to D}=1$); 3) $p^L_{R\to D}$, and 4) $p^L_{R\to R}$ (where $p^L_{R\to D}+p^L_{R\to R}=1$). The switching between these two sub-models is parameterized by the probabilities $p_{Grd>Est}$ and $p_{Grd\leq Est}$ where $p_{Grd>Est}+p_{Grd\leq Est}=1$ and where Grd is the ground truth and Est is the ACT-R estimation. As a result, we establish a Markov model of action selection assuming that any action selection is independent of the history given the previous action and the chosen sub-model (either U or L).

To situate the Markov model in the human-swarm environment, we discretize the observation space (the dispersion and eccentricity) as a grid of cells (see the left-hand side of Figure 2). Each cell in the grid is associated with an overall Markov model as described above.

After the ACT-R model generates the data, the training and prediction test procedure is as follows: (1) locate the cells to which the data instances belong; (2) train a Markov model for each cell; (3) make predictions for each instance in test data based on the Markov transition probabilities.

Training Procedure

Situated in the grid of the environment, a Markov model (see the right-hand side of Figure 2) is trained for each cell.

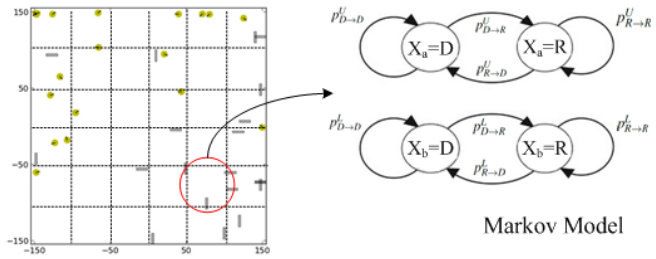


Figure 2: The framework for the training procedure in the Markov model.

The nodes $X_{a/b} = D$ indicate the selection is Deploy and the nodes $X_{a/b} = R$ indicate the selection is Rendezvous. In each cell, the data is represented as a sequence of selections $X = \{x_1, ..., x_t\}$. During the training procedure, scanning through the sequences of action selections X, we record the counts of action selection transitions (x_i, x_{i+1}). Formally, the count $c^k_{s_i \to s_j}$ is the number of transitions from selection s_i to s_j ($s_i, s_j \in \{D, R\}$) of the sub-model k (where $k \in \{U, L\}$). For example, when an action selection transition $(D, R) \in X$ is encountered — the current selection is Deploy and the next selection is Rendezvous — and the feedback is that the ground truth is less than the estimation (in sub-model L), then the count is updated as: $c^L_{D \to R} \leftarrow c^L_{D \to R} + 1$. After counting all the instances in the data set, these counts are normalized into the transition probabilities following:

$$p^k_{s_i \to s_j} = \frac{c^k_{s_i \to s_j}}{\sum_{s_t \in \{D,R\}} c^k_{s_i \to s_t}}. \tag{1}$$

In addition, the switching probability $p_{Grd>Est}$ ($p_{Grd \leq Est} = 1 - p_{Grd>Est}$) between the sub-models is estimated by simply computing the ratio of the times that the ground truth coverage is larger than the estimation of ACT-R over the times that the ground truth coverage is less than or equal to the estimation of ACT-R. It can be proved that the above parameter estimation process computes the model parameters that maximize the posterior probability of generating the data conditional on the parameters.

Prediction Procedure

After the training procedure, we obtain the overall Markov model, which is characterized by the probabilities from Deploy to Deploy $p_{D \to D}$, from Deploy to Rendezvous $p_{D \to R}$, from Rendezvous to Deploy $p_{R \to D}$ and from Rendezvous to Rendezvous $p_{R \to R}$, as a switching mixture of the two sub-models (Model U and Model L):

$$p_{D \to D} = p_{Grd>Est} * p^U_{D \to D} + p_{Grd \leq Est} * p^L_{D \to D} \tag{2}$$

$$p_{D \to R} = p_{Grd>Est} * p^U_{D \to R} + p_{Grd \leq Est} * p^L_{D \to R} \tag{3}$$

$$p_{R \to D} = p_{Grd>Est} * p^U_{R \to D} + p_{Grd \leq Est} * p^L_{R \to D} \tag{4}$$

$$p_{R \to R} = p_{Grd>Est} * p^U_{R \to R} + p_{Grd \leq Est} * p^L_{R \to R}. \tag{5}$$

This overall model can be exploited to predict the next action selection s_{i+1} of the human players given the current action selection s_i following the decision rule:

$$s_{i+1} = \arg\max_{x \in \{D,R\}} p_{s_i \to x}.$$

For example, if the overall model states that $p_{D \to D} > p_{D \to R}$, and the current selection is Deploy, the predicted next action will be Deploy; otherwise, the next prediction is Rendezvous.

Results

The resulting Markov model is evaluated by two measures: **Accuracy** and **MSE** (Mean Square Error). The **Accuracy** is defined as:

$$Accuracy = \frac{|\mathbb{I}(SEL_{\text{pred}}, SEL_{\text{ACT-R}})|}{|\text{trials}|} \tag{6}$$

where $\mathbb{I}(SEL_{\text{pred}}, SEL_{\text{ACT-R}}) = 1$ if the prediction selection is the same as the ACT-R selection; otherwise $\mathbb{I}(SEL_{\text{pred}}, SEL_{\text{ACT-R}}) = 0$. And **MSE** is defined as:

$$MSE = \frac{1}{|\text{trials}|}(P_{\text{Markov-Grd}} - P_{\text{ACT-R-Grd}})^2 \tag{7}$$

where $P_{\text{Markov-Grd}}$ is the precision of the Markov model (i.e. the Markov model conforms with the ground truth) and $P_{\text{ACT-R-Grd}}$ is the precision of the ACT-R model (i.e. the ACT-R model conforms with the ground truth).

We evaluate our Markov model given the following discretization of the observation space: 17×17, 10×10, 5×5, 3×3 and 1×1. The model prediction performance is shown in Table 1. From Table 1, we can see that the performance improves as the granularity of the grid is increased but reaches a plateau around a 5×5 grid, which is a plausible discretization level. The limit on accuracy of about 75 percent fundamentally reflects the variability of human decisions. The limit on mean square error fundamentally reflects the discretization of the problem space and other factors averaged over by the Markov model training procedure.

Number of cells	Accuracy	Mean Square Error
17×17	75.07%	0.06718
10×10	74.90%	0.07671
5×5	74.48%	0.08449
3×3	69.61%	0.11254
1×1	52.99%	0.29963

Table 1: The prediction results of different numbers of cells.

Figure 3 presents the trial-by-trial performance of the human participants, the cognitive model, and three versions of the analytical model using various degrees of state coarseness. The cognitive model generally captures quite well the pattern of fluctuations of human performance across trials. The fluctuations reflect both the impact of previous outcomes,

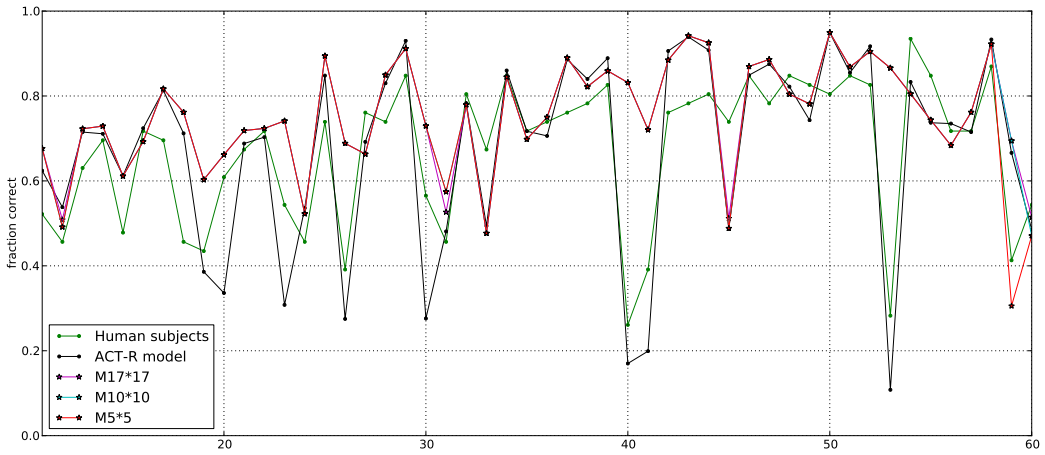


Figure 3: Fraction of decisions correct by trial number

which are captured in the cognitive model by the addition of a new chunk for each experience as well as the recency effect from activation decay, and the effect of each new trial context on the decision. The analytical models show occasional deviations from that pattern, which reflects the coarse nature of their state space and other simplifying factors. A learning effect can be observed in the increasing consensus across runs for each action choice, whether correct or incorrect.

Figure 4 graphs the fraction of choice of a specific option (Rendezvous) as a function of the difference in coverage between that option and the alternative (Deploy) in the ground truth data. The sharp sigmoid curve centered around the origin fit to the data indicates that both human participants and cognitive model learn to perform the task quite well, and nearly identically. Their errors primarily reflect contexts in which the two actions provide very similar performance. The analytical models are also sensitive to differences in coverage, but not nearly as sharply as their sigmoid fits are much flattened. This presumably reflects the coarse state representation that aggregates nearby contexts in identical bins as opposed to the more graded similarity-based partial matching of the cognitive model. In addition, when limited to 3x3 cells, the analytical model shows an inability to converge to the same certainty as the cognitive model for large differences in coverage.

Figure 5 graphs the pattern of choices in the two-dimensional context space of robot dispersion (x-axis) and eccentricity (y-axis). Green circles are associated with a correct choice of Rendezvous, and are typically associated with large dispersion values, while yellow circles are associated with a correct choice of Deploy, and are typically associated with small dispersion values. The size of the circle represents the probability of choosing the correct action. Larger choice probabilities are typically seen for extreme dispersion values, while smaller probabilities are seen for mid-range dispersion

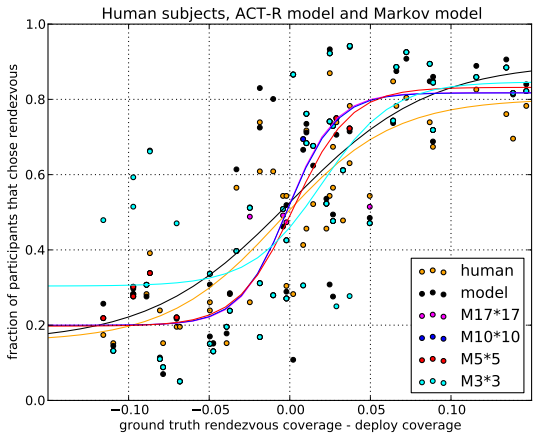


Figure 4: Fraction choice of the Rendezvous action as a function of the difference in percent coverage between Rendezvous and Deploy

values that correspond to the boundary between the two domains where the difference between the two actions is small. For each trial, circles centered on the same point are plotted for both human and cognitive model choices. Most pairs of circles overlap perfectly but specific discrepancies between human and model choice are visible, corresponding to trials 18, 20, 23, 30, 41, 45 and 59 (see corresponding data on Figure 3). All those trials are located in the boundary region where small differences in perception or experience might easily make the difference between choosing one action over the other.

Conclusion and Future Work

This approach can be understood as one of incremental abstraction in model development. We start with the full detail

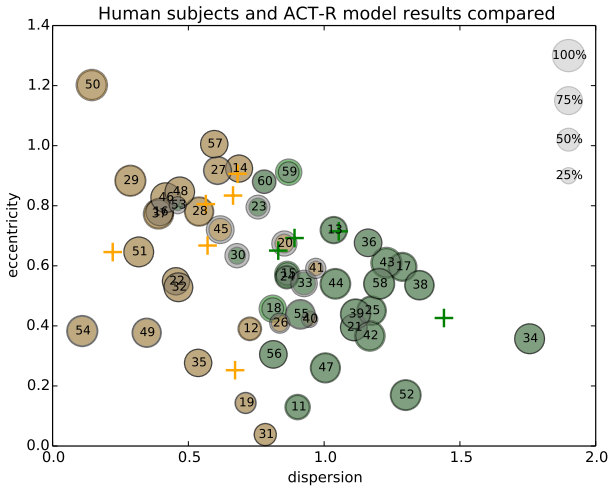


Figure 5: Performance of human participants and cognitive model (circles) graphed in the two-dimensional context space for Rendezvous (Green) and Deploy (yellow) trials. Crosses indicate training trials.

of the human data. In this case, it included only the choice between two competing actions. However, it could also include latency to make the decisions, individual variations, or any other observable relevant aspect of human performance. A process model of human control of the cyber-physical system is then developed using a cognitive architecture. The benefit of using a cognitive architecture is that it already includes many constraints on performance that do not need to be rederived from data. A further advantage of using the IBL modeling methodology is to further limit modeler choices and improve the automated nature of this approach by limiting modeler decisions to the representation of the context. Further abstraction can then be achieved by specifying the structure of a formal analytical model, such as the cells and states of the Markov model used here. Unlike limited and noisy human data, the cognitive model can then be run as many times as needed and the resulting data used to train the formal model to the needed accuracy. The models can be validated against each other and against the human performance data at each level of development: (a) the initial cognitive model can be compared to the human data that it is meant to capture, (b) the formal analytical model can be compared to the cognitive model from which it is abstracted, and finally (c) the formal analytical model can be validated against the human performance data.

One important question is which aspects of the cognitive model performance can be readily captured by this approach? We saw that in this domain the model's Markovian assumption was quite accurate at capturing the impact of experience on decisions. Coarsening the high-dimensional nature of declarative memory representation into a limited number of states can lead to some distortions but seems fairly accurate if the state space is above some minimum threshold. Another limitation is the need to restrict contextual generalization to an all-or-none division into independent cells. Another important question is how to generalize the Markov model for more realistic applications which have 1) larger action space (more than 2 actions), 2) higher dimension of the observation space (more than 2 observed parameters), and 3) more sophisticated performance dependency over action selections and environment observations.

Finally, the analytical model is trained on the entire data set generated by the cognitive model, including the model's initial learning curve. As it is, the analytical model is akin to a representation of average or asymptotic performance. More contextual elements would have to be added to enable a representation of cognitive learning processes in the analytical model.

Our future work involves two parallel thrusts. We want to generalize our approach to modeling human control of other cyber-physical processes to test its breadth of applicability. Also, we need to incorporate the resulting analytical models into formal verification frameworks, e.g. (Oishi, Mitchell, Bayen, & Tomlin, 2008), that can be used to derive formal guarantees on the human control of cyber-physical systems.

Acknowledgments

This work is funded by NSF awards CNS1329986, CNS1329762 and CNS1329878 to Katia Sycara, Michael Lewis and Christian Lebiere.

References

Anderson, J. (2007). *How can the human mind occur in the physical universe?* New York, NY: Oxford University Press.

Anderson, J., & Lebiere, C. J. (1998). *The atomic components of thought*. Mahwah, N.J.: Erlbaum.

Bullo, F., Corts, J., & Martinez, S. (2009). *Distributed control of robotic networks: A mathematical approach to motion coordination algorithms*. Princeton University Press.

Gonzalez, C., Lerch, J. F., & Lebiere, C. (2003). Instance-based learning in dynamic decision making. *Cognitive Science*, *27*, 591–635.

Gray, R. (2002). markov at the bat: A model of cognitive processing in baseball batters. *Psychological Science*, *13*(6), 542–547.

Lebiere, C. (1999). The dynamics of cognitive arithmetic. *Kognitionswissenschaft*, *8*(1), 5–19.

Lebiere, C., Gonzalez, C., & Martin, M. (2007). Instance-based decision-making model of repeated binary choice. In *Proceedings of the 8th international conference on cognitive modeling*. Ann Arbor, MI.

Oishi, M., Mitchell, I. M., Bayen, A. M., & Tomlin, C. J. (2008). Invariance-preserving abstractions of hybrid systems: Application to user interface design. *Control Systems Technology, IEEE Transactions on*, *16*(2), 229–244.

A Method for Building Models of Expert Cognition in Naturalistic Environments

W. Korey MacDougall (KoreyMacdougall@gmail.com)
Matthew Martin (MattMartin256@gmail.com)
Nathan Nagy (n.c.nagy1@gmail.com)
Robert L. West (rlWest@gmail.com)
Institute of Cognitive Science, Carleton University, Canada
Ottawa, ON K1S5B6 Canada

Abstract

We discuss a method for creating models of expert cognition and behavior in naturalistic environments. The method consists of video annotation and iterative model tracing, as well as a commitment to making all data and components of the model available for independent validation.

Keywords: Cognitive modeling; macro-cognition; expertise; expert cognition; methodology.

Introduction

Cognitive modeling draws largely upon theories and methods from psychology and artificial intelligence. While the insights and approaches developed in these fields are leveraged to great positive effect by cognitive modelers, there are limitations that arise from an overly strict adherence to the practices of these sister fields. Among these is the difficulty of modeling behavior and cognition in naturalistic environments by using laboratory-based methods. The reasons for this issue, and the approach we are using to address it are discussed below.

Laboratory Methods and Macro-cognition

Studying behavior and cognition in real-life scenarios is difficult. Environments can be chaotic, behavior can be inconsistent, and there can be an overwhelming number of variables involved. Laboratory experimentation is largely aimed at limiting this complexity and helping researchers to isolate and examine more precisely the factors they are interested in. This is achieved through simplification of the task environment, repetition of tasks, and the averaging of data across trials and participants.

This methodology has proven powerful and generative, but questions have been raised about the difficulties of "scaling up" findings from laboratory experimentation to explain cognition as it occurs in the real world (Klein et al., 2003). These limitations are particularly relevant in the study of expert behavior and cognition. In natural environments, expert behavior is dynamic, adaptive, and often idiosyncratic. These are important elements that are often obscured by traditional laboratory paradigms.

As an example, consider the difference between the study of chess players as it traditionally occurs in laboratory paradigms and the observation of a chess master playing a game at home. In lab studies, players may be presented with chess positions and asked to recall as many pieces as possible, after which reaction times and error rates can be measured. Alternatively, they may be shown a position and asked to make a single move, then asked how many candidate moves were considered, or to speak aloud while deciding what move to play. This is a form of protocol analysis commonly used to supplement experimental work (Ericsson, 2006).

Using the above methods, the motivation is often to examine how many elements can be "chunked" into a single memory representation, or to examine how deeply or broadly a player searches in choosing their next move. This is a reductive approach that splits the task performance into pieces in order to isolate and better understand those pieces, with the (possible) intention of later combining the components into a more holistic picture of the underlying cognition. This subsequent recombination can be directed in two ways: either to form a more complete picture of chess cognition *per se*, or to inform a broader theory of cognitive functions, such as memory encoding or pattern recognition, which in the general case are not specifically tethered to chess. Part of our motivation in creating the method described here is skepticism about whether this process of division and subsequent recombination can lead to an understanding of the cognitive system as a whole, particularly as it applies to understanding situated, real world, expert cognition.

The tension between the power of laboratory methods and the complexity of situated expertise has been addressed in a number of ways (Kieras & Meyer, 2000; Klein et al., 2003; Williams, 2006; West & Nagy, 2007). One approach that is useful in the context of cognitive modeling is to distinguish between micro-cognition and macro-cognition. Micro-cognition refers to those mental operations that are typically studied in cognitive psychology experiments and which are thought to be invariant and underlie all of cognition (Klein et al., 2003). These include such functions as memory encoding and retrieval, and serial versus parallel attentional mechanisms (Klein et al, 2003). Macro-cognition, on the other hand, refers to cognition as it occurs in naturalistic environments, and includes such high-level operations as complex decision making, resource allocation, team co-ordination, and responding to non-routine circumstances (Schraagen, Militello, Ormerod, & Lipshitz, 2008; West & Nagy, 2007).

The method we are using is aimed at elucidating macro-cognitive processes and therefore eschews the siloing of the

component functions, as is common in studies of micro cognition. Instead, we observe the expert performing the task as they naturally would, with minimal coaching or restrictions, and attempt to identify the components of the task afterward, in collaboration with the experts. We wish to point out that we are not creating this as an *alternative* to lab based micro-cognition research. Instead, our approach is intended for use *in conjunction with* traditional micro-cognitive research methods.

Method

Motivation, Philosophy and Scope

There are two guiding motivations for this methodology. The first is the belief in the value of unification in modeling, and the second is what we perceive as the importance of integrating and making explicit the relations between experimental design, data analysis, and theory construction. Concerning the first motivation, we agree with Newell's (1973) argument that, if we are ever to understand cognitive systems, the research community must attempt to integrate its efforts and avoid an unbounded proliferation of unconnected models.

As for the second point, we want to encourage explicit attention to the relations between theory and method in cognitive modeling and experimental work. In particular, for this discussion it is important to distinguish between (1) systems for building computational models (e.g., ACT-R, GOMS, SOAR); (2) methods for creating models using these systems (e.g., task analysis, cognitive walkthroughs, ethnology); and (3) methods for evaluating the resulting models (e.g., hypothesis testing, model fitting). What we are proposing is a method for *evaluating* models. It can be used with any computational modeling system and any methodology for generating models within these systems.

The intended scope of this methodology is the study of real world expertise, specifically in those domains for which some, but not all, behavior and cognition is routine. In the case of novices, behavior is generally too variable to be modeled using this approach. As practitioners develop proficiency in their domain, they generally converge upon optimal solutions (Shanteau & Hall, 2001), and thus we observe more consistency at higher levels of expertise than at lower levels (note: this methodology is not intended for "creative" expert domains such as music composition or fiction writing, where no convergence on an optimal process is expected). Due to factors such as chaotic environments, individual differences, the actions of co-workers, unexpected events, and/or the need to multi-task, experts in the same field do not always behave in the same way. We are interested in the middle ground between behavior that is fully routine and repetitive on the one hand, and that which seems entirely unsystematic on the other. We argue that this is the zone in which most real-world experts operate.

Our methodology is more akin to systems engineering practice than it is to experimentation. We are not attempting to generate and test hypotheses as we would in lab-based experiments. Rather, we are attempting to develop and refine models until they adequately capture the range of relevant behaviors and cognitive operations. This approach is more consistent with a Lakatosian scientific framework (Lakatos, 1970) than with a Popperian one (Popper, 1963). In short, the process of evaluating our models rests upon iteration rather than falsification.

While we respect the importance of falsifiability in theorizing, we must also be clear about when it is appropriate or possible. For example, it is problematic to use falsification to evaluate the validity of cognitive architectures, such as ACT-R, GOMS, or SOAR (Cooper, 2007; Newell, 1973). Although some models built in these systems can be falsified, the architectures generally cannot be falsified because there are usually multiple ways to model the same task within a single architecture. In other words, the model can be adjusted to fit the data.

Likewise, we argue that falsification is problematic for evaluating models of real world expert behaviour, but for different reasons. Specifically, although we are concerned with evaluating specific models, and not the architectures in which they are built, the naturalistic behaviour of experts across time is different each time they are observed, even for the same individual. Of course, many of the component behaviours, or unit tasks, are the same from scenario to scenario, and these can be isolated and studied in the lab, but this is not what we are evaluating. Our interest is in evaluating whether a model can realistically account for the sequence of decisions and behaviours as each different scenario unfolds. Lab-based hypothesis testing is inappropriate here because it is based on averaging across the same sequence of behaviours repeated within and/or across individuals, with no variations in the environment.

Lakatos (1970) defines a program of research as scientific if it is making progress over time, where progress may be demonstrated in multiple ways. For example, progress may include the discovery of new phenomena, falsification, hypothesis confirmation, increased parsimony, theory unification, counterfactual predictions, etc. Our method is based on two criteria for progress: (1) an increase in the amount of data accounted for by the model, and an increase in the percentage of times the model correctly predicts the next action of the human expert, and (2) an increase in the scope of the model, i.e., as we collect more and more samples of expert behaviours, the same model must cover all of them without any parameter changes.

Overview

The method we have been developing is an iterative, collaborative approach to creating macro-cognitive models. The process involves recording video footage of experts performing in naturalistic environments then using this information to construct a model of the task. Using this model as a base, we use an iterative model-tracing procedure to improve it until it is sufficiently robust to predict all or most of the high-level behavior observed. We do this using freely available tools and make our data and

models available to other interested parties. The process is laid out in more detail below.

Procedure

1 - Video capture The first step is to collect video footage of experts performing in a naturalistic environment, as well as documentation about the task and interviews with experts. Rather than constructing a simplified task environment and attempting to isolate components of the task performance, we aim to have the experts demonstrate their skills in the messiness and complexity to which they are accustomed and which forms the necessary background against which their training and expertise are normally expressed. This step is similar to techniques used in cognitive task analysis (Kieras & Meyer, 2000) and cognitive ethnography (Williams, 2006).

2 - Task Model Construction Once video footage has been collected, we review it with the experts and attempt to determine patterns and regularities in task performance. We ask the experts to tell us what their goals and sub-goals were at each given point, what their strategies for accomplishing these goals were, and to identify which elements of the environment were relevant in their decision making (note that this process can be begun before video data is collected).

3 - Cognitive Model Construction Once we have created a task model, we construct two separate but inter-related models: a cognitive process model capable of completing the tasks, and a perceptual model, which we have termed the situational awareness (SA) model, that describes what the agent pays attention to in the environment and how these environmental cues are combined into a meaningful interpretation. These two models are linked in that the process model relies on the SA model for a meaningful interpretation of the environment and the SA model relies on the process model to provide context (e.g., in terms of the current goals of the agent), which is used to interpret raw environmental cues to create situational awareness.

The framework we are using to inform this step is called Sociotechnical GOMS, or SGOMS (MacDougall, West, & Hancock, 2012; West & Nagy, 2007), which is an extension of the GOMS modeling framework (John & Kieras, 1996). However, our method is not necessarily tied to any particular theory of cognition and therefore we will not discuss SGOMS in detail.

4 – Video Annotation and Model Tracing Once we have constructed the two models, we use them to annotate the video footage we have collected. We identify which actions are being undertaken at each point in the video, what the current goals and constraints driving behavior are, and which elements of the context are relevant in decision making. As these are determined, we note on the video which actions are being undertaken and specify their time course. To create these annotations we are using the ANVIL Video Annotation software (Kipp, 2010). See Figure 1 in for a screenshot of video that has been annotated in ANVIL.

Figure 1: ANVIL-annotated video frame of gameplay.

The annotation procedure is accomplished through an iterative process of model tracing. To do this, we first annotate the video by noting when behavioural elements related to the cognitive model appear on the video. Then we annotate the video with regard to the SA model. This process occurs in multiple cycles or iterations. We repeatedly make additions and deletions to the models in order to more accurately capture the range of behaviors and relevant contextual elements demonstrated in the dataset.

In the refining process, we use an adapted form of model tracing, a practice that has been used to positive effect by designers of intelligent tutoring systems (Koedinger & Anderson, 1997; VanLehn, Freedman, & Jordan, 2000). In effect, we try to determine at each point in the task performance whether the observed human behavior is consistent with predictions made by the model. We do this by assessing whether the model could have *reasonably* (see the discussion section for more on this term) chosen the same action as the human agent did, given the states of both the SA and cognitive models' allowable responses to that state. If the models cannot account for the observed behavior, we attempt to modify them. This is one of the ways in which this method differs significantly from hypothesis-based experimentation: we do not conduct the annotation process in order to test whether the current build of our model is correct or incorrect, but rather with the intention of improving the existing model so that it more accurately and parsimoniously predicts the observed behavior. In a sense then, the model informing the annotation can be viewed as a "rolling hypothesis" which is updated and refined with each iteration over the video.

When making modifications to the model, we create a second "branch" of the model (as in software engineering), and test the new configuration against the previous one. If the new additions or deletions improve the accuracy of the model, they are maintained, otherwise they are rejected and the initial model is retained. We determine whether an

iteration is an improvement upon the previous model by noting the number of times each version fails to predict what the expert did. We consider the model construction process complete when further iterations cease improving the accuracy of the model. Some potential difficulties with this part of the method, such as the risk of over-fitting the model or of having an unbounded number of possible actions within the model, are examined in the discussion section.

5 – Model and Data Release Once we have finished developing a model and have used it to annotate video footage, we release online both the model and the annotated data (video footage) to other interested parties. This is, we think, a crucial component of cognitive modeling at the communal level. It encourages transparency and allows for more rigorous peer evaluation of research claims, and it facilitates collaboration between investigators. It also encourages data and model re-use.

None of the various elements that we have combined are new. Our contribution lies, we hope, in demonstrating the scientific potential of embedding iterative model building in a systematic, explicit methodology for evaluating models of real world expertise.

Example: Video Game Playing

We have used the method described here to construct models of video game playing, professional mediation, chess playing, and professional cooking. We will describe one of these cases, namely a model of playing Gears of War 3 (Activision), a third-person shooter game for the Microsoft XBOX 360.

In order to construct the model, we had several expert players play the game while we recorded the screen. Afterwards, we asked the individuals to discuss their strategies and thought processes while playing, and began to construct the task model. Once we had an idea of what they were paying attention to in the environment, and how they were making decisions, we began constructing the SA model and the cognitive model.

Figure 2, below, represents the process schematically. On the top is the video frame from Figure 1; this depicts what players would see on screen, and was the video data that formed the basis for the annotations. On the bottom left is a representation of the cognitive model. This contains cognitive and behavioral actions, such as "find cover", "engage enemy", or "assess threat" along with the conditions under which they can occur. The SGOMS model also covers high level planning and dealing with unexpected interruptions. The visualization of the cognitive model depicted in Figure 2 is output from software that we have developed in-house for visualizing these models; the software can be downloaded at https://github.com/mattmartin256/SGOMS_GUI. On the bottom right is a representation of the SA model, which lists the important elements of the environment that are attended to. Examples of these elements include the number of enemies on the screen, whether ammunition is running low, and the state of the character's health. The blue arrows between components represent the fact that the construction process is iterative and that each component is used to modify and refine the others.

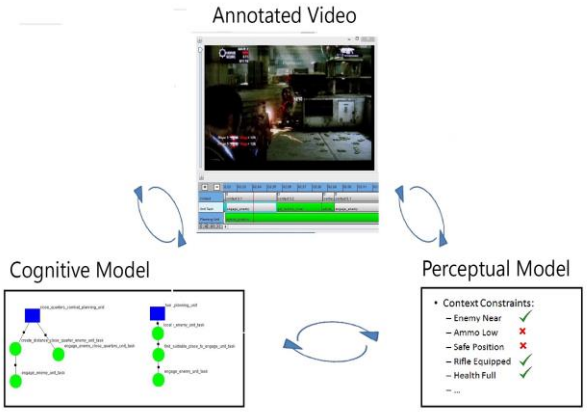


Figure 2: Schematic of Model Construction Process.

Discussion

The methodology presented here is a work in progress. We are attempting to develop a systematic way of modeling expert behavior in complex, real-life scenarios. Such a methodology would be valuable, we argue, both for basic cognition research and to inform the design of socio-technical systems. There are a number of potential difficulties, however, with such a method and we anticipate a number of criticisms here.

The principle difficulty, and the one which this methodology is most explicitly attempting to address, is that there can be an overwhelming amount of complexity in naturalistic environments, and determining what is important or relevant is not straightforward. Laboratory methods are aimed at carving out a tractable section of cognition or behavior, to save the investigators from being forced to address everything all at once: some sub-set of phenomena is selected as important. Determining what is to be included in a cognitive model or experimental design is not atheoretical or pre-theoretical though. Each experimental design or modeling framework necessarily includes certain elements and excludes others. Any chosen methodology thus "smuggles in" theoretical assumptions about what ought to be paid attention to and what we can safely ignore. This is the problem of deciding on one's "unit of analysis", which is a common problem in all scientific disciplines (Hutchins, 2010).

The various approaches for studying expertise define the unit of analysis in many different ways according to the methods and theories being employed. These may include error rates and reaction times (Burkhardt, Détienne, & Wiedenbeck, 1997), memory recall tasks (Vicente & Wang, 1998), eye movement patterns (Reingold, Charness,

Pomplun, & Stampe, 2001), and verbal protocols (Greenwood & King, 1995), among others. The unit of analysis we wish to use is the interaction of an expert (or group thereof) with a complex socio-technical system. We are trying to accommodate this complex unit by bridging the methodologies of experimental psychology approaches that use rich environmental and behavioral descriptions, such as cognitive ecology and anthropology (Bender, Hutchins, & Medin, 2010; D'Andrade, 1995). In essence, we are trying to combine the rigor and predictiveness of process modeling with the richness of ecological studies.

The second difficulty is that individuals are often unable to vocalize what they know (Clark, Yates, Early, & Merriënboer, 2008). We accept that much of the cognitive activity occurring "under the hood" will be invisible and may be unavailable for reporting by the expert. We thus do not assume that the input from our participant experts is the final word on what they are doing mentally. At the same time, however, we believe that this feedback from experts is a desirable component in modeling expertise, and a useful starting point for developing models of expert cognition. Also, it is important to note that the modeling approach used will affect the interactions with the experts. For example, we used SGOMS so our interactions with the experts were naturally geared toward eliciting the information and structures needed to build this type of model.

A third challenge with this method is the degree to which human judgment is required in the construction and evaluation of these models. When deciding whether to add or remove an element from a model, or in judging whether the model has accurately predicted a sequence of behavior, we must rely on the modellers' knowledge and discrimination, and these cannot be perfectly formalized. In other words, the evaluators must decide whether the model could have "reasonably" predicted each decision and action, given the elements in the cognitive and perceptual models. Because we are specifically interested in complex environments in which there is significant behavioral variability between participants and trials, we must use judgment in determining whether two instances of action are equivalent according the model. For example: in the game play scenario presented above, no two instances of the action "take cover" will be exactly the same on the screen, so we must be capable of abstracting from the data to equate the two instances.

Our stance on this issue is informed by Herb Simon's (1969) "ant on the beach metaphor". This states that the observed behavioral complexity and variability of an agent is often the result of the environment in which the agent acts, and does not originate within the agent itself. In the case of the ant, the insect is a rudimentary cognitive-behavioral system. Watching an ant navigate a sandy beach, it may seem that the ant is moving in complex patterns, when, in reality, it may only be obeying the simple heuristic of "do not climb hills". The point that we take from this is that superficially distinct behaviors may reflect the same underlying cognitive processes. It is in determining whether such equivalence exists between instances of behavior that the role of judgment comes into play in this methodology. Here, the public availability of the cognitive and perceptual models along with the annotated videos plays a crucial role. The claim that a judgment was reasonable must stand up to public scrutiny.

Another consideration in the use of this methodology is the difficulty of choosing which elements to include in a given model. We need to negotiate between two extremes: over-fitting and unbounded growth. In the former case, we want models to be capable of accurately predicting behavior by the collection of experts studied, and a model tied too specifically to a single instance or agent will fail to meet this goal. In the latter case, we want to avoid the temptation of endlessly adding elements to the model whenever something unexpected occurs. This is connected to Simon's ant metaphor: we need to determine when superficially dissimilar behaviors represent the same underlying mechanism, because without such abstraction and equating, the models will quickly become bloated and unwieldy. The final goal is to develop models that are informative, predictive, and lean, and this requires a balance between specificity and generality.

One way of evaluating progress on this path arises from our goal to create one unified model that applies across all instances of a field of expertise. Following from our use of branching and the comparison of new models with old, we would expect our current model to be backwards compatible across all the videos used up to that point. If the model has been over-fit it will not show a consistent advantage across all of the videos. With each iteration the demands on the model are actually increased. Eventually, diminishing returns on adjusting the model would signal that it is about as good as it will get. At this point the model and the annotated videos can be used as a benchmark to evaluate alternative models against.

In terms of quantitative evaluation, we are currently using as our metric the percentage of correct predictions the model makes of the expert's next action (this measure was also used in West & Nagy, 2007). However, we are working on other, more detailed ways of quantifying how good the models are. For example, in some cases the model makes a single prediction of what is likely to come come next, but in other cases the model allows for more than one possible next action. Currently, we are experimenting with incorporating the number of predicted next actions at each point into our quantitative measure. For example: if two models are equally predictive in terms of the percentage of actions accurately predicted, but one model regularly predicts a greater number of next possible actions, that model ought to score lower, as it is a less lean (more bloated) model of the expert behavior.

Conclusion

We have presented a methodology for modeling expert behavior and cognition in complex, naturalistic environments. Such a technique will, we hope, be valuable

in furthering our understanding of expertise and situated cognition, and may also be useful in improving the design of socio-technical systems, such as emergency operations centers or aircraft cockpits. We support an open-source approach to scientific research, and hope that explicit attention to methodology, along with the open sharing of tools, data, and models will facilitate collaboration among researchers and the development of more unified, comprehensive cognitive models.

References

Bender, A., Hutchins, E., & Medin, D. (2010). Anthropology in cognitive science. *Topics in Cognitive Science*, *2*, 374–385.

Burkhardt, J. M., Détienne, F., & Wiedenbeck, S. (1997, January). Mental representations constructed by experts and novices in object-oriented program comprehension. In *Human-Computer Interaction INTERACT'97*. Springer US.

Clark, R. E., Feldon, D., van Merriënboer, J. J., Yates, K., & Early, S. (2008). Cognitive task analysis. *Handbook of research on educational communications and technology*, *3*, 577-593.

Cooper, R. P. (2007). The role of falsification in the development of cognitive architectures: insights from a Lakatosian analysis. *Cognitive Science*, *31*, 509–533.

d'Andrade, R. G. (1995). *The development of cognitive anthropology*. Cambridge University Press.

Ericsson, A. K. (2006). Protocol Analysis and Expert Thought: Concurrent Verbalizations of Thinking during Experts Performance on Representative Tasks. In *The Cambridge Handbook of Expertise and Expert Performance*. Cambridge University Press.

Greenwood, J., & King, M. (1995). Some surprising similarities in the clinical reasoning of "expert" and "novice" orthopaedic nurses: report of a study using verbal protocols and protocol analyses. *Journal of Advanced Nursing*, *22*, 907–913.

Hutchins, E. (2010). Cognitive ecology. *Topics in Cognitive Science*, *2*(4), 705–715.

John, B. E., & Kieras, D. E. (1996). The GOMS family of user interface analysis techniques: Comparison and contrast. *ACM Transactions on Computer-Human Interaction (TOCHI)*, *3*(4), 320-351.

Kieras, D. E., & Meyer, D. E. (2000). The role of cognitive task analysis in the application of predictive models of human performance. In *Cognitive Task Analysis*. Psychology Press.

Kipp, M. (2010). Anvil: The video annotation research tool. In *Handbook of Corpus Phonology*. Oxford University Press.

Klein, G., Ross, K. G., Moon, B. M., Klein, D. E., Hoffman, R. R., & Hollnagel, E. (2003). Macrocognition. *Intelligent Systems, IEEE*, *18*(3), 81-85.

Koedinger, K. R., & Anderson, J. R. (1997). Intelligent tutoring goes to school in the big city. *International Journal of Artificial Intelligence in Education*, *8*, 30–43.

Lakatos, I. (1970). Falsification and the methodology of scientific research programmes. In *Criticism and the Growth of Knowledge*, *4*, 91–196

MacDougall, W., West, R., & Hancock, E. (2012). Modeling Multi-Agent Chaos: Killing Aliens and Managing Difficult People. *Proceedings of the Thirty-sixth Annual Conference of the Cognitive Science Society* (pp. 2603-2608). Hillsdale, NJ: Lawrence Erlbaum Associates.

Newell, A. (1973). You can't play 20 questions with nature and win: Projective comments on the papers of this symposium. In *Visual Information Processing*. Academic Press.

Popper, K. R. (1963). Science as falsification. In T. Schtick (Ed.), *Conjectures and refutations*. Mountain View, CA: Routledge and Keagan Paul.

Reingold, E. M., Charness, N., Pomplun, M., & Stampe, D. M. (2001). Visual span in expert chess players: evidence from eye movements. *Psychological Science: A Journal of the American Psychological Society*, *12*, 48–55.

Schraagen, J. M., Ormerod, T., Militello, L., & Lipshitz, R. (Eds.). (2012). *Naturalistic decision making and macrocognition*. Ashgate Publishing Ltd.

Shanteau, J., & Hall, B. (2001). What does it mean when experts disagree. In E. Salas, & G. Klein (Eds.), *Linking expertise and naturalistic decision making*. Mahwah, N. J.: Lawrence Erlbaum Associates.

Simon, H. A. (1996). *The sciences of the artificial* (Vol. 136). MIT press.

VanLehn, K., Freedman, R., Jordan, P., Murray, C., Rosé, C. P., Schulze, K., Shelby, R., Treacy, D., Weinstein, A., & Wintersgill, M. (2000). Fading and deepening: The next steps for Andes and other model-tracing tutors. In Gauthier, Frasson, VanLehn (Eds.), *Intelligent Tutoring Systems: 5th International Conference: Vol. 1839. Lecture Notes in Computer Science*. Springer-Verlag Berlin & Heidelberg GmbH & Co. K.

Vicente, K. J., & Wang, J. H. (1998). An ecological theory of expertise effects in memory recall. *Psychological Review*, *105*, 33–57.

West, R. L., & Nagy, G. (2007). Using GOMS for Modeling Routine Tasks Within Complex Sociotechnical Systems: Connecting Macrocognitive Models to Microcognition. *Journal of Cognitive Engineering and Decision Making*, *1*(2), 186–211.

Williams, R. (2006). Using cognitive ethnography to study instruction. *Proceedings of the 7th International Conference on Learning Sciences*, 838–844. International Society of the Learning Sciences

Mathematical Formalization and Optimization of an ACT-R Instance-Based Learning Model

Nadia Said[1, 2] (nadia.said@psychologie.uni-heidelberg.de)
Michael Engelhart[2] (michael.engelhart@iwr.uni-heidelberg.de)
Christian Kirches[2] (christian.kirches@iwr.uni-heidelberg.de)
Stefan Körkel[2] (stefan.koerkel@iwr.uni-heidelberg.de)
Daniel V. Holt[1] (daniel.holt@psychologie.uni-heidelberg.de)

[1] Heidelberg University, Department of Psychology, Hauptstr. 47–51, 69117 Heidelberg, Germany
[2] Interdisciplinary Center for Scientific Computing (IWR), Im Neuenheimer Feld 368, 69120 Heidelberg, Germany

Abstract

The performance of cognitive models often depends on the settings of specific model parameters, such as the rate of memory decay or the speed of motor responses. The systematic exploration of a model's parameter space can yield relevant insights into model behavior and can also be used to improve the fit of a model to human data. However, exhaustive parameter space searches quickly run into a combinatorial explosion as the number of parameters investigated increases. Taking an established instance-based learning task as example, we show how simulation using parallel computing and derivative-free optimization methods can be applied to investigate the effects of different parameter settings. We find that both global optimization methods involving genetic algorithms as well as local methods yield satisfactory results in this case. Furthermore, we show how a model implemented in a specific cognitive architecture (ACT-R) can be mathematically reformulated to prepare the application of derivative-based optimization methods which promise further efficiency gains for quantitative analysis.

Keywords: cognitive modeling; ACT-R; instance-based learning; discrete-time systems; derivative-free optimization; parameter identification

Introduction

Most formal models of cognitive processes contain adjustable parameters which moderate model behavior. Exploring the effects of different parameter settings in a cognitive model is important to fully understand its behavior, to identify parameter combinations providing the best fit to human data, and to analyze sensitivity towards parameter variations (see Roberts & Pashler, 2000). In practice, this exploration is still often conducted manually, guided by researcher intuition or sometimes just by trial-and-error. The systematic exploration of a given parameter space is often desirable, but quickly runs into difficulties, as processing time increases exponentially with the number of parameters and the resolution of analysis (the *curse of dimensionality*). Compounding the problem, the computational performance of cognitive models is often comparatively poor as cognitive plausibility usually has priority over computational performance. The development of more efficient methods for parameter space exploration has therefore become an emergent topic in cognitive modeling research (e.g., Best et al., 2009; Gluck, Scheutz, Gunzelmann, Harris, & Kershner, 2007; Lane & Gobet, 2013; Moore, 2011). Complementing existing approaches, we will illustrate how optimization-methods from the field of scientific computing can be applied to parameter search problems. As example we use a model of an implicit learning task, originally implemented in the ACT-R cognitive architecture (Taatgen & Wallach, 2002).

While parallel high-performance computing can improve the speed of parameter space searches, the combinatorial explosion inherent in this task easily exceeds the capacity even of large computing resources (Gluck et al., 2007). Another possibility is to optimize the efficiency of search algorithms. In the current literature, two approaches of this kind can be found. One is to sample the search space selectively, e.g., by Adaptive Mesh Refinement or Regression Trees (Best et al., 2009; Moore, 2011). Areas of the search space with high-information content (e.g., containing discontinuities or non-linear gradients) are sampled more densely, areas with low-information content only sparsely. This strategy allows to preserve most information relevant for modeling purposes while reducing the amount of sampling required. However, instead of approximating the full parameter space, it its sometimes sufficient to find particular points or areas with certain characteristics, e.g., parameter combinations providing the best model fit to empirical data. To reach this goal, derivative-free optimization methods such as genetic algorithms have been employed, which use an evolutionary generate-and-select-strategy to find optimal parameter combinations (e.g., Kase, Ritter, & Schoelles, 2008; Lane & Gobet, 2013). Here, we will illustrate how a cognitive model implemented in a specific cognitive architecture (ACT-R) can be mathematically reformulated towards applying a third general approach, derivative-based optimization, which promises further efficiency gains and additional analytic insights compared to derivative-free optimization.

The Sugar Factory Paradigm

The task used to illustrate this approach is the *Sugar Factory*, a computer-simulated scenario developed by Berry and Broadbent (1984) in order to study how people interact with dynamic systems. In behavorial studies participants are often able to control the *Sugar Factory* system above chance level yet can not verbalize how the system works (Berry & Broadbent, 1984, 1988). This can be explained by assuming that

participants learn to associate particular states of the simulation with specific actions (*instance-based learning*) instead of inducing generalized rules about system behavior (Dienes & Fahey, 1995; Taatgen & Wallach, 2002). The *Sugar Factory* has repeatedly been used in experimental research and the underlying principles of instance-based learning have been shown to apply to more complex situations, ranging from playing backgammon (Sanner, Anderson, Lebiere, & Lovett, 2000) to economic decision making (Gonzalez & Lebiere, 2005) or air-traffic control (Lebiere, Anderson, & Bothell, 2001).

In the *Sugar Factory*, participants are asked to reach a specific sugar production p^* by choosing the number of workers x in each round $j \in [1,N]$. The equation below describes the behavior of the *Sugar Factory*:

$$p_{j+1} = 2 \cdot x_j - p_j + u_r, \tag{1a}$$

$$u_r \in \{-1,0,1\}, \tag{1b}$$

$$p \in \{1,...,12\}, \tag{1c}$$

$$x \in \{1,...,12\}, \tag{1d}$$

where u_r is a random component. If the resulting production is less than 1 or greater than 12, then p is set to 1 or 12 respectively. The goal is to produce 9000 tons of sugar which corresponds to $p = 9$ on each of a number of trials. The initial production value is $p_1 = 6$ and the task is run for 40 rounds. Participants are not informed about the system structure. A sugar output of $p \in [8,10]$ is scored as being *on target*, making it possible to be on target 100 % of the time, despite the random component.

ACT-R Model of the Sugar Factory

Instanced-based learning can be modeled using the declarative memory module of the ACT-R architecture (Anderson et al., 2004). We used an adapted version of the Taatgen and Wallach (2002) model of the *Sugar Factory*, in which instances are represented as memory chunks encoding task state, participant action, and outcome (i.e., current production p_j, number of workers x_j, and new production p_{j+1}).

Each chunk i has an activation value A_i which is computed from three components: the base-level activation B_i, a context component C_i and a noise component $u_{n,ij}$,

$$A_i := B_i + C_i + u_{n,ij}. \tag{2}$$

To retrieve a chunk from memory a retrieval request is made to the declarative module. Only chunks with an activation level above threshold τ are eligible for retrieval.

The base-level activation B_i is calculated from the number n_i of presentations of a chunk i, its the time since its creation L_i and the decay parameter d[1],

$$B_i := \ln\left(\frac{n_i}{1-d}\right) - d \cdot \ln(L_i). \tag{3}$$

[1]Note that we used the ACT-R optimized learning equation.

In this model the context component C_i is determined by the similarity between the numerical values for workers and productions in the retrieval request and corresponding values of the chunks in declarative memory:

$$C_i := P \cdot \sum_k M_{ik}, \tag{4}$$

with M_{ik} as similarity values [2]. The parameter P reflects the amount of weighting given to the similarities.

As the *Sugar Factory* model is very simple, only few production rules are necessary. The steps our model runs through are described below.

1. Start with a number of workers $x = \{7,8,9\}$.
2. Request to retrieve a chunk from memory which matches the current task state and results in a production of $p = 9$.
3. (a) If there is such a chunk and the activation of this chunk is above the threshold τ: perform the action stored in the retrieved chunk, i.e., change the workforce.
   (b) If no chunk reaches the activation threshold τ, perform a random action. If the sugar production is below or above target, then $\{-2,...,2\}$ is added to the current workforce. If the sugar production is on target, then $\{-1,0,1\}$ is added to the current workforce.
4. Create or update a chunk with the information from this round.

The model actions described in rule 1 and rule 3 (b) are based on empirical observations reported in Dienes and Fahey (1995).

Mathematical Reformulation

We start by rewriting the logical relations in the ACT-R model as a recurrence relations system, which is then reformulated using only numerical formulas.

Let N_r be the number of rounds. The general outline of the cognitive process is as follows.

In every round

1. compute the activations of chunks,
2. select the chunk with the highest activation,
3. retrieve information stored in this chunk,
4. perform action,
5. update memory.

[2]The formula for the similarities of numbers was taken from the ACT-R 6.0 Tutorial (2012):

$$M_{ik} := -\frac{|a-b|}{\max(a,b)},$$

For the *Sugar Factory* each chunk represents an instance with i three slots: current production p_j, action (i.e., number of workers x_j), and the new production p_{j+1}. The maximum total number N_c of chunks present in the model can be calculated from the number of values possible for its slots c_{ik}, $k \in \{1,2,3\}$, slot k of chunk i. Feasible values for new workforce (c_{i1}), the current production (c_{i2}) and the new production (c_{i3}) are $\{1,\ldots,12\}$ each. Thus,

$$N_c = 12 \cdot 12 \cdot 12 = 1728. \tag{6}$$

We allocate every possible chunk and set its initial activity to $-M$ where M is sufficiently large.

The lifetime of a chunk consists of the round of its generation t_i, the current round j and a time constant $T = 0.05\,\mathrm{s}$.

The model contains different types of variables:

- *states* including the activation of the chunks and process specific states as the current number of workers and the current production,
- *parameters* τ, d, P and s describing the cognitive properties of the individual participant (the default values in this model are: $\tau = 0$, $d = 0.5$, $P = 10$ and $s = 0.2$)
- *inputs*, containing the cognitive noise u_n, random decisions by the participants $u_w + u_{\text{on}}$ resp. $u_w + u_{\text{off}}$ and system inputs u_r. The sequences of random values are generated a priori as reproducible pseudo-random numbers.

Feasible values for the inputs are: u_{on}, $\in \{-1,0,1\}^{N_r}$ if the production is on target and u_{off}, $\in \{7,8,9\} \times \{-2,...,2\}^{(N_r-1)}$ if the production is off target as well as u_r, $\in \{-1,0,1\}^{N_r}$ for the random vector. All inputs are vectors of length N_r, except $u_{n,ij}$, which is of length $N_r \cdot N_c$. If the target has been reached in round j, i.e. the new production p_{j+1} equals 8, 9, or 10, an indicator R_{j+1} is set to 1. The overall score is computed by summation of R_{j+1}.

This modeling approach leads to a nonlinear recurrence relation system, see Algorithm 1.

In our approach, the logical phrases from the ACT-R formalism are modeled by arg max, $|.|$, max and *if-then* statements. We formulate them using the Heaviside and Delta functions and write the *if-then* statements

$$x = \begin{cases} a, & \text{if} \quad s \geq 0 \\ b, & \text{if} \quad s < 0 \end{cases}, \quad x = \begin{cases} c, & \text{if} \quad t = 0 \\ d, & \text{if} \quad t \neq 0 \end{cases}$$

as

$$x = H(s) \cdot a + (1 - H(s)) \cdot b, \quad x = \delta(t) \cdot c + (1 - \delta(t)) \cdot d.$$

So we substitute $\max(x,y)$ and $\frac{|x-y|}{\max(x,y)}$. To calculate

$$i^* = \arg\max A_i, \quad x_j = \begin{cases} c_{i^*1}, & A_{i^*} \geq \tau \\ u_{w,j}, & A_{i^*} < \tau \end{cases},$$

for $j = 1,\ldots,N_r$ **do**

(1) **for** $i = 1,\ldots,N_c$ **do**

$L_i := (j - t_i) + T;$
$B_i := \ln(n_i/(1-d)) - d \cdot \ln(L_i);$
$M_{i1} := -|p_j - c_{i2}|/\max(p_j, c_{i2});$
$M_{i2} := -|9 - c_{i3}|/\max(9, c_{i3});$
$A_i := B_i + P \cdot (M_{i1} + M_{i2}) + u_{n,ij};$

end

(2) $i^* := \text{argmax}_i A_i;$

(3) $A_{i^*} \geq \tau$?
(i) $A_{i^*} \geq \tau$: $x_j := c_{i^*1}$;
(ii) $A_{i^*} < \tau$: $x_j := u_{w,j}$;

(4) $p_{j+1} := 2 \cdot x_j - p_j + u_{r,j}$;
(i) $p_{j+1} > 12$: $p_{j+1} = 12$;
(ii) $p_{j+1} < 1$: $p_{j+1} = 1$;
(iii) $p_{j+1} = 9$?
(a) $p_{j+1} = 9$: $u_{w,j+1} := u_{w,j} + u_{\text{on},j}$;
(b) $p_{j+1} \neq 9$: $u_{w,j+1} := u_{w,j} + u_{\text{off},j}$;

(5) $u_{w,j+1} > 12$: $u_{w,j+1} = 12$;

(6) $u_{w,j+1} < 1$: $u_{w,j+1} = 1$;

(7) $p_{j+1} \in \{8,...,10\}$?
(i) $p_{j+1} \in \{8,...,10\}$: $R_{j+1} := 1$;
(ii) $p_{j+1} \notin \{8,...,10\}$: $R_{j+1} := 0$;

(8) $\exists i = 1,\ldots,N_c : c_i = (x_j, p_j, p_{j+1})$?
(i) $\exists i$: $n_i := n_i + 1$
(ii) $\nexists i$: $N_c := N_c + 1$;
$c_{N_c} := (x_j, p_j, p_{j+1})$;
$n_{N_c} := 1$;
$t_{N_c} := j$;

end

Algorithm 1: ACT-R algorithm of *Sugar Factory*

we firstly compute $A^* = \max_i A_i$ and then

$$x_j = \sum_{i=1}^{N_c} H(A_i - A^*) \cdot (H(A^* - \tau) \cdot c_{i^*1} + (1 - (A^* - \tau)) \cdot u_{w,j}).$$

In order to compute sensitivities and use derivative-based optimization algorithms at a later stage, we aim for a continuous model formulation. We replace Heaviside and Delta functions by continuous approximate redefinitions

$$H(x) := \frac{1}{\pi} \arctan(h \cdot x) + \frac{1}{2}, \quad \delta(x) := \exp\left(-\frac{x^2}{a^2}\right),$$

with $h = 1000$, $a = 0.01$.

The limits on the production in the *Sugar Factory* are implemented by *if-then* statements. Those rules appear as follows in our mathematical description:

$$p_{j+1} > 12 : p_{j+1} = 12,$$
$$p_{j+1} < 1 : p_{j+1} = 1.$$

In our reformulation those *if-then* statements are smoothened

using the Heaviside function $H()$:

$$\begin{aligned}\tilde{p}_{j+1} &= 2 \cdot x_{j+1} - p_j + u_{rj} \\ p_{j+1} &= H(\tilde{p}_{j+1} - 12) \cdot 12 + \big(1 - H(\tilde{p}_{j+1} - 12)\big) \\ &\quad \cdot \Big(H(1 - \tilde{p}_{j+1}) \cdot 1 + \big(1 - H(1 - \tilde{p}_{j+1})\big) \cdot \tilde{p}_{j+1}\Big).\end{aligned}$$

Simulation

We implemented the reformulation of the model in the *Python* programming language. Simulations were run on a 48-core high-performance server ($4 \cdot 12$-core *AMD Opteron 6176*, non-dedicated) with 256 GB RAM. The maximum runtime per simulation run did not exceed one day. We focused on parameters P and τ, as they have considerable effect on model performance yet no strong empirically based recommendations for default values exist. All other parameters where left at the recommended default values.

For each grid point the simulation was run 100 times with different input vectors, see Figure 1. The activation noise $u_{n,ij}$ was set to zero, as it does not lead to a notable change in mean performance. In a second step, only instances in which a chunk was retrieved were counted as being on target and compared to our previous results (see Figure 1 (b)). Figure 2 shows the sensitivity of results with regard to different initial production values.

In general, all simulation results show a similar pattern in response to parameter variation. Figure 1 (a) shows a characteristic interaction of parameters τ and P, with the *highest performing* parameter combinations located in a wedge-shaped area at the center of the plot and lower performance in both lower left and upper right corners. Considering whether model responses where based on memory retrieval as opposed to random behavior (see 1 (b)) shows that learning occurs in the lower left corner. In contrast, in the upper right corner behavior is almost exclusively driven by random behavior.

Furthermore, Figure 2 shows that the initial starting values of the *Sugar Factory* problem have a notable influence on overall performance, but do not interact in an unpredictable way with parameter settings. The pattern is cognitively plausible, as starting values closer to the actual best control values make system control easier.

Optimization

In order to determine the parameter combination with the highest performance and the best model fit, we applied a number of derivative-free optimization algorithms: Nelder-Mead Simplex (Nelder & Mead, 1965) with explicit support for bound constraints (Box, 1965) and BOBYQA (Powell, 2009) which are both local derivative-free optimization solvers. BOBYQA uses an iteratively constructed quadratic approximation for the objective function. Additionally, we applied ESCH, a modified genetic algorithm (Beyer & Schwefel, 2002) and Controlled Random Search (CRS) with local mutation (Kaelo & Ali, 2006). ESCH and CRS are both heuristic global solvers. CRS starts with a population of random points, and evolves these heuristically, a method comparable to genetic algorithms. All optimization algorithms were applied using the Python interface of NLopt (Johnson, 2014). The stopping criterion for the local solvers was a relative tolerance on the optimization parameters of 0.1. For the heuristic global solvers the stopping criterion was set to a maximum runtime of $960s$.

The objective function for the highest performance was a weighted sum consisting of the mean of the results on target and the standard deviation,

$$a \cdot \frac{1}{n}\sum_{i=1}^{n} R^i + b \cdot \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left(R^i - \frac{1}{n}\sum_{i=1}^{n} R^i\right)^2},$$

where $R^i = \sum_j R^i_{j+1}$. R^i_{j+1} is the indicator *on target* in round $j = 1, \ldots, N_r$ for input $i = 1, \ldots, n$, $n = 100$.

Table 1 shows the results for $a = 1$ and $b = 0$ and 100 inputs using multiple start values (see Figure 1 (a)). The local solvers Nelder-Mead and BOBYQA both found the same local maximum ($\tau = -4$, $P = 27$ with objective = 20.15), for ESCH and CRS we successively increased the time limit from 120 to 960 seconds. Table 1 shows the maxima found by the heuristic global solvers after 960 seconds. For $a = 1$ and $b = -1$, all solvers found the same point as a maximum ($\tau \approx -6.5$, $P \approx 30$ with objective ≈ 13.87), except CRS which found a slightly better point ($\tau \approx -8.15$, $P \approx 34.9$ with objective ≈ 14.04).

For optimizing the model fit, the objective function was the root mean square deviation (RMSD) of the model performance and a human reference value $R_{\text{ref}} = 7.9$ taken from the literature (Dienes & Fahey, 1995),

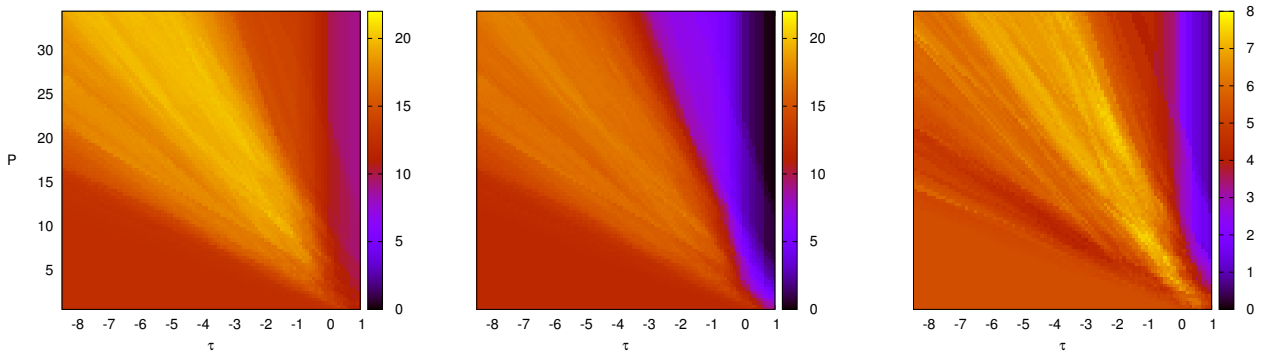
$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(R^i - R_{\text{ref}}\right)^2}.$$

Table 2 shows the results for the different solvers again using multiple start values. All solvers found the same point as a maximum ($\tau \approx 0.5$, $P \approx 30$ with objective $= 4.05$). The time limit for ESCH and CRS was set to 5 seconds.

Table 1: Solver comparison, highest performance

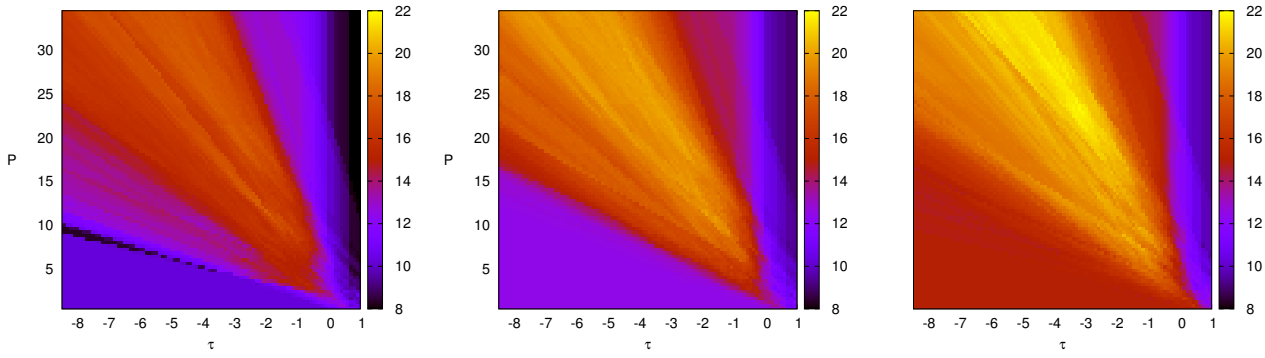
Solver	τ	P	Maximum	#Evaluations
Nelder-Mead	-4	27	20.15	36
BOBYQA	-4	27	20.15	43
ESCH	-3.13	22.36	20.13	863
CRS	-4.21	28.52	20.2	860

The parameter combinations that provide the best fit to average human performance are located at about $\tau = 0.5$ and $P \approx 30$, as indicated by Figure 1 (a) and the results of optimization routines. Interestingly, this combination is far removed from the possible optimal performance, located at parameter values $\tau = -4.21$ and $P = 28.52$. Apparently, consid-



(a) Mean number of rounds on target (b) Mean number, only activated chunks (c) SD, only activated chunks

Figure 1: Rounds on target for 100 inputs over 40 rounds, initial production $p_0 = 6$, medium grid (8256 grid points)



(a) Initial production $p_0 = 1$ (b) Initial production $p_0 = 6$ (standard) (c) Initial production $p_0 = 9$

Figure 2: Rounds on target for 100 inputs over 40 rounds with different initial production values, medium grid (8256 grid points)

Table 2: Solver comparison, best model fit

Solver	τ	P	Maximum	#Evaluations
Nelder-Mead	0.5	28.13	4.05	67
BOBYQA	0.5	27.80	4.05	54
ESCH	0.45	27.88	4.05	6374
CRS	0.48	32.94	4.05	4500

ering even weak memories of previous instances (i.e., a low retrieval threshold τ) is an advantage in this task.

Conclusions

In this work we derive a mathematical formalization of an ACT-R cognitive model to enhance its numerical tractability. The formulation contains both specific parts describing the instances and rules of a particular task and generic parts modeling the declarative memory module of the ACT-R framework. This enables us to apply the approach to a broader range of scenarios. Our formulation transfers the logic-based ACT-R rules to a recurrence relation, which after smoothing represents a differentiable mapping from model parameters and model inputs to the model outputs. In this work we used this formulation for a simulation-based study of the *Sugar Factory* paradigm where we explored the parameter space by parallel computing and computed optima by search-methods and derivative-free approaches. Our next step will be the evaluation of the derivatives of the recurrence relation by using techniques of algorithmic differentiation (Griewank & Walther, 2008) and to apply them in derivative based numerical approaches for sensitivity analysis, parameter estimation and optimum experimental design for efficient model calibration (Körkel, Kostina, Bock, & Schlöder, 2004). Using these methods promises a considerable reduction of computational costs for the quantitative analysis of suitable cognitive process models.

Acknowledgments

This project was supported by the Excellence Initiative of the German Research Council (DFG) at Heidelberg University and the Heidelberg Graduate School of Mathematical and Computational Methods for the Sciences at the Interdisciplinary Center for Scientific Computing (IWR).

References

ACT-R 6.0 Tutorial. (2012). Retrieved from http://act-r.psy.cmu.edu/software/

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological Review*, *111*, 1036–1060.

Berry, D. C., & Broadbent, D. E. (1984). On the relationship between task performance and associated verbalizable knowledge. *The Quarterly Journal of Experimental Psychology*, *36*, 209–231.

Berry, D. C., & Broadbent, D. E. (1988). Interactive tasks and the implicit-explicit distinction. *British Journal of Psychology*, *79*, 251–272.

Best, B. J., Furjanic, C., Gerhart, N., Fincham, J., Gluck, K. A., Gunzelmann, G., & Krusmark, M. A. (2009). Adaptive mesh refinement for efficient exploration of cognitive architectures and cognitive models. In *Proceedings of the ninth international conference on cognitive modeling* (paper 252). Manchester, United Kingdom: University of Manchester.

Beyer, H.-G., & Schwefel, H.-P. (2002). Evolution strategies – a comprehensive introduction. *Natural Computing*, *1*, 3–52.

Box, M. J. (1965). A new method of constrained optimization and a comparison with other methods. *The Computer Journal*, *8*, 42–52.

Dienes, Z., & Fahey, R. (1995). Role of specific instances in controlling a dynamic system. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *21*, 848–862.

Gluck, K., Scheutz, M., Gunzelmann, G., Harris, J., & Kershner, J. (2007). Combinatorics meets processing power: Large-scale computational resources for BRIMS. In *Proceedings of the sixteenth conference on behavior representation in modeling and simulation* (pp. 73–83). Orlando, FL: Simulation Interoperability Standards Organization.

Gonzalez, C., & Lebiere, C. (2005). Instance-based cognitive models of decision-making. In D. J. Zizzo & A. Courakis (Eds.), *Transfer of knowledge in economic decision making.* New York: Palgrave McMillan.

Johnson, S. G. (2014). The NLopt nonlinear-optimization package. Retrieved from http://ab-initio.mit.edu/nlop

Kaelo, P., & Ali, M. M. (2006). Some variants of the controlled random search algorithm for global optimization. *Journal of Optimization Theory and Applications*, *130*, 253–264.

Kase, S. E., Ritter, F. E., & Schoelles, M. (2008). From modeler-free individual data fitting to 3-d parametric prediction landscapes: A research expedition. In *Proceedings of the 30th annual conference of the cognitive science society* (pp. 1398–1403). Austin, TX: Cognitive Science Society.

Lane, P. C. R., & Gobet, F. (2013). Evolving non-dominated parameter sets. *Journal of Artificial General Intelligence*, *4*, 358–367.

Lebiere, C., Anderson, J. R., & Bothell, D. (2001). Multitasking and cognitive workload in an ACT-R model of a simplified air traffic control task. In *Proceedings of the tenth conference on computer generated forces and behavior representation.* Norfolk, VA.

Moore, L. R. J. (2011). Cognitive model exploration and optimization: a new challenge for computational science. *Computational and Mathematical Organization Theory*, *17*, 296–313.

Nelder, J. A., & Mead, R. (1965). A simplex method for function minimization. *The Computer Journal*, *7*, 308–313.

Powell, M. J. D. (2009). The BOBYQA algorithm for bound constrained optimization without derivatives. *Cambridge Report DAMTP NA2009/06, University of Cambridge, United Kingdom*.

Roberts, S., & Pashler, H. (2000). How persuasive is a good fit? A comment on theory testing. *Psychological Review*, *107*, 358–367.

Sanner, S., Anderson, J. R., Lebiere, C., & Lovett, M. C. (2000). Achieving efficient and cognitively plausible learning in backgammon. In *Proceedings of the seventeenth international conference on machine learning* (pp. 823–830). Stanford, CA: Morgan Kaufmann.

Taatgen, N. A., & Wallach, D. (2002). Whether skill acquisition is rule or instance based is determined by the structure of the task. *Cognitive Science Quarterly*, *2*, 163–204.

A specification-aware modeling of mental model theory for syllogistic reasoning

Yutaro Sugimoto[1] **and Yuri Sato**[2]
[1]Department of Philosophy, Keio University [2]Interfaculty Initiative in Information Studies, The University of Tokyo
sugimoto@abelard.flet.keio.ac.jp, sato@iii.u-tokyo.ac.jp,

Abstract

Computational cognitive models can embody the structures and processes proposed in a cognitive theory. However, they do not necessarily reveal the theory's underlying assumptions and specifications. This study aims to bridge the gap between cognitive theory and its computational implementation, focusing on a case of mental model theory on human reasoning. Using a mathematics-based and statically-typed programming language (Haskell), we provide a specification-aware computational implementation of syllogistic reasoning with mental models.

keywords: Mental model theory, Type system, Specification, Cognitive modeling, Logical reasoning.

Introduction

The specification problem in cognitive modeling

In cognitive science research, mental representations and processes underlying human cognition have been treated in analogy with computer programs consisting of data structures and algorithms.However, it is not always easy to understand the relationship between the two in cognitive theories employing natural language for description. This occasionally leads to ambiguous formulations of cognitive theories. To remove these ambiguities, researchers have developed computer implementations of these theories.This type of modeling has revealed more detailed structures and processes than the existing formulations using natural language.

Recently, some researchers have questioned whether or not cognitive models should actually reveal theory *specifications*. McClelland (2009) pointed out that computational cognitive modelings are not full accounts of cognitive theories; implementations are not intended to embody the aspects of a theory's assumptions or specifications that are loosely defined in a cognitive theory. However, Cooper and Guest (2014) argue that *McClelland (2009) downplays the role of modeling in theory specification, and in particular the possibility that a model might embody a set of assumptions (p. 43).* To remedy this, they suggest that it is necessary to improve the current situation whereby modeling approaches are not suitable for theory specification, by bridging the gap between computational implementation and cognitive theory. The present study was conceived in a similar vein. We introduce a modeling method consisting of full-fledged descriptions of theory specifications: "type system based modeling."

A case of mental model reasoning

Cognitive studies involving the mental model theory, which was introduced by Johnson-Laird (1983; 1984), are appropriate examples with which to discuss the specification problem in cognitive modeling. The mental model theory is a cognitive theory of sentence interpretation and inference. The theory was first formulated for categorical syllogisms, but has now been extended to various domains of reasoning (for a review, see Khemlani & Johnson-Laird, 2013). Syllogism with quantificational sentences is one of the most elementary forms of natural language inference, and is important in the analysis of human reasoning (cf. Moss, 2008). The present study considers the domain of syllogisms only, focussing on the recent version (Bucciarelli & Johnson-Laird, 1999) and its corresponding computer implementation (Mental Models & Reasoning Lab, abbreviated as MMRLab.[1]).

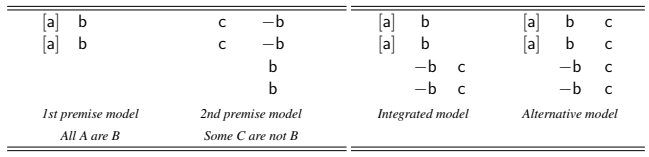


Fig 1: Syllogistic reasoning with mental models

Natural language description Syllogistic reasoning processes in mental model theory as described with natural language are briefly illustrated below. The basic idea underlying the theory is that people interpret sentences by constructing mental models corresponding to possibilities and make inferences by constructing counter-models. (1) Mental models consist of a finite number of tokens, denoting the properties of individuals. *All A are B* has a model illustrated on the leftmost side of Fig.1, where each row represents an individual. A row consisting of two tokens, a and b, refers to an individual that is *A* and *B*. The tokens with square brackets, [a], express that the set containing them is exhaustively represented by these tokens and that no new tokens can be added to it. A sequence of tokens without square brackets can be extended with new tokens so that an alternative model is constructed. (2) *Some C are not B* has a model illustrated on the second from the left of Fig.1. A row with a single token, b, refers to an individual that is *B* but not *C*. A row consisting of two tokens, c and –b, refers to an individual that is *C* but not *B*, by using "–" to denote negation. (3) These two models for premises are integrated into a single model, as shown in the second from the right in Fig.1. Two tentative conclusions, *Some A are not C* and *Some C are not A*, are extracted. (4) To search counterexample for them, an alternative model is constructed by adding new tokens (token c), as shown in the rightmost side of Fig.1. Since each token of *A* is corresponding to tokens of *C*, *Some*

[1]As described on p.14 of Khemlani and Johnson-Laird (2013), the major part of the program code for syllogistic fragments (MMRLab) still lives on in another implementation of the mental model theory, what is called *mReasoner*, which can cope with various domains beyond categorical syllogisms.

A are not C is refuted. Instead, *Some C are not A* survives.

Modeling implementations Following the theory informally described above, Johnson-Laird and his colleagues have developed several computational cognitive models, some of which have been published (p. 443 of Khemlani & Johnson-Laird, 2012). Their implementation of syllogisms is found in MMRLab. It is written in Common Lisp and a mental model is represented as a *list*, which is the most basic data structure in Lisp. For example, the 1st and 2nd premises models in Fig. 1 are represented as lists:

```
(((* A) (B)) ((* A) (B)))

(((C) (- B)) ((C) (- B)) ((B)) ((B)))
```

Their implementation of mental model theory was successful in embodying structures and processes proposed in the theory but it was unable to fully account for its specification. More concretely, it lacked a mathematical formalization of the concept of a "mental model." This ambiguity has resulted in various misunderstandings of the theory, and most of the controversy around the theory in cognitive science and logic has focused on this point (Braine, 1994; Hintikka, 1987; Stenning & Van Lambalgen, 2008). It is therefore important to understand the nature of the theory by addressing the specification problem.

Beyond the specification problem

Formal modeling A possible method to bridge the gap between implementation and theory is to convert natural language description theories to formal systems. This method is called *formal cognitive modeling*: structures and processes described in a theory are decomposed at a more abstract (mathematical) level than existing implementations (e.g., Arkoudas & Bringsjord, 2008; Bosse, Jonker, & Treur, 2006). Koralus and Mascarenhas's (2013) modeling is an example of this approach. They constructed a formal (propositional) inference system, called "the erotetic theory of reasoning", which essentially corresponds to mental model reasoning. (i) They started by converting from mental models to algebraic structures such as $p \sqcup q$, consisting of representational units (p and q standing for propositions) and operations ($\sqcup$ standing for conjunction), which can be translated to logical formulas such as $p \wedge q$. Here non-standard semantics, not model-theoretic semantics, were given for the interpretations of sentences. (ii) Update (erotetic) rules for adding the new premise, which is treated as an answer to the question in discourse, and operation rules for making inference as refutation were given. (iii) In this system, soundness and completeness for classical propositional semantics were shown via translation.

To our knowledge, Koralus & Mascarenhas's work is the first major formalization of mental model reasoning.[2] This approach has at least an advantage in analyzing computational complexities of solving tasks rather than predicting human performance data of them (for more details see e.g. Verbrugge, 2009; Isaac, Szymanik, & Verbrugge, 2014). However, this modeling is abstract in that it can be implementation-independent. Therefore it may not address problems caused by a particular representation in a cognitive theory. Thus, certain specifications, such as the mental models definition may not be provided at an appropriate level.

Specification-aware modeling An alternative way to bridge the gap between implementation and theory is to use a programming language suitable for specification descriptions, which we call *specification-aware cognitive modeling*. This approach was manifested in the seminal study of Cooper et al. (1996). According to them, (i) ordinary (Lisp and C) implementations themselves do not include the specifications of cognitive theories; (ii) formal theories of specifications, including mathematical descriptions, are not appropriate for bridging the gap between implementation and theory since they are implementation-independent. In our view, Johnson-Laird's mental model implementation in Common Lisp corresponds to case (i) and the formal modeling approaches of Koralus and Mascarenhas (2013) correspond to case (ii). Alternatively, Cooper et al. (1996) have provided some cognitive modeling implementations written by an executable specification language *Sceptic*, consisting of the declarative (logic programming) language Prolog, with the additional device of control structures. Their implementations include the mental model theory for syllogistic reasoning (for more details, see Cooper, 1992)[3]. Here, rewrite rules obey certain mathematical manipulations, such as rewriting from one term to another, as is the case in lambda calculus. This rewriting can be regarded as a specification in itself. Furthermore, an advantage of declarative language is that one can check the logical relationships between input and output representations without specifying a strategy of computation. (cf. sec.3 of Cooper & Guest, 2014).

This study put forward the line of specification-aware modeling. We try to give a "computational" (i.e. executable) semantics for the mental model theory. For the purpose of this research, we consider a reconstruction based on type system (cf. Mitchell, 2003). We refer to this approach as *type system based modeling*. Importantly, we chose a purely-functional programming language *Haskell* that has strong static typing and lazy evaluation strategy (Jones, 2003; Marlow, 2010). In contrast to the Common Lisp language used in Johnson-Laird and his colleagues' implementation (MMRLab), Haskell is a purely functional programming language with strong static typing. Explicit type annotations are useful for defining mental models within the system, and Haskell encourages to pro-

[2] Recently, Clark (submitted) straightforwardly provided a specification of mental model theory using total and partial truth functions, which correspond to exhaustive and non-exhaustive models.

[3] Based on Sceptic, the COGENT environment for cognitive modeling has subsequently been developed. An application to mental model theory is found in chap.5 of Cooper (2002). It is intended to provide theory specifications by reconstructing information-processing models such as box-and-arrow diagrams.

gram in a declarative style. This property can contribute to reasoning about the program itself (e.g. to give a denotational semantics for it Haftmann, 2010; Vytiniotis et al., 2013). It is important not only to guarantee the fact that a program is executable (there is no syntactical error), but also to verify whether a program works as intended by theorists (the aspect of semantics.) Note that this brief paper focuses on a part of the long-term study. Just a relatively weak formal verification (static type-checking based on explicit typing) is available here. To provide a formal verification by a theorem prover or other semantic verifications leave for future work.

An Implementation of the Mental Model Reasoning System

An outline of the reasoning system is given in Sugimoto, Sato, and Nakayama (2013). The system is portrayed as a conceptual diagram Fig. 2 that shows translations from one model to another by the following steps: 1. constructing mental models of premises, 2. integrating these premise models into an initial (integrated) model, 3. drawing a tentative conclusion from the initial model, 4. constructing alternative models by falsification, 5. producing a final conclusion.

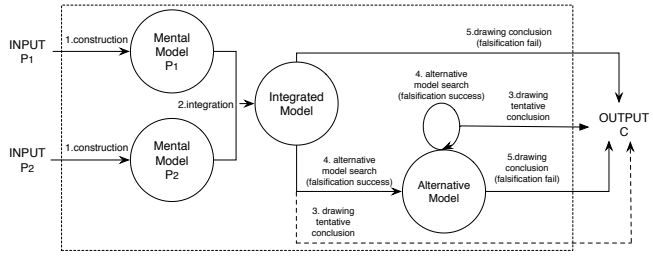


Fig 2: Diagram for syllogistic reasoning system

Steps 1 and 5 involve IO actions. Steps 2, 3 and 4 consist of stateless (no side effects) functions only.

Mental model construction

The syllogistic language (input) is considered as a kind of controlled natural language, and its grammar can be defined by BNF$^+$. We define a combinator parser to parse this language instead of using a standard bottom-up parser (MMRLab.) An implementation of the language and the parser is given in Code 1:

```
pS,pNp,pNegNp :: Parser String String
pPred,pNegPred :: Parser String String
pTerm,pQuant,pNegQuant :: Parser String String
pNeg,pCop :: Parser String String
pS    = pNp <*> pPred <|> pNp <*> pNegPred <|> pNegNp <*> pPred
pNp       = pQuant <*> pTerm
pNegNp    = pNegQuant <*> pTerm
pPred     = pCop <*> pTerm
pNegPred  = pCop <*> pNeg <*> pTerm
pTerm     = symbol "A" <|> symbol "B" <|> symbol "C"
pQuant    = symbol "All" <|> symbol "Some"
pNegQuant = symbol "No"
pNeg      = symbol "not"
pCop      = symbol "are"

type Parser a b = [a] -> [(b,[a])]
symbol :: Eq a => a -> Parser a a
symbol c []               = []
symbol c (x:xs) | c == x    = [(x,xs)]
                | otherwise = []
succeed :: b -> Parser a b
succeed r xs = [(r,xs)]
failp :: Parser a b
failp xs     = []
token :: Eq a => [a] -> Parser a [a]
token cs xs | cs == take n xs = [(cs,drop n xs)]
            | otherwise       = []
        where n = length cs
satisfy :: (a -> Bool) -> Parser a a
satisfy p []                = []
satisfy p (x:xs) | p x      = [(x,xs)]
                 | otherwise = []
```

Code 1: The definition for syllogistic language and its parser

```
data MToken = AToken Atom |
  FToken Exh Atom | NToken Neg Atom | Nil
data Atom  = ASymbol | BSymbol | CSymbol
type Symbol = Char

type Exh = Symbol
type Neg = Symbol
type Nil = Symbol
type ASymbol = Symbol
type BSymbol = Symbol
type CSymbol = Symbol

type MModel = [Indiv]
type Indiv  = [Token]
type Token  = [Symbol]

exh :: Symbol
exh = '*'
asymbol :: Symbol
asymbol = 'a'
bsymbol :: Symbol
bsymbol = 'b'
csymbol :: Symbol
csymbol = 'c'
neg :: Symbol
neg = '-'
```

Code 2: The definition for mental model components

The mental models for syllogistic reasoning are defined in Code 2[4]. A *mental model* is a *class of models*[5] s.t. $m \times n$ matrix (multi-list) of *tokens*. A *row* or an *individual* of a mental model is a finite sequence of tokens (*model*) where each atom occurs at most once. A *column* or a (*property*) of a mental model is a finite sequence of tokens where tokens containing different atoms cannot occur. If square bracketed tokens occur in a column, only negative tokens can be added. The translation from the syllogistic language to mental model representations is performed by monadic parsing (a recursive descent parsing technique well known in functional programming community. See, e.g. Van Eijck and Unger 2010). The parser for syllogistic language constructs abstract syntax trees and then converts mental model representations by the following compositional semantics (Fig. 2). As an example, let X,Y denote terms A, B, C, the four types of syllogistic sentences can be translated to mental models as follows:

All X are Y

$$\Rightarrow \begin{bmatrix} x & y \\ [x] & y \end{bmatrix}$$

Some X are Y

$$\Rightarrow \begin{matrix} x & y \\ x & \\ & y \end{matrix}$$

No X are Y

$$\Rightarrow \begin{matrix} & -y \\ [x] & -y \\ [y] & \\ [y] & \end{matrix}$$

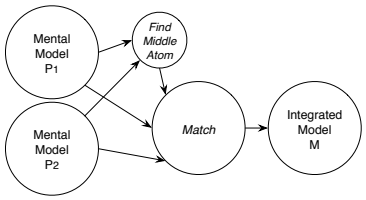
Some X are not Y

$$\Rightarrow \begin{matrix} x & -y \\ x & -y \\ & y \\ & y \end{matrix}$$

[4] In Bucciarelli and Johnson-Laird (1999) "exhaustive model" is represented in square brackets, here we use the "*" symbol as used elsewhere (MMRLab.)

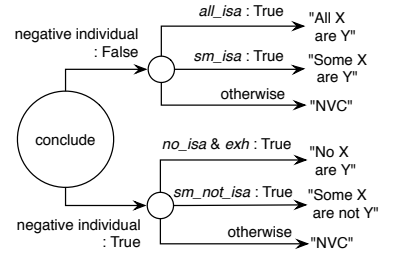
[5] For a treatment of a mental model as a class of models, see Barwise (1993).

(a)



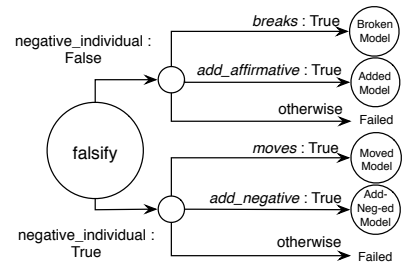

```
combine :: MModel → MModel → MModel
  find_middle_atom :: MMModel → MModel → Atom
  match :: MModel → MModel → Atom → MModel
  join :: Atom → Indiv → Indiv → Indiv
```

(b)




```
conclude :: Symbol → MAtom → Symbol → MModel → Ans
  all_isa :: MModel → Symbol
  sm_isa :: MModel → Symbol
  no_isa :: MModel → Symbol
  sm_not_isa :: MModel → Symbol
```

(c)




```
falsify :: [Symbol] → MModel → MModel
  breaks :: MModel → MModel
  add_affirmative :: MModel → MModel
  moves :: MModel → MModel
  add_negative :: MModel → MModel
```

Fig 3: Process diagrams and their type information: (a) Integration, (b) Drawing a conclusion, (c) Falsification

Integrating Premises into Initial Model

We give a description of the integration process of premises into an initial model via mid term tokens (Fig. 3-a.) The integration function `combine` takes two models (premises) P_1, P_2 and returns an integrated model M with the help of find_a_middle_atom. The type signature for this function is *integration* :: $MModel \rightarrow MModel \rightarrow MModel$. This is implemented as follows:

```
combine :: MModel -> MModel -> MModel
combine mod1 mod2 =
  match mid_atom mod1 mod2
    where mid_atom = find_middle_atom mod1 mod2
```

Reordering and Switching Since syllogisms have several "figures" according to the order of their premises and term arrangements, the actual integration should occur after the preprocesses of reordering terms and switching premises. This preprocessing consists of the following four patterns: (1) If the term order of P_1 is AB and P_2 is BC, nothing happens. (2) If the term order of P_1 is BA and P_2 is CB, integration starts with P_2. (3) If the term order of P_1 is AB and P_2 is CB, the second model is swapped round and added. (4) If the term order of P_1 is BA and P_2 is BC, the first model is swapped round and the second model added.

Finding a middle atom The function `find_middle_atom` has a type signature `find_middle_atom :: MModel → MModel → Symbol`. The implementation of this is similar to a set intersection operation for the affirmative tokens (tokens that do not contain negatives.) An example is when two premises are presented as in Fig.1, $\{a,a,b,b\} \cap \{c,c,b,b\} = b$. The following is an implementation:

```
find_middle_atom :: MModel -> MModel -> Symbol
find_middle_atom mod1 mod2
  | null mod1 = []
  | not (null mid) = mid
  | otherwise = find_middle_atom (tail mod1) mod2
  where mid = find_middle_from_ind (head mod1) mod2
```

Match The function for matching premises P_1, P_2, and middle atom a has the type signature `match :: MModel → MModel → Symbol → MModel`. This function recursively calls `join_` to join the premises to an integrated model.

```
match :: Symbol -> MModel -> MModel -> MModel
match mid_atom mod1 mod2
  | null mod1 = mod2
  | not (null (find_poslis_in_indiv mid_atom $ head mod1)) =
    joining mid_atom (head mod1) mod2 :
    (match mid_atom
           (tail mod1)
           (remove_indiv (find_poslis_in_mod mid_atom mod2) mod2))
  | otherwise = head mod1 : (match mid_atom (tail mod1) mod2)
```

Join The recursive function `join_` takes a mid atom and two individuals, and joins two individuals together setting the new mid to exhausted if either the first individual or second individual were exhausted. The type signature of `join_` is: `join_:: Symbol → Indiv → Indiv → Indiv`

```
join_ :: Symbol -> Indiv -> Indiv -> Indiv
join_ mid_atom indiv1 indiv2 =
  if exhausted $ find_poslis_in_indiv mid_atom indiv1
  then indiv1 ++ remove_lis (find_poslis_in_indiv mid_atom indiv2) indiv2
  else remove_lis (find_poslis_in_indiv mid_atom indiv1) indiv2
```

Drawing a Conclusion from a Model

Drawing a conclusion (Fig.3-b) is a function that takes an integrated (initial) model and dispatches whether it contains negative token or not. Based on the predicates (all_isa, some_isa, no_isa, and sm_not_isa) it then dispatches further and returns *corresponding answers* (action). conclude has type signature: `conclude :: Symbol → Symbol → Symbol → MModel → [Symbol]` [6]. If the predicates return `False`, then it returns `"no valid conclusion"`. The below is an implementation:

[6]Note: since possible conclusions have term order: Subj-Obj and Obj-Subj, `conclude` is executed twice. For simplicity, we omit the second execution of `conclude`.

```
conclude :: Symbol -> Symbol -> Symbol -> MModel -> [Symbol]
conclude subj mid_atom obj model =
  if not (negative_individual model)
  then if all_isa subj obj model
       then "all " ++ [subj] ++ " are " ++ [obj]
       else if sm_isa subj obj model
            then "some " ++ [subj] ++ " are " ++ [obj]
            else "no valid conclusion"
  else if no_isa subj obj model &&
          ((exh_ subj model && exh_ obj model) ||
           (exh_ mid_atom model && exh_ subj model) || (exh_ obj model))
       then "no " ++ [subj] ++ " are " ++ [obj]
       else if sm_not_isa subj obj model
            then "some " ++ [subj] ++ " are " ++ "not " ++ [obj]
            else "no valid conclusion"
```

The following are sub functions called by `conclude`:

all_isa takes a model *M* that has the end terms X, Y and returns True iff all subjects are objects in individuals in a model, then *conclude* returns the answer All X are Y. This has a type signature: `all_isa :: Symbol → Symbol → MModel → Bool`. For example, if a model *M*: $\begin{matrix}[a] & b & c\\ [a] & b & c\end{matrix}$ is given, where the end terms are *A* and *C*, it returns the answer "All A are C." This is implemented as follows:

```
all_isa :: Symbol -> Symbol -> MModel -> Bool
all_isa subj obj model =
  sm_isa subj obj model && not (sm_not_isa subj obj model)
```

sm_isa takes a model *M* that has end terms X, Y and returns True iff at least one individual in the model contains positive occurrences of both subject and object atoms. Then conclude returns the answer "Some X are Y". This has the type signature: `sm_isa :: Symbol → Symbol → MModel → Bool`. For example, if a model: $\begin{matrix}[a] & [b] & c\\ [a] & [b] & \\ & & c\end{matrix}$ is given where the end terms are *A* and *C*, it returns the answer "Some A are C".

```
sm_isa :: Symbol -> Symbol -> MModel -> Bool
sm_isa subj obj model
  | null model = False
  | not (null $ find_poslis_in_indiv subj $ head model) &&
    not (null $ find_poslis_in_indiv obj $ head model) = True
  | otherwise = sm_isa subj obj $ tail model
```

no_isa takes a model *M* that has the end terms X, Y and returns True iff no subject end term is object end term in any individuals in the model, conclude returns "No X are Y". This has a functional type: *no-isa* : $M \rightarrow A$. For example, if a model *M*: $\begin{matrix}[a] & -b\\ [a] & -b\\ [b] & [c]\\ [b] & [c]\end{matrix}$ is given where the end terms are *A* and *C*, it then returns the answer "No A are C".

```
no_isa :: Symbol -> Symbol -> MModel -> Bool
no_isa subj obj model =
  sm_not_isa subj obj model && not (sm_isa subj obj model)
```

sm_not_isa takes a model *M* that has the end terms X, Y and returns True iff at least one subject occurs in individuals without an object, then conclude returns "Some X are not Y". This has a functional type: *sm_not_isa*: $M \rightarrow A$. For example, if a model *M*: $\begin{matrix}[a] & b & c\\ [a] & b & c\\ & -b & c\\ & -b & c\end{matrix}$ is given where the end terms are *A* and *C*, it returns the answer "Some A are not C".

```
sm_not_isa subj obj model
  | null model = False
  | not (null $ find_poslis_in_indiv subj $ head model) &&
    not (null $ find_poslis_in_indiv obj $ head model) = True
  | otherwise = sm_not_isa subj obj $ tail model
```

Falsification

Once the mental model theory constructs an initial model and draws a tentative conclusion, the theory, according to its rules, tries to construct an alternative model to refute the conclusion (the default assumption). The falsification function (Fig. 3-c) takes a model and dispatches whether or not it contains a negative token. Then, based on the predicates (breaks, add_affirmative, moves, and add_negative), it tries to modify the model. If successful, it returns an alternative model and calls conclude recursively. If it fails, this function terminates and conclude outputs a final conclusion. The below is an implementation of falsify:

```
falsify :: [Symbol] -> MModel -> MModel
falsify concl model =
  if not (negative_individual model)
  then if not (null br_model) then br_model
       else if not (null ad_model) then ad_model
            else []
  else if not (null mv_model) then mv_model
       else if not (null an_model) then an_model
            else []
  where
    br_model = breaks concl model
    ad_model = add_affirmative concl model
    mv_model = moves concl (neg_breaking concl model)
    an_model = add_negative concl model
```

Here are the main constructs of `falsify`:

breaks finds an individual containing two end terms with non-exhaustive mid terms, divides it into two, and then either returns new (broken) model or returns *nil*. Its type signature is: `breaks : MModel → MModel`. For example: when a model *M* is $\begin{matrix}a & b & c\end{matrix}$, then breaks M $\rightsquigarrow$ $\begin{matrix}a & b\\ & b & c\end{matrix}$. This is implemented as follows:

```
breaks :: [Symbol] -> MModel -> MModel
breaks concl model =
  if falseif model newmod
  then newmod
  else []
    where newmod = breaking concl model
```

add_affirmative has the type signature: add_affirmative :: MModel → MModel. If add_affirmative succeeds, then it returns a new model with added item (added model), else it returns nil if the conclusion is not A-type ("All X are Y") or if there is no addable subject item. For example, if a given model *M* is $\begin{matrix}[a] & [b] & c\\ [a] & [b] & c\end{matrix}$, then add_affirmative *M* $\rightsquigarrow$ $\begin{matrix}[a] & [b] & c\\ [a] & [b] & c\\ & & c\end{matrix}$.

```
add_affirmative :: [Symbol] -> MModel -> MModel
add_affirmative concl model =
  if not (a_conclusion concl) then []
  else if not (null subj)
       then if subj == first then [[first]] : model
            else if subj == last then model ++ [[[last]]] else []
       else []
    where
      subj      = addable (subject concl) model
      end_terms = find_ends concl model
      first     = head end_terms
      last      = head $ tail end_terms
```

moves has the type signature: moves :: MModel → MModel. If there are exhausted end items not connected to other end items or their negatives (i.e E-type ("No X are Y") conclusions), and if the other end items are exhausted, or O-type ("Some X are not Y") conclusions, then it joins them. Otherwise it joins one of each and returns nil if the first end item

cannot be moved, regardless of whether or not a second one can be. E.g., if a given model M is

$$\begin{array}{cccc} [a] & -b & & \\ [a] & -b & & \\ & [b] & -c & \\ & [b] & -c & \\ & & [c] & \\ & & [c] & \end{array}$$, then moves $M \rightsquigarrow$ $$\begin{array}{ccc} [a] & -b & [c] \\ [a] & -b & [c] \\ & [b] & -c \\ & [b] & -c \end{array}$$.

When this function is called by falsify, neg_breaking (similar procedure to breaks) is also called as an argument.

```
moves :: [Symbol] -> MModel -> MModel
moves concl model =
  if falseif model newmod
  then newmod
  else []
    where newmod = moving concl model
```

add_negative has the type the signature: add_negative :: MModel → MModel. It returns a new model with the added item (add_neged model), or returns nil if the conclusion is not O-type or if there is no addable subject item.
E.g., if a given model M is

$$\begin{array}{cccc} [a] & b & & \\ [a] & b & & \\ & -b & c & \\ & -b & c & \end{array}$$, then add_negative $M \rightsquigarrow$ $$\begin{array}{ccc} [a] & b & c \\ [a] & b & c \\ & -b & c \\ & -b & c \end{array}$$.

```
add_negative :: [Symbol] -> MModel -> MModel
add_negative concl model =
  if falseif model newmod
  then newmod
  else []
    where newmod = adding_neg concl model
```

Concluding Remarks

We have proposed *type system based modeling* as a novel approach in (specification-aware) cognitive modeling. To show the advantages of this approach, we have implemented the syllogistic reasoning system with mental models in Haskell (a popular pure functional statically typed programming language). Compared with other approaches, our implementation includes some aspects of theory specification such as mental model definitions and type information for each process. Our type system based modeling sheds light on the ambiguity problem of mental model theory, which has been discussed at the crossroad between cognitive science and logic. In the modeling work done so far, we have analyzed mental model reasoning for syllogistic fragments. However, our work is not limited to this. There is plenty of scope for further work in several kinds of mental model reasoning (Khemlani & Johnson-Laird, 2013; Johnson-Laird, Byrne, & Tabossi, 1989; Johnson-Laird & Byrne, 1989); more generally, various cognitive activities involving propositions taking some semantic values.

Acknowledgements

This study was supported by MEXT-Supported Program for the Strategic Research Foundation at Private Universities and Grant-in-Aid for JSPS Fellows (25·2291). The authors are grateful to Sangeet Khemlani for helpful comment and Micah Clark for opportunity to read his paper under submission.

References

Arkoudas, K., & Bringsjord, S. (2008). Toward formalizing common-sense psychology: An analysis of the false-belief task. In *PRICAI 2008* (pp. 17–29). Springer.

Barwise, J. (1993). Everyday reasoning and logical inference. *Behavioral and Brain Sciences*, *16*(2), 337–338.

Bosse, T., Jonker, C. M., & Treur, J. (2006). Formalization and analysis of reasoning by assumption. *Cognitive Science*, *30*(1), 147–180.

Braine, M. D. (1994). Mental logic and how to discover it. In *The logical foundation of cognition* (pp. 241–263). Oxford Univ Pr.

Bucciarelli, M., & Johnson-Laird, P. N. (1999). Strategies in syllogistic reasoning. *Cognitive Science*, *23*(3), 247–303.

Clark, M. H. (submitted). Mathematical description of sentential mental models theory: Reconstructing Johnson-Laird's mental models.

Cooper, R. (1992). *A sceptic specification of Johnson-Laird's "mental models" theory of syllogistic reasoning* (Tech. Rept. UCL-PSY-ADREM-TR4 (2nd ed.)). Department of Psychology, University College London.

Cooper, R. (2002). *Modelling high-level cognitive processes*. Psychology Press.

Cooper, R., Fox, J., Farringdon, J., & Shallice, T. (1996). A systematic methodology for cognitive modelling. *Artificial Intelligence*, *85*(1), 3–44.

Cooper, R., & Guest, O. (2014). Implementations are not specifications: Specification, replication and experimentation in computational cognitive modeling. *Cognitive Systems Research*, *27*, 42–49.

Haftmann, F. (2010). From higher-order logic to haskell: there and back again. In *Proceedings of the 2010 ACM SIGPLAN workshop on partial evaluation and program manipulation* (pp. 155–158).

Hintikka, J. (1987). Mental models, semantical games and varieties of intelligence. In *Matters of intelligence* (pp. 197–215). Springer.

Isaac, A. M., Szymanik, J., & Verbrugge, R. (2014). Logic and complexity in cognitive science. In *Johan van Benthem on logic and information dynamics* (pp. 787–824). Springer.

Johnson-Laird, P. N. (1983). *Mental models*. Harvard Univ Press.

Johnson-Laird, P. N., & Bara, B. G. (1984). Syllogistic inference. *Cognition*, *16*(1), 1–61.

Johnson-Laird, P. N., & Byrne, R. M. (1989). Only reasoning. *Journal of Memory and Language*, *28*(3), 313–330.

Johnson-Laird, P. N., Byrne, R. M., & Tabossi, P. (1989). Reasoning by model: The case of multiple quantification. *Psychological Review*, *96*(4), 658–673.

Jones, S. L. P. (2003). *Haskell 98 language and libraries: the revised report*. Cambridge University Press.

Khemlani, S., & Johnson-Laird, P. N. (2012). Theories of the syllogism: A meta-analysis. *Psychological Bulletin*, *138*(3), 427-457.

Khemlani, S., & Johnson-Laird, P. N. (2013). The processes of inference. *Argument & Computation*, *4*(1), 4–20.

Koralus, P., & Mascarenhas, S. (2013). The erotetic theory of reasoning: Bridges between formal semantics and the psychology of deductive inference. *Philosophical Perspectives*, *27*(1), 312–365.

Marlow, S. (2010). *Haskell 2010 language report.* Retrieved from `http://www.haskell.org/onlinereport/haskell2010`

McClelland, J. L. (2009). The place of modeling in cognitive science. *Topics in Cognitive Science*, *1*(1), 11–38.

Mental Models & Reasoning Lab. (n.d.). *Syllogistic reasoning code [computer program].* Retrieved from `http://mentalmodels.princeton.edu/programs/Syllog-Public.lisp`

Mitchell, J. C. (2003). *Concepts in programming languages*. Cambridge University Press.

Moss, L. S. (2008). Completeness theorems for syllogistic fragments. *Logics for linguistic structures*, *29*, 143–173.

Stenning, K., & Van Lambalgen, M. (2008). *Human reasoning and cognitive science*. MIT Press.

Sugimoto, Y., Sato, Y., & Nakayama, S. (2013). Towards a formalization of mental model reasoning for syllogistic fragments. *In Proceedings of the 1st International Workshop on Artificial Intelligence and Cognition, CEUR Vol.1100*, 140–145.

Van Eijck, J., & Unger, C. (2010). *Computational semantics with functional programming*. Cambridge University Press.

Verbrugge, R. (2009). Logic and social cognition. *Journal of Philosophical Logic*, *38*(6), 649–680.

Vytiniotis, D., Peyton Jones, S., Claessen, K., & Rosén, D. (2013). Halo: Haskell to logic through denotational semantics. *ACM SIGPLAN Notices*, *48*(1), 431–442.

Modeling the Workload Capacity of Visual Multitasking

Leslie M. Blaha (leslie.blaha@us.af.mil) and James Cline (james.cline.ctr@us.af.mil)
711th Human Performance Wing, Air Force Research Laboratory
Wright-Patterson AFB, OH 45433 USA

Tim Halverson (thalverson@gmail.com)
Oregon Research in Cognitive Applications, LLC
Oregon City, OR 97045 USA

Abstract

We utilize the capacity coefficient to characterize the workload capacity of visual multitasking. The capacity coefficient compares cognitive work completed on multiple information sources against a baseline parallel model prediction. Capacity coefficient results subsume standard mean response time (RT) dual-task findings while providing a description of workload effects on the whole RT distribution. This yields a theoretically-grounded characterization that can inform computational and process models of multitasking.

Keywords: Workload Capacity; Multitasking; Multi-Attribute Task Battery; Human Information Processing; Dual-Task

Introduction

We seek to provide a better mathematical characterization of cognitive performance during multitasking, the simultaneous execution of more than one task within the same experimental environment. Often, characterizations of multitasking performance are limited to assessments of dual task decrements, wherein mean response time (RT) or accuracy are compared across only two tasks. Increased workload is inferred when an RT increase and/or accuracy decrease is observed when switching from a single task to the dual-task environment. These performance metrics may be further correlated with subjective workload ratings. While these measures do give some indication of participants' experiences of workload, they do not provide strong insight into the cognitive mechanisms supporting multitasking behaviors or the mechanistic reasons for changes in performance under changing workload demands.

We report on an effort to utilize human information processing modeling to provide qualitative and quantitative characterization of the cognitive mechanisms engaged in multitasking. In particular, we focus on changes in workload capacity, the efficiency with which the system responds to the changing number of tasks in a dynamic environment.

Modified Multi-Attribute Task Battery

To study multitasking, we utilize a web-browser implementation of the modified Multi-Attribute Task Battery (mMATB; Cline, Arendt, Geiselman, & Blaha, 2014), developed in the JavaScript D3 library (Bostock, Ogievetsky, & Heer, 2011). The mMATB consists of four possible visual decision making tasks: Tracking, Monitoring, Communication, Resource Management. In our implementation, all aspects of the workload can be manipulated: entire tasks (quadrants) can be turned on or off, the rate of alerting events can be varied as

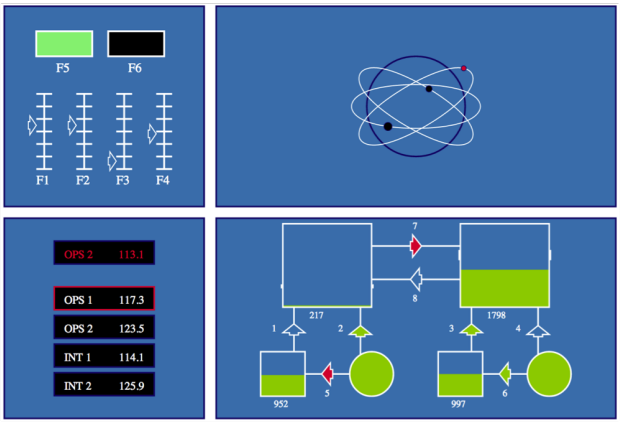


Figure 1: The browser-based modified Multi-Attribute Task Battery (mMATB) used in the present study. The four visual tasks are (clockwise from upper left): Monitoring, Tracking, Resource Management, Communications.

can the probability of simultaneous alerting events, and the speeds at which the moving parts of the displays move can be adjusted. We will focus herein on manipulations of the total number of tasks to be performed simultaneously.

Figure 1 shows the mMATB environment. The Tracking Task, contained in the upper right, entails physically tracking three colored circles moving continuously along individual ellipsoid trajectories. High performance on this task requires continual mouse motion and attention to switching targets.

Both the Monitoring Task (upper left) and the Communications Tasks (lower left) require keypress responses to alert events. In the Monitoring Task, the participant's task is to provide the appropriate response if a parameter is out of its normal state. In the Communications Task, participants must adjust a channel to a new value upon target cuing.

The lower right quadrant contains a Resource Management Task which requires only strategic attention to gates (switched by keypress) in order to maintain fuel levels within a predetermined range for two schematic resource tanks.

The mMATB, thus, demands a division of visual attention and motor activity across the four tasks. During multitasking, participants are instructed to emphasize accuracy in the Tracking Task as their primary task, and to respond to all other alerts appropriately. RTs are collected to cued events; response choices are collected for all interactions.

The Capacity Coefficient

Workload capacity is defined as the ability of the cognitive information processing mechanisms to respond to changes in cognitive load. This is usually interpreted as changes in the number of items that need to be processed within a task. Capacity is assessed with RT data, in order to make inferences about information processing speeds. Qualitatively, the possible capacity classes are unlimited, super, and limited capacity, corresponding to processing speeds remaining steady, increasing, or decreasing, respectively.

Our primary measure of workload capacity is the capacity coefficient (Houpt, Blaha, McIntire, Havig, & Townsend, 2014). This is a ratio measure which compares the observed participant's RTs during multitasking to a model-based prediction about multitasking speed. The baseline RT model is an unlimited capacity independent parallel model (UCIP). We utilize the capacity coefficient for ST-ST responses (Blaha, 2010):

$$C(t) = \frac{K_k(t)}{K_{k,C}(t)}. \quad (1)$$

In Equation 1, the numerator gives the cumulative reversed hazard function for individual target channel k when processed alone; this is the UCIP model prediction. The denominator is the cumulative reversed hazard function for target channel k when additional tasks (set C) are performed.

Figure 2 illustrates $C(t)$ results for one typical participant in both dual-task multitasking (upper plot) and four-task multitasking (lower plot). Relative to the UCIP baseline at $C(t) = 1$, the data indicate that while while all conditions showed mean RT dual-task decrements, the functional data are more nuanced. Under dual-task conditions, performance in the tracking task improved, showing super capacity $C(t) > 1$ for most times, but falling to limited capacity $C(t) < 1$ when the number of tasks increased to four. Thus, additional task demands have the potential to improve tracking performance.

Detection performance in the monitoring task, on the other hand, was limited capacity in both the dual task and four-task multitasking conditions. Communication task detection was also limited capacity in the four-task condition. This indicates that division of attention across multiple tasks slows alert detection responses.

Discussion

The present work is the first to apply the capacity coefficient to a multitasking situation, where the number of tasks is manipulated while the features within each task (when present) remain unchanged. Current results indicate that some tasks benefit from additional workload demands, while others are slowed. The capacity coefficient can capture both types of effects. This more nuanced characterization can then be used to inform computational and process models of multitasking (e.g., Salvucci & Taatgen, 2008), and to study task switching and divided attention strategies.

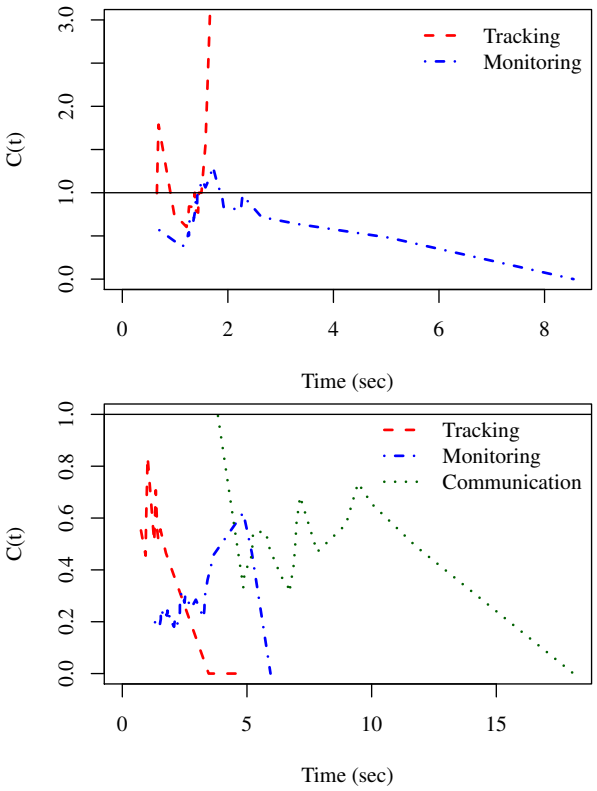


Figure 2: Capacity coefficient results for a typical participant in the dual task (upper) and multitask (lower) conditions.

Acknowledgments

This research was funded by the Air Force Office of Scientific Research. Distribution A: Approved for public release; distribution unlimited. 88ABW Cleared 12/16/2014; 88ABW-2014-5937.

References

Blaha, L. M. (2010). *A dynamic hebbian-style model of configural learning* (Unpublished doctoral dissertation). Indiana University, Bloomington, Indiana.

Bostock, M., Ogievetsky, V., & Heer, J. (2011). D3: Data driven documents. *IEEE Transactions on Visualization and Computer Graphics*, 2301–2309.

Cline, J., Arendt, D. L., Geiselman, E. E., & Blaha, L. M. (2014, May). Web-based implementation of the modified multi-attribute task battery. In *Fourth annual midwestern cognitive science conference.* Dayton, Ohio.

Houpt, J. W., Blaha, L. M., McIntire, J. P., Havig, P. R., & Townsend, J. T. (2014). Systems Factorial Technology with R. *Behavior Research Methods*, *46*, 307–330.

Salvucci, D. D., & Taatgen, N. A. (2008). Threaded cognition: an integrated theory of concurrent multitasking. *Psychological review*, *115*(1), 101.

SIMCog-JS: Simplified Interfacing for Modeling Cognition - JavaScript

Tim Halverson (thalverson@gmail.com)
Oregon Research in Cognitive Applications, LLC
Oregon City, OR 97045 USA

Brad Reynolds (reynolds.157@wright.edu)
College of Engineering and Computer Science, Wright State University
Dayton, OH 45435 USA

Leslie Blaha (leslie.blaha@us.af.mil)
711th Human Performance Wing, Air Force Research Laboratory
WPAFB, OH 45433 USA

Abstract

A continuing hurdle in the cognitive modeling of human-computer interaction is the difficulty with allowing models to interact with the same interfaces as the user. Multiple attempts have been made to add this functionality (e.g., Hope, Schoelles, & Gray, 2014) in limited domains. This paper presents a solution allowing models to interact with web browser-based software, while requiring little modification to the task code. Simplified Interfacing for Modeling Cognition - JavaScript (SIMCog-JS) allows the modeler to specify how elements in the interface are translated into ACT-R chunks, allows keyboard and mouse interaction with JavaScript code, and allows sending ACT-R commands from the external software (e.g., to add instructions). The benefits, drawbacks, and future functionality of SIMCog-JS are discussed.

Keywords: Cognitive Architectures; Task Interface; ACT-R; WebSockets; JSON; HTML; JavaScript; D3

Introduction

A substantial challenge with modeling human cognition is the presentation of task environments to the simulated human. Software re-implementation provides little scientific reward, yet modelers face this burden every time they utilize a new or modified task. The situation is further complicated if a modeler is studying human-computer interaction (HCI) with complex software in which users are engaging in ongoing, dynamic, and interleaved or multi-tasking behaviors. Because the focus of cognitive modeling in HCI is often either explaining or predicting performance differences between alternative interfaces, substantial research time is spent re-implementing multiple, complex interfaces; this effort is further multiplied if multiple cognitive architectures are used.

Although re-implementation within a modeling architecture framework can allow maximum control by the modeler, it introduces additional challenges: (a) Re-implementation increases the likelihood that the fidelity of the simulation is degraded by an imperfect porting of the user interface or task dynamics. (b) Iterative changes to the original software/task require additional efforts to integrate these changes into the model's task environment. (c) Task-simulation environments for cognitive architectures are sometimes written in programming languages not commonly used for building HCI interfaces (e.g., ACT-R uses Lisp; Anderson et al., 2004) and often provide limited facilities for building the task simulations. Thus, the process of re-implementation forces a trade-off between task fidelity and time savings. An alternative to re-implementation is to allow a model to communicate directly with a user interface that is external to the cognitive architecture. Previous research has attempted to solve this challenge, although in limited domains. Computer vision (CV) has been used to automatically extract relevant visual features from an existing computer interface (e.g., Halbrügge, 2013; St Amant, Riedl, Ritter, & Reifers, 2005). While CV solutions remove the burden of "translating" the interface to symbols understood by the architecture, they also reduce the control the modeler has on how the visual interfaces are specified. Additional control requires the modeler to customize the CV algorithms or specify screen element "templates" at the pixel level. Other solutions provide the ability for models to act within specialized environments, like games (e.g., Veksler, 2009) or robotics (e.g., Kennedy, Bugajska, Adams, Schultz, & Trafton, 2008). These solutions are incredibly useful but are limited to their specialized environments. Still other solutions provide a more general framework for interfacing models with external software by using interprocess communication protocols available in many programming languages (e.g., Büttner, 2010; Hope et al., 2014). The solution presented herein falls into this final category.

We present a solution to the challenge of communication between external task environments and cognitive architectures: Simplified Interfacing for Modeling Cognition - JavaScript (SIMCog-JS). Our approach supports communication between Java ACT-R (Salvucci, 2013) and HTML-/JavaScript-based software in a user-friendly manner. In the remainder of this article, we specify some design requirements, describe the functionality provided by SIMCog-JS, and provide an example of SIMCog-JS applied to a dynamic, multitasking experiment environment.

SIMCog-JS Design Requirements

SIMCog-JS is a technology that allows cognitive modelers to specify how visual information is extracted from external software, passes that information to ACT-R, and passes keyboard and mouse events back to the external software. The primary motivation for SIMCog-JS is rooted in a desire to

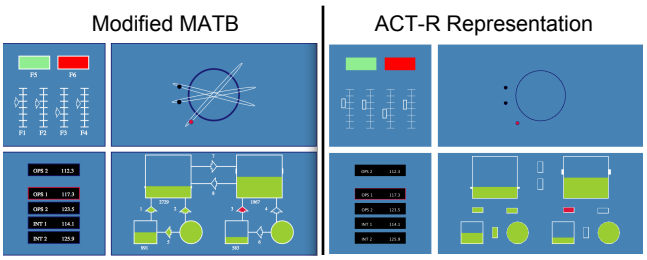


Figure 1: The browser-based modified Multi-Attribute Task Battery (mMATB) as it would appear to a human participant (left) and a representation of the ACT-R visicon (right).

apply cognitive architectures to dynamic, multitasking experiments, such as training simulations or naturalistic web-browsing. We desire a flexible system allowing interaction between multiple cognitive modeling formalisms and existing software/HCI environments.

SIMCog-JS attempts to minimize the modeler's burden in multiple ways. First, SIMCog-JS requires minimal modification of existing task code, although it does require that the modeler have access to the JavaScript task code. Second, SIMCog-JS includes an extension to Java ACT-R that replaces Java ACT-R task code and requires no modifications for a wide variety of tasks. Third, SIMCog-JS provides a user-friendly, flexible syntax for specifying which visual elements should be passed to ACT-R, when those elements should be updated in ACT-R, and how they should be perceived (i.e., slot values).

In order to provide this functionality, SIMCog-JS had three critical design requirements:

1. SIMCog-JS must use standard software protocols for communication between models and experimental software.
2. Integrating model interactions with the task makes minimal modifications to the experimental code, minimizing interference with human data collection or natural behaviors.
3. Model execution occurs in real time.[1]

We note that as our initial target task environment, the modified Multi-Attribute Task Battery (mMATB), executes in a web-browser and the modeling formalism, Java ACT-R, is written in Java, we were required to implement a new solution to facilitate interaction between cognitive models and a task environment. Hope et al. (2014) introduced a similar solution for interfacing Lisp ACT-R with stand-alone software. However, that published solution does not support either Java or JavaScript. Our solution took motivation from Hope et al.'s work.

The mMATB Task Environment

We apply SIMCog-JS to a dynamic, multitasking environment, mMATB (Cline, Arendt, Geiselman, & Blaha, 2014). This is a generalized version of the MATB developed to assess multitasking in pilot-like environments (Arnegard & Comstock, 1991); the modifications in this environment make similar cognitive demands on the participants, but the tasks are less pilot-specific in nature. Our browser-based implementation is written with the D3 JavaScript library (Bostock, Ogievetsky, & Heer, 2011) integrated with a Python django database. Participants interact with the environment through keyboard button presses and mouse clicks and movements.

The mMATB, shown in the left panel of Figure 1, entails four separate tasks, which we summarize clockwise from the upper left. The upper left quadrant is a Monitoring Task, consisting of a set of sliders and two color indicator blocks. The participant's task is to provide the appropriate button press (F1-F6, labeled on each indicator/slider) if a parameter is out of its normal state. For the sliders, this means moving above or below ± 1 notch from the center. For the indicators, the normally green (black) might turn black (red).

A Tracking Task is contained in the upper right quadrant, wherein three colored circles move continuously along individual ellipsoid trajectories. At any time, one of the circles may turn red, indicating it is the object to be tracked by the participant. The participant tracks the target by mousing to the target, clicking on it, and then following it with the mouse, until the next target object is indicated with a color change.

The lower right quadrant contains a Resource Management Task. Two resource tanks are schematically illustrated, together with representations of fuel sources, reserve tanks, and gated connections (each numbered 1-8) between all tanks. The participant's task is to maintain the resource levels within a range specified by bars on the sides of the tanks. The on/off states of the gates are controlled with number pad key presses. The participant can control the gates with any strategy of choice to maintain the resource levels.

Finally, the lower left quadrant contains a Communications Task. The display shows four channels (Int1, Int2, Ops1, Ops2) together with the current channel values; the topmost line gives a target channel and value. If a red cued target appears in the top box, the participant uses the up/down arrow keys to select the cued channel and the right/left arrow keys to adjust the channel value to the new cued value. The enter key submits the corrected channel, which changes the topmost cue box to white until the next channel cue appears.

Cognitive modeling of mMATB performance aims to capture behavioral impacts of changes in workload, operator stress levels, or fatigue levels and to characterize the high-level strategies engaged during continuous multitasking.

SIMCog-JS Software Architecture

SIMCog-JS uses a client-server software architecture. The server exists within Java ACT-R (Salvucci, 2013) as a "generic task." This generic task is populated with environment-specific information as the server receives messages from a client describing the current state of a task interface. The server dynamically changes the ACT-R envi-

[1] As our target software does not support synchronized execution with external software. This is not a constraint unique to our target software, as web browsers (and most software) do not allow external synchronization.

ronment based on the messages received from the client and sends messages to the client describing ACT-R's actions.

The client is built in JavaScript, allowing it to run within all modern web browsers. The client runs alongside the browser-based task translating the task interface for the server and processing interactions from ACT-R. The client is integrated into existing code by referencing the client script in the task's primary web page (e.g., index.hmtl). Further, the modeler specifies three things within the client: (a) a list of visual chunks to be represented in ACT-R, (b) a list of ACT-R commands for the model, and (c) handlers for interactions received from the task.[2] Once these are in place, the system is ready for use.

The client and server communicate via WebSockets and JavaScript Object Notation-Remote Procedure Call (JSON-RPC). WebSockets (Fette & Melnikov, 2011) allow reliable, simultaneous connections between the client and server.[3] Once connected, the client and server use JSON-RPC (JSON-RPC Working Group, 2010) to send information. JSON-RPC is a standardized protocol for sending messages based on the JSON standard. Both the WebSocket and JSON-RPC protocols are standards that have been implemented in many programming languages, allowing SIMCog to be easily extended to task interfaces and cognitive modeling formalisms in other programming languages through the use of these standard protocols and reuse of the SIMCog-JS's messaging specification. WebSockets and JSON are native to JavaScript, but requires additional libraries for Java.

Figure 2 shows the flow of information between the browser task environment (i.e., client), the server, and ACT-R. After Java ACT-R and the client are configured and running, the client sends all information about the interface to the server at the start of the task, along with any initial ACT-R commands. As keyboard and mouse events are generated by ACT-R, these actions are passed to the client to affect the interface. Details on how to configure the client and server can be found with the SIMCog-JS Software Design Document included in SIMCog-JS distribution.[4] The following section provides details on how this communication takes place between the client and ACT-R.

Communicating through SIMCog-JS

SIMCog-JS allows the modeler to specify how interface elements in software will be represented in ACT-R, to send ACT-R commands from the task client to Java ACT-R, and to determine how the task client will respond to keypresses and cursor movements made by the model. The following sections describe how these three facilities are used.

[2]As discussed in Keypress and Mouse Events section below, default keypress and mouse click handlers are provided for the modeler's convenience.

[3]The client and server may be run on separate computers and over the internet. However, doing so may introduce additional lag that could reduce the fidelity of the simulation.

[4]The SIMCog-JS distribution can be downloaded from: http://sai.mindmodeling.org/simcog/

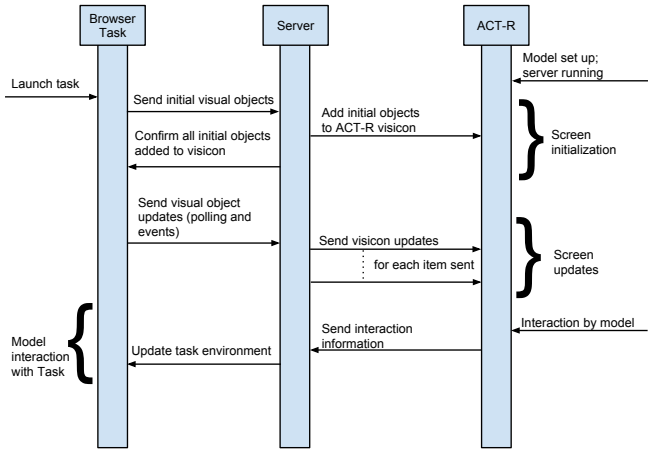


Figure 2: Information flow through SIMCog-JS. The information events start with the modeler initiating the Java ACT-R model and then launching the browser-based task. SIMCog-JS then connects the two environments. The vertical dimension captures time flowing from top to bottom.

Specifying Visual Chunks

The left side of Figure 1 shows the visual interface for mMATB task as presented to the human participant in the web browser. The right side of Figure 1 shows a visual representation of ACT-R's visicon, as specified by the modeler and displayed by the SIMCog-JS server. The modeler specifies which web-browser elements become visual chunks in ACT-R, how those elements will be represented in ACT-R, and when those elements will be updated. SIMCog-JS does not send all interface elements to ACT-R; doing so could unnecessarily complicate the modeling. The modeler may have observational data or theoretical reasons for hypothesizing that some interface elements are completely ignored by users. For example, a uniform background frame may have no impact on performance, assuming adequate contrast between the background and foreground elements. Therefore, the modeler must specify the set of interface elements that become visual chunks in ACT-R.

The modeler must specify the interface element `id` and the element's shape `type`. The object's coordinates, width, height, color, and text (if applicable) are automatically extracted from the task interface using JavaScript DOM function calls and jQuery-dependent CSS specificity computations.[5] The syntax for specifying visual objects is:[6]

[5]All attributes can be specified manually. See the Design Document at http://sai.mindmodeling.org/simcog/ for more information and useful links to the libraries utilized.

[6]All syntax descriptions follow the same convention. Angle brackets are used to indicate a value that must be specified by the modeler. Values enclosed in quotation marks indicate that the value is a string. For example, id:"<unique_name>" indicates that unique_name should be replaced by a string that is the value of the id, like id:"foo". Alternative values are separated by "|".

```
{id:"<unique_name>", type:"<valid_type>"}
```

The id uniquely identifies the object. If the interface element has an explicitly labeled id in the document-object model (DOM; e.g.,), that string can be used as the SIMCog-JS id. If an object does not have a unique ID, the id can be specified with two attributes, name and domLocation. The name value is a string that must be unique to the object. It is helpful to make the name meaningful. A domLocation value is the node of the object located within the DOM tree. This node can be found in multiple ways. One way is to identify the relation to another named object within the DOM tree. Another is to locate the object in the DOM tree relative to the root (i.e., document). Syntax for these methods are:

```
{id:{domLocation:document.getElementById(
    "<element_id>").nextElementSibling,
  name:"<unique_name>"}, ...}

{id:{domLocation:document.body.
    lastElementChild,
  name:"<unique_name>"}, ...}
```

The type is a string that determines how the object will be represented within ACT-R. A screen object must be one of nine types: "Line", "Cross", "Label", "Oval", "OvalOutline", "OvalOutlineFill", "Rectangle", "RectangleOutline", or "RectangleOutlineFill". The first three types are "native" to Java ACT-R;[7] the remaining items are custom task components added by the authors. The type of an object is represented in the visual chunk's "isa" attribute (e.g., "OvalOutlineFill" has an attribute of "isa oval"). Additionally, if the shape is specified with two colors (e.g., "OvalOutlineFill" has a fill and outline color), then SIMCog-JS adds a borderColor chunk slot that contains the value of the border's color and the standard ACT-R color slot contains the value of the fill color. The coordinates, dimensions, and colors of objects are determined differently for different object types. If an object is declared with the wrong type, it is likely that the object will be misrepresented in ACT-R.

The modeler may also specify when changes to interface elements are sent to ACT-R. The default is to update whenever the element changes using DOM Mutation Observers. This event-based functionality is most useful when one or more attributes of the interface element changes infrequently. The modeler may also specify that updates occur at a configurable, regular interval (e.g., polling). This polling functionality is most useful when the attributes of objects are rapidly changing. In such cases, the polling method can substantially decrease the number of messages to the server, decreasing computational demands. Specifying polling-based changes is done by adding a change attribute with the value "poll" to the element declaration. Finally, an object can be declared as static. Static elements are never updated. The modeler may specify an object as static by adding a change attribute with the value "static" to the element declaration. Syntax for change declarations is:

```
{..., change:"evt"|"poll"|"static"}
```

In addition to specifying when updates for an object are sent, the modeler may specify which visual properties are updated. By default, all properties are updated. Listing only those properties that will change can improve software performance. For example, a light may only change color but not move, or tracking reticles may only change coordinates but not colors. The list of properties that will be updated are appended to the value given to the change attribute. If no such list is given, all properties are updated. Valid attributes are "x", "y", "height", "width", "color", "secondaryColor", and "stringVal". Only labels have "stringVal" attributes. Syntax of these expanded change declarations are:

```
{..., change:["<attribute_name>",
    "additional_attribute_name>, ...]}
{..., change:["poll", "<attribute_name>",
    "additional_attribute_name>, ...]}
```

It is also possible to add objects to the ACT-R task environment that are not relevant to the model but are useful for the modeler (i.e., for debugging the visual interface). This is done using "task-irrelevant" objects. Task-irrelevant objects never appear in the model's visicon. For example, a task-irrelevant object may be used as a background to make objects easier to see for the modeler. There are four possible task-irrelevant objects: Cross, Label, Line, and Rectangle. Task-irrelevant objects are not updated throughout the task. All objects default to being task-relevant. To declare an object as task-irrelevant, the attribute taskRelevant is added to an object declaration with a value of false. The syntax for this option is:

```
{..., taskRelevant:true|false}
```

Example Specifications from mMATB This section provides examples of how interface elements in the mMATB task, shown in Figure 1, are specified. The examples start with simple specifications and progress to the more complex.

Perhaps the simplest interface elements in mMATB are the background color panels underlying all four quadrants. They never change (i.e., are static), are filled with a single color ("steel blue"), and are rectangular. If one hypothesizes that these background colors are ignored by the users, these elements can be declared as task-irrelevant. Alternatively, the cognitive model could simply ignore these elements, or the modeler could choose to exclude these elements. Making them task-irrelevant will improve software performance ever so slightly. Including them in the interface specification will make the interface in ACT-R more readable. Although the interface element is simple, it is not uncommon for HTML ids to be missing from background elements, which complicates the id for these elements. In this example, the domLocation value is used to determine the id based on the modeler's knowledge of the location of these elements in the DOM tree.

[7]Note that "Button"s are not supported. As discussed later, any type of object can be clickable.

```
{type:"Rectangle",
  id:{name:"svg0",
    domLocation:d3.
      selectAll("svg")[0][0].firstChild},
  change:"static",
  taskRelevant:false}
```

The Monitoring Task color indicator blocks (upper left quadrant) provide a straightforward example for displaying event-based task elements. The following example is the specification of the green color indicator block; the specification of the red block is similar. The id of this rectangular element is known, monitor_button_0. The only property that changes is the color and so the only value assigned to the change attribute is color. The changes are infrequent, normally changing only a few times per second, so the change attribute is given the value of "evt". Note that "evt" is the default and is not required in the declaration.

```
{type:"Rectangle",
  id:"monitor_button_0",
  change:["evt", "color"]}
```

Label interface elements are unique in that they contain text that can be updated. The mMATB Communications Task's channel values provide examples of changing labels. As with the indicator blocks, the ids are known, like "comm_channel_1_frequency" in the example below. However the text of the label changes. In the example below, the change attribute is labeled as event-based (e.g., "evt") because the values rarely change, and only the text of the label is marked for change with " stringval ".

```
{type:"Label",
  id:"comm_channel_1_frequency",
  change:["evt","stringVal"]}
```

The most dynamic elements in the mMATB interface are the colored circles in the Tracking Task. Each oval moves continuously along a path using the D3 animation library. The constant motion produces a lot of events; this could generate a lot of network traffic and decrease software performance. Therefore, these elements are specified with the "poll" value for the change attribute. The location ("x" and "y") and "color" change, and so all three values are listed in the change attribute. The final attribute of the example specification given below is clickable ; this attribute will be described in the next section.

```
{type:"OvalOutlineFill",
  id:"track_circle_0",
  change:["poll","x","y","color"],
  clickAble:true}
```

Keypress and Mouse Events

To complete the interaction loop, actions taken by the model are transmitted to the task environment. There are three types of interaction currently supported by SIMCog-JS: key press, cursor move, and mouse click. The server sends all interactions to the client; the modeler has full control of how to handle (or ignore) events.

The simplest of the three interactions is key press. Key press interactions are handled automatically by the system. This is done by mapping ACT-R keycodes to JavaScript keycodes and dispatching a keydown event to the task. Currently only keydown events are supported; the modeler may modify the client code to support keyup and keypress events.

When a click is performed, a message is sent to the client containing the location of the mouse and the event type (mouseClick). While mouse coordinates may be enough for many tasks, more information is provided, for example, to deal with the asynchronous nature of the system or facilitate a deeper analysis. An example from the mMATB task is when the model clicks on circles in the tracking task that are moving quickly; the circle could move a couple of pixels out from under the cursor before the click event reaches the client. To handle such circumstances, objects can be declared as clickable . Anytime a click is performed by the model, the server determines if the click was performed within any of the clickable objects. If it is determined that one or more objects were clicked, the message to the client will also include the unique IDs of the items clicked, along with the location, type, and ID of every clickable object. This information allows for cases where the unique identifier is needed to click an object within the task and even more complex cases where specific information and computation is desired.

To declare a visual chunk as clickable, add the clickable attribute to an object's specification and set it to true.

```
{..., clickable:true}
```

The client automatically handles clicks by dispatching a JavaScript mouse click event. If a clickable object was clicked, the client dispatches a click event for that element. Otherwise, the client finds the element at the location of the click and simulates the click there.

For mouse movements, JavaScript does not allow control of the cursor in web browsers. Such control is not allowed by code in web browsers for security and usability reasons. To simulate a model's mouse movements in the task, SIMCog-JS generates mouse movement messages for the client. This approach offers both reliability and speed without introducing external software systems.

When the model moves its simulated mouse, a mouseMove message is sent to the client that contains the location of the model's simulated cursor. With this information, the modeler can record the simulated mouse movements similarly to how human mouse movement data are recorded. To do so, the modeler will likely need to modify the client code. For example, in mMATB the cursor-recording code looks like:

```
ws.onmessage = function (evt) {
    //Called when server message received
    var serverMessage = JSON.parse(evt.data);
  ...
  else if(serverMessage.Command == "mouseMove"){
    track_chart.mouseLocation(
      {x:modelInteraction.mouseX,
      y:modelInteraction.mouseY});
    }}
```

Sending ACT-R Commands

SIMCog-JS supports sending model commands from the task to the model. Doing so is straightforward and takes advantage of existing Java ACT-R methods for executing ACT-R commands. The modeler adds commands to a list in the client code that is sent to the server at the start of execution. For example, to represent the Resource Management Task instructions to maintain the resource level within a target range, the modeler may specify:

```
["(add-dm (resourceTask isa goal
      minLevel 2000 maxLevel 3000))",
 "(goal-focus resourceTask)"]
```

Conclusion and Future Work

SIMCog-JS is a system that allows cognitive models to interact with external software, minimizing the task re-implementation burden on the modeler. The system currently facilitates communication between Java ACT-R and HTML/JavaScript. In addition to describing the architecture of SIMCog-JS, this paper reported on using SIMCog-JS to (a) specify visual interface elements for use by ACT-R and how those interface specifications can be customized, (b) integrate ACT-R responses into JavaScript software, and (c) execute ACT-R commands from the task interface. The strengths of SIMCog-JS are the easy specification of visual objects and interactions with minimal task-code modifications and the seamless interaction between models and browser-based tasks. The modeler need only specify the identity and shape for visual objects to reach ACT-R.

Development is ongoing to improve and extend the functionality of SIMCog-JS. A mid-term goal is to add synchronous execution modes, where the task and model use the same simulation clock, relaxing design requirement 3 without negatively impacting real-time execution. Additional planned features include audio event specification and support for multiple cognitive modeling formalisms, like EPIC architecture (Kieras & Meyer, 1997) and Python-based mathematical models.

By harnessing standard programming protocols and languages, the SIMCog approach can lighten the modeler's burden while broadening the environments in which computational cognitive models operate. Because SIMCog-JS can operate in an environment with facilities for complex data visualization (e.g., D3), we will be pushed to enhance ACT-R's functionality. In the future SIMCog-JS could be integrated with an artificial vision system to, for example, automatically determine object shape; this combined approach could, in fact, bolster both candidate solutions to the task re-implementation challenge.

Acknowledgments

This research was supported AFOSR. Distribution A: Approved for public release; distribution unlimited. 88ABW Cleared 12/16/2014; 88ABW-2014-5938.

References

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004, October). An integrated theory of the mind. *Psychological Review*, *111*(4), 1036–1060.

Arnegard, R. J., & Comstock, J. R. (1991, May). Multi-attribute task battery: Applications in pilot workload and strategic behavior research. In *6th international symposium on aviation psychology* (pp. 1118–1123). Columbus, Ohio.

Bostock, M., Ogievetsky, V., & Heer, J. (2011). D3: Data driven documents. *IEEE Transactions on Visualization and Computer Graphics*, 2301–2309.

Büttner, P. (2010). "Hello Java!" Linking ACT-R 6 with a Java simulation. In D. D. Salvucci & G. Gunzelmann (Eds.), *International conference on cognitive modeling* (pp. 289–290). Philadelphia, PA.

Cline, J., Arendt, D. L., Geiselman, E. E., & Blaha, L. M. (2014, May). Web-based implementation of the modified multi-attribute task battery. In *4th annual midwestern cognitive science conference.* Dayton, Ohio.

Fette, I., & Melnikov, A. (2011, November). *The WebSocket Protocol* (Tech. Rep. No. RCF 6455). Internet Engineering Task Force.

Halbrügge, M. (2013). ACT-CV: Bridging the Gap between Cognitive Models and the Outer World. In E. Brandenburg, L. Doria, A. Gross, T. Güntzler, & H. Smieszek (Eds.), *Berliner werk- statt mensch-maschine-systeme* (pp. 205–210). Berlin.

Hope, R. M., Schoelles, M. J., & Gray, W. D. (2014). Simplifying the interaction between cognitive models and task environments with the json network interface. *Behavior Research Methods*, *46*, 1007–1012.

JSON-RPC Working Group. (2010, March). *JSON-RPC 2.0 Specification.* http://www.jsonrpc.org/specification/.

Kennedy, W. G., Bugajska, M. D., Adams, W., Schultz, A. C., & Trafton, J. G. (2008). Incorporating Mental Simulation for a More Effective Robotic Teammate. In *Conference on artificial intelligence* (pp. 1300–1305). Chicago, IL.

Kieras, D. E., & Meyer, D. E. (1997). An overview of the EPIC architecture for cognition and performance with application to human-computer interaction. *Human-Computer Interaction*, *12*(4), 391–438.

Salvucci, D. D. (2013, August). ACT-R: The Java Simulation & Development Environment. *http://cog.cs.drexel.edu/act-r/index.html*.

St Amant, R., Riedl, M. O., Ritter, F. E., & Reifers, A. L. (2005). Image Processing in Cognitive Models with SegMan. In *Human-computer interaction international* (pp. 1869:1–1869:19).

Veksler, V. D. (2009). Second Life as a Simulation Environment: Rich, high-fidelity world, minus the hassles. In *International conference on cognitive modeling.* Manchester, United Kingdom.

Modeling Password Entry on a Mobile Device

Melissa A. Gallagher (melissa.gallagher@rice.edu)
Department of Psychology, MS-25
Houston, TX 77005 USA

Michael D. Byrne (byrne@rice.edu)
Departments of Psychology and Computer Science, MS-25
Houston, TX 77005 USA

Abstract

Password authentication is a widely deployed security feature on desktop and mobile systems. Inputting complex passwords on mobile devices can be an onerous task. The composition of the passwords creates a unique challenge for people to input as not all characters are displayed on the keyboard at the same time, forcing the user to switch between multiple screens. The results from a previous study informed an ACT-R model of password input on mobile devices. The timing data generated from the model fits the experimental results well. The strategy that the model employs compliments the results from the experiment providing further information into the strategy subjects employed. Validated models of password input on mobile devices are an important tool that can aid designers in usability testing and security professionals when creating new password policies.

Keywords: Passwords; mobile typing; touchscreens; human factors; useable security.

Introduction

Twelve characters long, one number, one uppercase letter, and one special character; password must contain at least two of the following types of characters: letters, numbers, and symbols; these are just two examples of enforced password policies meant to make passwords more secure. Many systems now set a minimum length as well as force users to include non-alphabetic characters in their password. The policies are enforced by the system because they give the passwords high entropy. Passwords are considered high-entropy if they are long and contain a variety of character types. This combination makes them harder to crack using a brute force attack. As the computational power of computers increases, systems are increasingly enforcing password policies that make them high-entropy. These policies vary from system to system and make passwords not only difficult to remember but difficult to input. At the same time users are creating more and more accounts with different systems and must remember an ever-increasing number of passwords (Florencio & Herley, 2007). But there is recognition that the user is an integral part of the security of the overall system (Adams & Sasse, 1999). Being able to test the effects of new password policies prior to enforcing them would be beneficial to security professionals as well as users of the system.

Inputting passwords on mobile devices presents a unique challenge for the users not present on desktop systems. Commonly typed characters are always visible on physical keyboards; on mobile devices characters are on multiple screens that the user must shift between. Navigating between different screens not only increases the number of taps before the character can be input but it also requires the user to remember the keyboard screen, or screen depth, in addition to the location of the character. Now users must recall the password, keep track of a character's position within a password, its spatial location on the keyboard, and its relative screen depth. This becomes even more complicated if the current character is available on multiple screens. Passwords of longer length provide more opportunities for input error. These differences can place significant perceptual-motor and cognitive demands on users. Figure 1 shows the layout of the keyboard at the different screen depths on an iPhone.

Figure 1. The categories of keys based on their screen depth and type on an iPhone

ACT-R (Anderson, 2007) is a multi-domain cognitive architecture for simulating and understanding human cognition and performance. ACT-R's current typing ability is comparable to that of a moderately skilled touch typist. Moderately skilled is defined as typing about 30-40 WPM and knowing the location of the keys without having to look for them but not performing as rapidly as an expert typist (John, 1996). Das and Stuerzlinger (2007) built an ACT-R model simulating expert text input on a cellular telephone with a 12-button telephone keypad using multi-tap as the input method. Multi-tap is an older system of text entry where the user presses a key to cycle through the letters associated with that key; for example pressing "6" twice would produce an "N." With the proliferation of smartphones, this older input method is much less common today as smartphones supply full keyboards as well as other methods for text input. A validated ACT-R model of text input on a smartphone is not available. In making progress toward building a working model of password input on mobile devices, a model of the typing portion of password input was built in isolation from the memory component. Since the models are expandable, as more aspects of the task need to be modeled, the model can change to incorporate errors and the memory component of the task.

Modeling

An ACT-R model was built to model the input speed from the password transcription typing experiments in Gallagher (2014). The task presented subjects with a string similar to a high entropy password. The string was always present at the top of the screen. Subjects were instructed to type the string exactly as it appeared as quickly and accurately as possible. Two of the experimental results heavily influenced the strategy employed by the model. The first result was that as subjects progressed through this task their input speed increased because they spent less time on the page before inputting the target character. The second result was that throughout the experiment subjects did not navigate efficiently between keyboard screen depths. The model was built to provide insight into some of the performance aspects of the task that were not explained by the subject data alone. One question was why did the time spent on the page before symbols were typed remain slower than non-symbols. A second question was how were subjects searching through keyboard screens to find the characters they were less familiar with.

Method

Data Modeled

The model was constructed based on the subjects who interacted with the smartphone using one finger to input text. We took interkey interval (IKI) as our primary dependent measure rather than the more coarse measure of words per minute. This has the advantage of increasing the constraint on the model, since matching only global performance obscures details of how that performance arises. Furthermore, this allowed us to focus on error-free performance, since WPM also includes error correction, which was beyond the scope of this initial inquiry.

There were six character categories: Lowercase, Uppercase, Number, Symbol1, Symbol2, and Symbol3. These categories were determined based on which screen depth characters were on, whether a shift key was required, and their type. The letters were in the categories Lowercase and Uppercase, and were distinguished by case due to the shift key having to be pressed prior to inputting an uppercase letter. The category Number represented the numbers, which are on screen depth 2; Symbol1 represented the symbols that were visible only on the same page as the numbers. The decision to separate these two groups was because we hypothesized that subjects would be more familiar with where the numbers were on the screen and in relation to each other than they would be with the symbols. Symbol2 represented the symbols that were visible only on screen depth three. Symbol3 were the symbols that were visible on both screen depth 2 and 3. Figure 1 shows the location of the key categories.

Materials

Since there is no way to interface ACT-R directly with the iOS simulator, a custom environment was built in Common Lisp for the model to interact with. The model environment mimicked the mobile application used by the subjects. The arrangement, space, and size of the interface elements were the same. Figure 2 shows the two interfaces. The interface elements that are unique to the mobile application, like the touch keyboard and the masking password field, were reconstructed as accurately as possible in the model environment.

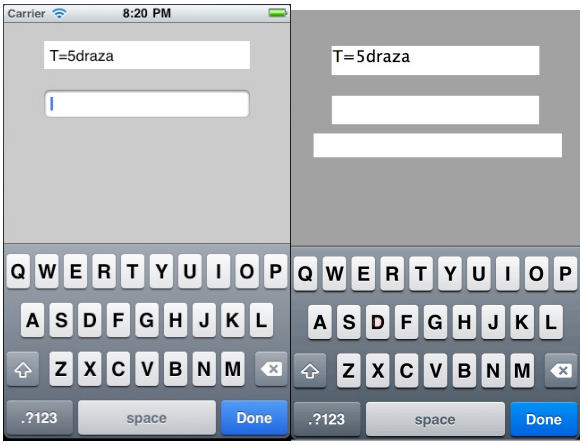


Figure 2. The layout of the iOS application (left) and the simulated Lisp environment (right)

The keyboard was built so that the visible key size was the same as the visible key size in the mobile application, but the functional key size extended beyond that. Since the exact functional key size is not publicly available information, the simulated keyboard evenly split the difference between adjacent keys. Another feature unique to

the mobile application that was reconstructed was the visual change on a key press. When a key is pressed an enlarged version of that key is displayed above the regular position of the key; upon release of the press the enlarged version of the key disappears. In the mobile application, the password field shows the character for a short amount of time after it is typed before masking it with an asterisk. If characters are typed into the password field in rapid succession then the characters are masked as the next key is input even if the normal time before masking has not expired. The password field used in the model environment recreated this behavior and used a time of one second before masking the most recently typed character. The mouse device module in ACT-R was used as a stand-in for a finger in touch interactions.

Design

The model was given a moderate amount of knowledge in declarative memory. For all characters on the keyboard, the model knew if they were letters, numbers, or symbols. The model knew the locations and screen depths of all the letters and numbers. The model only knew the location and screen depth of symbols that were presented to the subjects in the practice blocks of the experiment. For all other symbols, the model did not start with the knowledge of their location or screen depth. The final part of the model's knowledge was which keyboard change key needed to be pressed to navigate between different screens. These assumptions were based on most subjects' familiarity with a QWERTY keyboard, the relationship between numbers, and the experienced gained from doing the practice block.

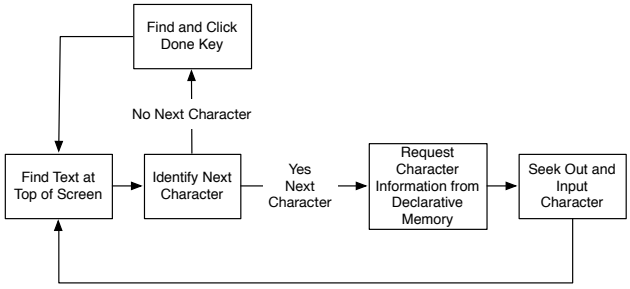


Figure 3. The overall model strategy

The overall strategy that the model employed to input the stimuli was straightforward, a flowchart of it is shown in Figure 3. At the beginning of each trial the model would determine where the stimulus was by picking out the text that was at the top of the screen. The model would then shift visual attention to the text, identify what the first character was, and store it in the goal buffer. The model would then make a recall request to the declarative memory for all available information regarding that character. Either just the type of the character would be recalled or the type of the character, the screen depth, and the location of the key. The model would then seek out and input that character. Once the character was input the model would shift visual attention back to the stimulus and pick out the next character. This process repeated for all the characters in the stimulus. After the last character was input, the model would shift attention back to the stimulus and identify there were no more characters and seek out the done key. To identify the done key the model would both shift visual attention and the mouse in parallel to the keyboard key in the bottom right of the screen, identify it was the done key, and click on it.

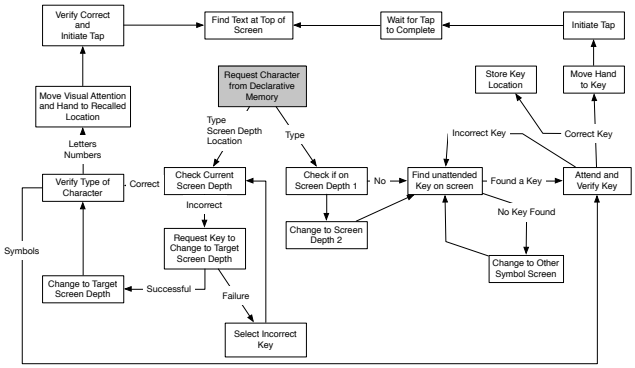


Figure 4. The process the model used to seek out and input a character. The grey box is where the process starts.

The different strategies for seeking out and inputting the characters were based on their category and whether or not the model knew where the key was located. Figure 4 illustrates the process and corresponds to the box "Seek Out and Input Character" in Figure 3. The location and screen depth of the letters and number keys were always recalled correctly. The model took an aggressive approach to inputting letters and numbers by doing as many actions in parallel as possible. After recalling the character, if the model was on the correct screen depth it would move both visual attention and the mouse to the key location in parallel. Once visual attention shifted to the key the model would verify that it was the correct key and as soon as the hand reached the key a press action was initiated. Once the press action was initiated, the model would shift visual attention back to the stimulus to determine the next character. For keyboard change keys as well as the shift key the model would move the mouse and visual attention in parallel. When changing the keyboard the model had to wait for the press action to complete and the keyboard to change before it could shift visual attention. When the target character was a symbol of a known location the model took a more conservative input approach and the process was similar but conducted in a serial manner. If the model was at the correct screen depth it would shift visual attention to the key first, verify it was the correct key and then move the mouse toward the key and initiate the press action after arriving. The model waited until the press action was complete before shifting attention back to the stimulus to determine the next character. If the model was searching for a symbol when the key location was not known it would

start the search from the first page of symbols it encountered and randomly select an unattended key and compare it to the target character. This process continued until the target key was found or there were no more unattended symbols. If there were not more unattended symbols it would switch to the next page of symbols and start searching there. On screen depth two it did not search through the numbers. If the model started on screen depth one it would switch to screen depth two and start the search there. If the model started on screen depth two or three it would start searching on that page and switch to the other one if it was unable to find the key on the starting page. Once the model successfully located the symbol the location and screen depth were added to declarative memory.

To navigate between screens the model had to correctly recall the key necessary to switch between the current screen and target screen. If the model was unable to recall the correct key it would select the incorrect key. The majority of the extra taps between categories subjects made occurred when the previous character was a number or a symbol and the target character was also a number or a symbol. The number of transitional taps to and from letters was closer to the minimum. Due to the difference in number of taps, the chunks represented by those transitions involving letters started with higher base-level activation than the chunks representing the transitions between number and symbols.

There are three processes in the model that vary in the amount of time they take to complete. The first two are related to the recall of chunks from declarative memory. Recall times for this model are based on base-level activation and a random noise component. The chunks representing character keys started with a higher level of activation and recall for them never failed. The amount of time to recall did vary based on the activation decay and the random noise component. For the chunks representing the character transitions the initial activation started lower so that they would not always be recalled. The time to navigate between screens was influenced by recall time as well as success or failure of recall. The third source of variation is the visual search time when a symbol's screen and location are not known. The preceding character determines the page the model is on before it starts starting searching for a symbol and the order it examines keys is random, the amount of time it takes to find a symbol varies for each run.

To approximate touch screen interaction we used the mouse in place of the finger, similar to Salvucci, Taatgen, and Kushleyeva (2006).

Results

To be able to compare the subject data and the model data, the procedure from Byrne (2013) was used determine the number of times the model needed to be run to build a 95% confidence interval within 5% of the mean interkey intervals. The model was run 20 times to estimate the coefficient of variation for each of the interkey interval. The largest coefficient of variation was used in the computation, and the minimum number of model runs was determined to be 93. One hundred model runs were performed.

The interkey interval for each pair of categories for the model and the subjects can be seen in Figure 5 and Figure 6. The matched pairs of the model and subject data are shown in Figure 7. The results of a simple linear regression indicated that the model was able to predict 88% of the variance (R^2 = .88, subject = 0.69 * model + 0.58). The mean absolute deviation of the model from the subject data was 260 ms or 15.7%. Figure 8 and Figure 9 show the average number of taps the model and the subjects made transitioning between character categories. A simple linear regression indicated that the model was able to predict 90% of the variance (R^2 = .90, subject = 0.95 * model - 0.15). The mean absolute deviation of the model from the subject data was 0.22 taps.

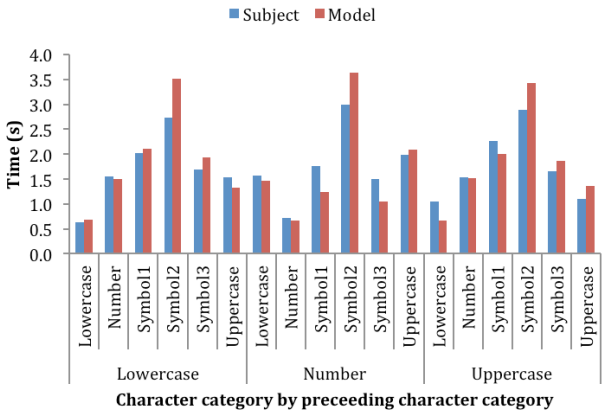


Figure 5. IKI for transitions from Lowercase, Number, and Uppercase

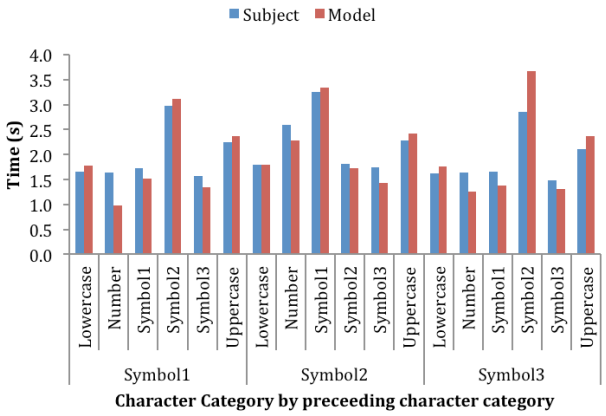


Figure 6. IKI for transitions from Symbol1, Symbol2, and Symbol3

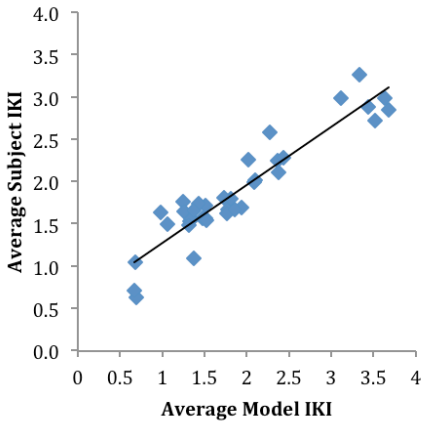


Figure 7. Matched Pair IKI between the subjects and the model

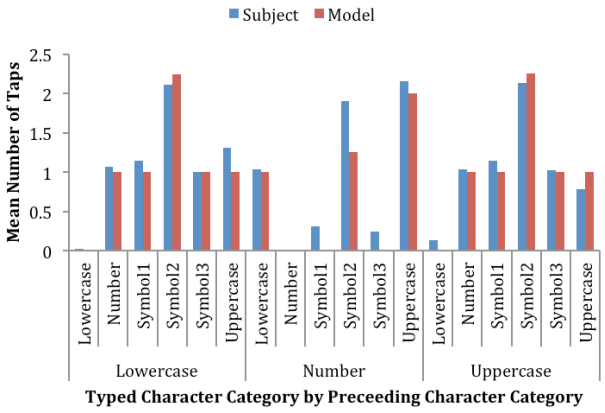


Figure 8. Average number of taps between character categories

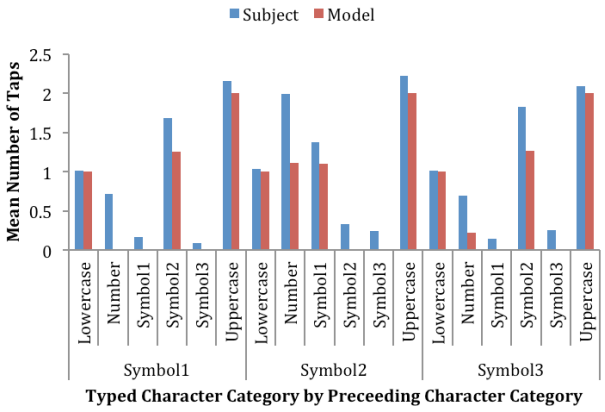


Figure 9. Average number of taps between character categories

Discussion

The strategy that the model uses highlights the three main reasons why inputting high-entropy passwords on mobile devices is slow. The first is different strategies are employed for different character categories. The model and subjects are able to efficiently input characters they are familiar with but take a more conservative approach when inputting symbols. The second reason is that the model does not have knowledge of where the symbols are and has to locate the symbols the first time they are input. The third reason is not being able to navigate efficiently across screen depths. Like the subjects, the model does not navigate efficiently across pages and will not always take the shortest path to the correct page.

These deficiencies in task performance highlight ways that passwords could be structured to improve task performance. Concerning which symbols are selected, using ones on the first page reduces the amount of time spent searching through keyboard screens. Rather than using symbols that subjects are unfamiliar with, symbols that are input frequently could be used so that subjects are more likely to remember their location on the screen. While using high-frequency symbols subjects may eventually employ a less conservative approach when inputting the symbols as they grow more accustomed to it. With regard to inefficient navigation, not using characters from the different symbol screens in sequence would be beneficial because it would eliminate the need to navigate across screens. If the user could modify the keyboard layout, they could place the symbols they would like to use more frequently on the first screen. This would aid them in search time, as they know which symbols they use most and could give them priority. These recommendations for structuring passwords and keyboard designs could be tested with the model to determine if there are performance benefits.

For example, take the passwords *Af_3+2=5_Fa!* and *aAfF235__+=!*. They are both twelve characters long, comprised of the same characters, and meet the requirements of having at least two characters of each type. The first one was created using recommendations for memorability, *A*ddition *f*act _ *3+2=5* _ *F*act *a*ddition*!*. The second one rearranged the characters so that they were in order of screen depth. To predict input time without the variability introduced by visual search the model was modified so that it knew the locations of all the symbols. With this modification the model predicts an input time of 24.57s in landscape and 24.12s in portrait for the first version and 15.99s in landscape and 15.59s in portrait. For the first password the model inefficiently navigated on two transitions but never inefficiently navigated on the second version. Arranging the characters in an order that allows them to be input in the most efficient manner gives a savings of almost 10 seconds.

Future Modeling Directions

There are a number of steps that could be taken to improve the model's performance in the remaining categories. When typing a Number the model is faster when coming from Symbol1 and Symbol3 than the subjects are. One of the reasons that this happens is because the model never misidentifies the page that it is on while subjects do and will navigate off of screen depth two even when they do not have to. When typing a character in Symbol2 the model is slower than subjects for all preceding character categories except Symbol2. One of the reasons that this could be happening is the model is using completely random visual search. There are six symbols that are part of the category Symbol2 that are pairs, [ ] { } and < >. If a subject knows the location and screen depth of one of members of a pair, they can use that information to more quickly find the other member of that pair instead of blindly searching for it. Additionally if subjects are searching and find one half of the pair they can use this as a cue to find the other half where the model just continues searching randomly.

In advancing the model forward the first major change to make would be to have the model make errors when inputting the stimuli. This can likely be accomplished by turning on motor movement noise in ACT-R. With noise on the model would not always successfully acquire the target key and would also not always land in the center of the key. This would introduce the most common kind of errors, adjacent key errors. The subject data indicates that while not all errors are caught and corrected the majority of them are. Therefore the model's strategy would need to change so that sometimes it would verify the input. After inducing the model to make errors, the next step in development would be to branch out to the other input styles and devices. Additionally, there may be variations in subject strategy. Subjects may keep more than one character in working memory at a time instead of referring to the stimuli after each character is typed.

Additionally this model only used an iPhone. Further research needs to be done to be able to generalize to other iOS devices and platforms, e.g., Android and Windows Phone, and the variety of devices they run on. Using Android phones. Now that the size of iPhones has increased and alternative keyboards are available for iOS, these new features can be tested to see if they provide any advantage when typing passwords.

Conclusions and General Discussion

Although mobile device keyboards were designed to be similar to physical keyboards, they are not the same and many of the limitations of the mobile device make typing passwords slower. One of the main factors in the slow input speed is inputting the symbols. While the model learns the location of the symbols the initial act of searching through the keyboards screens is time consuming. Compounding the slow down is the conservative approach taken during the typing process to ensure accuracy. Since the model does not navigate between the symbol pages and number pages efficiently it could be helpful for passwords to not require symbols and numbers in sequence. Having a working model of password input on mobile devices has a number of benefits. When presented with novel passwords, the model can make predictions of password typing time. This is especially useful because as new password policies are generated they can be tested to see if they are detrimental or beneficial to improving input speed. In addition to different password policies, different keyboard designs can be tested prior to implementation. Typing the password is only one component of authentication and needs to be incorporated with work that looks at the cognitive component as well.

Acknowledgments

This research was supported by the National Institute of Standards and Technology under grant #2012-NIST-MSE-01. The views and conclusions contained herein are those of the authors and should not be interpreted as representing the official policies or endorsements, either expressed or implied, of NIST, the U.S. Government, or any other organization. The authors would like to thank Clayton Stanley for his help with the Lisp programming.

References

Adams, A., & Sasse, M. A. (1999). Users are not the enemy. *Communications of the ACM*, 42(12), 40-46.

Anderson, J. R. (2007). *How can the human mind occur in the physical universe?* New York: Oxford University Press.

Byrne, M. D. (2013). How Many Times Should a Stochastic Model Be Run? An Approach Based on Confidence Intervals. *In The 12th International Conference on Cognitive Modeling*, 445-450.

Das, A., & Stuerzlinger, W. (2007). A cognitive simulation model for novice text entry on cell phone keypads. *Proceedings of the 14th European conference on Cognitive ergonomics: invent! explore!*, 141-147.

Florencio, D, & Herley, C. (2007). A large-scale study of web password habits. *Proceedings of the 16th international conference on World Wide Web*, 657-666.

Gallagher, M. A. (2015) Modeling Password Entry on Mobile Devices: Please Check Your Password and Try Again, Doctoral Dissertation, Rice University, Houston TX.

John, B. E. (1996). TYPIST: A theory of performance in skilled typing. *Human-computer interaction*, 11(4), 321-355.

Salvucci, D. D., Taatgen, N. A., & Kushleyeva, Y. (2006). Learning when to switch tasks in a dynamic multitasking environment. *Paper presented at the Proceedings of the seventh international conference on cognitive modeling,* 268 –273.

Fast-Time User Simulation for Dynamic HTML-based Interfaces

Marc Halbrügge (marc.halbruegge@tu-berlin.de)
Quality & Usability Lab, Telekom Innovation Laboratories
Technische Universität Berlin, Ernst-Reuter-Platz 7, 10587 Berlin

Keywords: Cognitive Modeling; Modeling Tools; ACT-R; HTML; POMDP.

Introduction

Using cognitive modeling to predict user behavior in the human-computer interaction domain is a promising field, but often hindered by practical problems. Especially the creation of a mock-up of the technical system under evaluation is often a tedious and time-consuming task. For the growing number of HTML-based applications, the modeling toolbox ACT-CV (Halbrügge, 2013) provides direct access to the user interface, removing the necessity to create a mock-up before the actual modeling can start. A down-side of this approach is that unaltered applications often cannot be used for fast-time simulation. This paper presents a new tool that solves this problem by capturing UI states and also the *control flow* of HTML applications and by transforming both into finite automata that can be used for fast-time simulation.

Related Work

The amount of work that needs to go into the creation of a link between a user interface (UI) and a cognitive architecture should not be underestimated, a fact that has been well put by Kieras and Santoro (2004, page 102): "Programming the simulated device is the actual practical bottleneck". There are solutions to this, but they often require instrumented applications (e.g., Urbas et al., 2006; Büttner, 2010). Other approaches like SegMan (Ritter, Van Rooy, St. Amant, & Simpson, 2006) allow unaltered applications, but work on a pixel-by-pixel basis, forcing the modeler to re-create the symbolic information from its lowest-level graphical equivalent.

ACT-CV

The cognitive modeling toolbox ACT-CV has started as a visual device for ACT-R (Anderson et al., 2004) that uses computer vision (hence ACT-*CV*) for the creation of symbolic representations of a visual scene from a video camera or a computer screen capture. HTML support was added in version 2 (Halbrügge, 2013), building ACT-R's visicon (visual icon) directly from the textual and clickable elements in the browser window and applying ACT-R's mouse clicks to them.

While direct access to real-world user interfaces eliminates the tedious need to create mock-ups, some inconveniences remain: Fast-time simulation is impossible, especially in the presence of graphical transitions and animations that take fixed amounts of time. It is also very hard to parallelize the model runs during batch processing as every model would need at least its own browser instance. In the case of stateful user interfaces, extra checks would be necessary to ensure that the model sessions do not interfere with each other.

Creating Finite Automata from Dynamic Web Pages

When we bring to our mind that ACT-R's visual module only uses the geometry, color, and (textual) content of a screen element, it should be obvious that creating and holding a complete browser instance to create this small amount of information is highly inefficient. How can we improve this?

The approach taken in this paper is to keep a history of the observed visicons in their reduced form, and taken together with the actions taken by the model, using this history to create a computational representation of the system that completely replaces the original browser content. As a cognitive model can only "see" through ACT-R's visicon, the removal of the actual browser is transparent to it.

Formalization The state $s \in S$ of the user interface is the set of currently visible elements as represented in ACT-R's visicon. The visicon is a table of chunks of type visual-object that contain their positions, dimensions, colors, and symbolic values. For reasons of simplicity, we consider every visicon entry as a possible action $a \in A$. An action is executed by interacting (i.e., clicking or tapping) with its corresponding UI element and usually leads to a new UI state. We are assuming discrete time, i.e., every transition to a new state denotes a new time step.

How can we represent the UI logic? If the UI had Markov properties, i.e., no hidden states, it could be captured by a deterministic transition function $\delta(S,A) \mapsto S$. This is usually not the case, though. Stateful interfaces are the rule rather than the exception; examples on the web are shopping carts, stored billing information, or personalized news and ads with the help of browser cookies.

In order to support hidden states, ACT-CV uses a deterministic finite partially observable Markov decision process (POMDP) representation for the UI logic. If a pair of state and action does not lead deterministically to another state, we go back in history until we can establish a deterministic transition function again, e.g., $\delta(S,A,S,A) \mapsto S$. Implementation-wise, we only store the part of the history that is needed to reproduce the observed behavior of the UI.[1] During the learning phase, the complete history needs to be kept in order to be able to adapt to newly discovered hidden states.

Exploration of the User Interface Finding an optimal solution to a POMDP problem is known to be NP-hard (i.e., not

[1] The flexible history approach presented here is not always optimal, e.g., because states with self-transitions can fill up the history without adding any information. A better, but more complex solution would be Looping Suffix Trees (Holmes & Isbell Jr, 2006).

easily computable for non-trivial cases, Kaelbling, Littman, & Moore, 1996). Fortunately, we are not in need of optimal solutions, often not even complete ones. The part of the state space that is actually visited by the cognitive model under investigation is most of the time much smaller than the complete state space of the application.

If the cognitive model is deterministic, we can build the UI representation from the observation history of a single complete model run. Non-deterministic models may visit much more states than deterministic ones. Due to this, we need to explore the application beforehand and independently of the model. ACT-CV procides a simple exploration mechanism that is inspired by the RMax algorithm (Brafman & Tennenholtz, 2003) for this situation.

Application Example and Results ACT-CV has been used during a modeling attempt targeted at a HTML-based home assistance system (Halbrügge & Engelbrecht, 2014). The work comprised an analysis of sensitivity towards global ACT-R parameters that needed hundreds of model runs. If this analysis had to be done in real-time, it would have needed several weeks to complete. With the help of ACT-CV and the state machine approach introduced above, the analysis could be run approximately 150 times faster on a commonplace desktop computer and finished within a few hours.

Open Issues While the automaton approach has worked very well in the example given above, issues remain. First, graphical transitions like fade-ins make it hard to determine when human users can actually see and act upon different elements on the screen. As ACT-CV does not capture the fade-in, some timing information may get lost. Secondly, page scrolling behavior has not yet been modeled using the automaton approach. Scrolling can lead to many different UI states and may need extra treatment. And finally, if the UI allows data entry, e.g., in text fields, this instantly blows up the amount of possible states of the application. Free exploration cannot be used in this case.

Conclusion

Especially when using cognitive modeling to capture rare events like errors, it is often necessary to execute many model runs. In this case, it is very important to be able to run the model in fast-time simulation. In the case of HTML-based interfaces, this is only possible if a non-HTML mock-up is available. While some modeling tools (e.g., CogTool, John, Prevas, Salvucci, & Koedinger, 2004) have import functions for static HTML content, dynamic applications were not yet covered. ACT-CV closes this gap by allowing to extract the information on the screen that is needed by ACT-R's visual module from the rendered browser content. The application logic is transformed into a finite state machine by the means of guided or free exploration using a POMDP approach.

ACT-CV is freely available for download at http://act-cv.sourceforge.net.

Acknowledgements

The author gratefully acknowledges financial support from the German Research Foundation (DFG) for project "Automatische Usability-Evaluierung modellbasierter Interaktionssysteme für Ambient Assisted Living" (AL-561/13-1).

References

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological review*, *111*(4), 1036–1060.

Brafman, R. I., & Tennenholtz, M. (2003). R-MAX: A general polynomial time algorithm for near-optimal reinforcement learning. *The Journal of Machine Learning Research*, *3*, 213–231.

Büttner, P. (2010). Hello Java! Linking ACT-R 6 with a Java simulation. In D. D. Salvucci & G. Gunzelmann (Eds.), *Proceedings of the 10th international conference on cognitive modeling* (pp. 289–290).

Halbrügge, M. (2013). ACT-CV: Bridging the gap between cognitive models and the outer world. In E. Brandenburg, L. Doria, A. Gross, T. Günzlera, & H. Smieszek (Eds.), *Grundlagen und anwendungen der mensch-maschine-interaktion* (pp. 205–210). Berlin: Universitätsverlag der TU Berlin.

Halbrügge, M., & Engelbrecht, K.-P. (2014). An activation-based model of execution delays of specific task steps. *Cognitive Processing*, *15*, S107-S110.

Holmes, M. P., & Isbell Jr, C. L. (2006). Looping suffix tree-based inference of partially observable hidden state. In *Proceedings of the 23rd international conference on machine learning* (pp. 409–416).

John, B. E., Prevas, K., Salvucci, D. D., & Koedinger, K. (2004). Predictive human performance modeling made easy. In *CHI '04: Proceedings of the SIGCHI conference on human factors in computing systems* (pp. 455–462). New York, USA: ACM Press.

Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, *4*, 237–285.

Kieras, D. E., & Santoro, T. P. (2004). Computational GOMS modeling of a complex team task: Lessons learned. In *Proceedings of the SIGCHI conference on human factors in computing systems* (pp. 97–104).

Ritter, F. E., Van Rooy, D., St. Amant, R., & Simpson, K. (2006). Providing user models direct access to interfaces: an exploratory study of a simple interface with implications for HRI and HCI. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, *36*(3), 592–601.

Urbas, L., Heinath, M., Trösterer, S., Pape, N., Dzaack, J., Kiefer, J., & Leuchter, S. (2006). AGImap: A tool-chain to support the modeling of the interaction level of dynamic systems. In *Proceedings of the 7th international conference on cognitive modeling* (p. 409). Trieste: Edizioni Goliardiche.

Cognitive Modelling for the Prediction of energy-relevant Human Interaction with Buildings

Jörn von Grabe (Joern.vGrabe@uni.li)
Institute for Architecture and Planning, University of Liechtenstein

Keywords: buildings; energy consumption; human-building-interaction; prediction

Introduction

Building occupants interact with various elements of a building in order to satisfy their diverse needs – such as thermal comfort, privacy, task implementation, etc. As a result they significantly influence the energy balance of a building. For example, operating windows has a major influence on heating demand; operating lighting has a major influence on electricity demand, etc. If energy saving is an aim, this human interaction with buildings must not be ignored.

In building sciences building simulation systems are very often used in order to predict and optimize the energy balance of newly planned buildings. Based on local weather data these codes calculate the dynamic energy flows in and around the building caused by solar radiation, temperature, and humidity differences and convective energy transport, etc. The algorithms are well established and validated.

In an attempt to reproduce human interactive behavior in buildings, various algorithms have been developed in recent years that can be used in building simulations. Usually, these algorithms are rather simple stochastic models – such as probit models – that link a (limited set of) predictor(s) – such as solar radiation – to the execution of a particular action – such as closing the sun screen. Due to their simplicity they fail to capture both, the multifaceted character of the context of interaction as well as the complex cognitive processes underlying interaction.

Research aims

A larger research project has been started in the past in order to improve the predictability of energy- relevant human interaction with buildings by means of analyzing the involved psychological processes and by developing an according cognitive model (Grabe, 2013, 2014b). This research project can roughly be subdivided into three main parts (Grabe, 2014a): The first part aimed for the systematic identification of the contextual factors and their interrelations that are relevant for this type of interaction and developed a heuristic method to achieve this goal (Grabe, 2015). An example for a typical contextual factor is the structural and mechanical characteristics of the element the occupant wishes to interact with and its reachability (e.g. the window). These factors determine the ease of use of the element and thus (part of) the costs of interaction.

To be useful for prediction, these somewhat qualitative relationships need to be transferred – in a second step – into scientific conceptualizations of pertinent disciplines. This has already partly been done by identifying a vast number of theories from psychology and social sciences that theoretically conceptualize the above mentioned qualitative relationships and that have potential to be useful for a predictive model. Among those, cognitive architectures – such as ACT-R (Anderson, 2007) – play an important role.

Finally, the third step will aim at the integration of these theories into building simulation systems. This requires a specific adaptation to the syntax and semantics of building simulation systems and the transformation into software code for the quantitative prediction of behavior.

Cognitive modelling fields

A number of psychological processes that take place prior, during and after interaction seem to be particularly suited to be modelled by the principles of cognitive architectures. An example will illustrate this point.

During interaction, a building occupant is confronted with a basic type of decision: The operation of which element of the building is best suited to satisfy his or her actual need given the actual context? Imagine a person that is feeling too warm and sets the goal to feel cooler. In principal, there is a multitude of action options to choose from: opening the window, switching off the heating, removing part of the clothing, closing the sun screen, switching on the cooling, etc. However, these action options are not equally suited to satisfy the actual need in the given context. To make a decision, further contextual information must be received directly from the environment via some sensory system (e.g. the fact that the sun is currently not shining) or must be retrieved from declarative memory (e.g. the external temperature as experienced before entering the building, or the fact that the heating is currently not switched on). Further on, each action is associated with a certain probability to be successful in satisfying the actual need and there are also costs attached to the execution of each action. Both, probability of success and costs are not easily defined. A measure of success might include how fast a temperature drop can be achieved and how sustainable the new conditions are. Costs might include diverse aspects like the physical effort to move to the window or in how far the (anticipated) new conditions will interfere with the satisfaction of other needs (for example, opening the window might increase noise level and interfere with task implementation). Moreover, satisfaction of a need might be

more or less urgent (e.g. feeling hot instead of simply warm increases urgency). This urgency can be expressed as the value of achieving the particular goal.

From the perspective of cognitive architectures, it is plausible to regard these different action options as a set of potentially adequate production rules to reach the goal. Each production rule has its own specific utility, given a particular context. Determining factors such as probability of success or costs are learned during interaction with a specific building in a specific context and utilities can be established.

Relevant specificities of building simulations

Behavior cannot be analyzed or predicted without considering the physical environment in which it takes place. Building simulation systems are well suited to simulate essential parts of this environment and can thus represent the required counterpart to the simulation of particular types of behavior. However, some specificity has to be taken into account. A main example of such specificity is the simulation time span and the time discretization in building simulations. Since we are interested in the building's performance during the whole year, simulations usually comprise 8760 hours. The involved thermo-dynamic processes usually show such a low dynamic that a time discretization below 30 minutes is seldom reasonably; and even with such a discretization the amount of produced data is enormous. This is not well in agreement with the dynamic of cognitive processes which usually requires a far finer time discretization.

References

Anderson, J.R. (2007). *How can the human mind occur in the physical universe?* : Oxford University Press.

Grabe, J.v. (2013). *Ein heuristisches Verfahren, das es ermöglicht, die prognostische Simulation menschlichen Interagierens mit Gebäuden realitätsgerechter zu gestalten - Prototypische Entwicklung am Beispiel des Energiehaushalts von Gebäuden.* (doctoral thesis), University of Technology, München. Retrieved from http://mediatum.ub.tum.de/pnode?id=1144961&key=VcmG2bRMhShJqeNN

Grabe, J.v. (2014a). The context of the energy-relevant human interaction with buildings. *International Journal of Human Factors Modelling and Simulation, 4*(2), 102-120. doi: 10.1504/IJHFMS.2014.062382

Grabe, J.v. (2014b). Mensch-Gebäude-Interaktion im Kontext des energierelevanten Handelns von Gebäudenutzern. *Bauphysik, 36*(4), 200-208. doi: 10.1002/bapi.201410028

Grabe, J.v. (2015). A heuristic method for the systematic identification and classification of the context of energy-relevant human interaction with buildings. *unpublished article, submitted to Indoor and Built Environment*.

Visual Search of Displays of Many Objects: Modeling Detailed Eye Movement Effects with Improved EPIC

David E. Kieras (kieras@umich.edu)
Electrical Engineering & Computer Science Department, University of Michigan
2260 Hayward Street, Ann Arbor MI 48109-2121, USA

Anthony Hornof (hornof@cs.uoregon.edu)
Department of Computer and Information Science, University of Oregon
1202 University of Oregon, Eugene, Oregon 97403 USA

Yunfeng Zhang (zywind@cs.uoregon.edu)
Department of Computer and Information Science, University of Oregon
1202 University of Oregon, Eugene, Oregon 97403 USA

Abstract

This paper describes a new set of results on visual search of displays of 75 objects that differ in size, shape, and color, and presents a cognitive architecture model based on the active vision concept that accounts for the effects using object eccentricity and size effects, noisy saccades, and fixation memory provided by a persistent visual store. The data confirm older, less complete studies of this task. The model is a significant refinement of earlier visual search models and preliminary fits show that it promises to provide an integrated architectural account of these effects.

Keywords: cognitive architecture, visual search; cognitive modeling; eye movements

Introduction

Many everyday and work activities involve visual search, the process of visually scanning or inspecting the environment to locate an object of interest that will then be the target of further activity. An especially tractable form of visual search takes place in many human-computer interaction tasks in which a particular icon coded by color, shape, and other attributes must be located on a screen and then clicked on using a mouse. Such visual search takes place in a visual environment that is much simpler than natural scenes, and so is a both a good theoretical and practical domain to model visual search processes. It combines relative simplicity of the visual characteristics of the searched-for objects with practical relevance. The task is a natural one in the sense that such activities are very common in current technology; an example is current radar displays in military applications, which can contain a large number of icons and other objects (cf. Kieras & Marshall, 2006). Thus understanding in detail how visual search works in such domains can lead to better system designs.

Kieras (2010) presented a model for the results of a classic study by Williams (1967), who using early film-based eye tracking methodology, explored the visual search of large and dense displays of many items that can be searched by multiple attributes. He manipulated the size of the objects along with their color and shape, an unusual combination in the visual search literature. Kieras and Hornof (2014) showed how the model could be used in a simpler form applicable to interface design problems. But some key issues in the model could not be addressed because Williams reported only a small subset of the potential data, and essentially no characteristics of the eye movements themselves.

What's new. New eye movement data was collected in a Williams-like task that includes the complete eye movement trajectory and precise search completion times. This allows analysis of additional effects, such as those of object size, saccade distance, the characteristics of fixated objects that do not match the search cues, refixation effects, and so forth - far beyond what is possible with the Williams (1967) data.

The EPIC architecture was improved in two significant ways: First, the acuity functions that describe whether an object property can be detected as a function of object eccentricity and size were given the same form as psychophysical functions resulting from an especially relevant class of experiment. Second, EPIC's eye movement mechanism has been completely accurate - if the cognitive processor issued an instruction to fixate a certain object, the eyes always moved exactly to that object. However, there is abundant literature that eye movements to a target normally fall short and have variability linear with the distance. EPIC models would thus be more efficient than humans, meaning that to match human data, other parameters might have to be distorted from their realistic values. Thus, EPIC's oculomotor processor now makes "noisy" eye movements. To compensate, the visual search strategy has to be adapted to complete the task in spite of the unreliability of fixations.

Thus despite the superficial similarity of this work to the earlier, there are new challenges in the modeling. The work reported here is preliminary - there is much new ground to explore. First the experiment will be presented, followed by the architectural changes and the current modeling results.

The Visual Search Experiment

The task was to locate a target object in a field of seventy-five distractor objects. Each object on the display had a unique two-digit number and a unique combination of color, size, and shape. Participants were precued with the number of the target, and some combination of the target's color, size, and shape.

Twenty-four participants were recruited from the

University of Oregon campus community. Two were excluded because the eye tracker could not be calibrated to them. All had normal or corrected-to-normal vision. Participants received a base payment of $10 plus a bonus (ranging from $5 to $8) based on their speed and accuracy.

Search fields were presented on the central 1600 x 1200 portion of a color-calibrated Dell 2407WFP 24-inch monitor connected to a 3.06 GHz Intel Core 2 Duo Macbook Pro running Mac OS 10.8. The data collection software was written in C++, Objective-C, and Cocoa. Eye movement data were collected using a binocular 120 Hz LC Technologies Eyegaze tracker after a nine-point calibration. The monitor was positioned 60 cm from the participant.

Each participant was presented with ninety-six search fields, each with seventy-five randomly arranged objects. Figure 1 shows one of the search fields. Each search field was preceded by the presentation of a precue that described the target in text and included the target's two-digit number and, depending on the condition, some combination of the the target's color, size, and shape. Because each precue could include any combination of the three features, including none, there were a total of eight possible precue types. Each combination was used in twelve trials, resulting in the ninety-six trials per subject.

Search fields contained seventy-five objects on a 67% gray background that subtended 39° by 30° of visual angle. Each object had a unique combination of color, size, and shape. Colors were blue, green, yellow, red, and purple. Sizes were small (0.8º), medium (1.6º), and large (2.8º), measured as the diameter of the circular object of that size, with other shapes normalized to the same area. Shapes were circles, semi-circles, squares, equilateral triangles, and crosses. Each object had a one-pixel black border.

The seventy-five unique objects were randomly distributed across the search field with at least one degree of visual angle between adjacent objects. A unique two-digit number from 01 to 75 appeared in the center of each object with a height of 0.26° (10 pixels). The precue appeared in the center of the display in the same typeface, with each feature listed on a separate line. Participants started each trial by clicking on an XX above the target description.

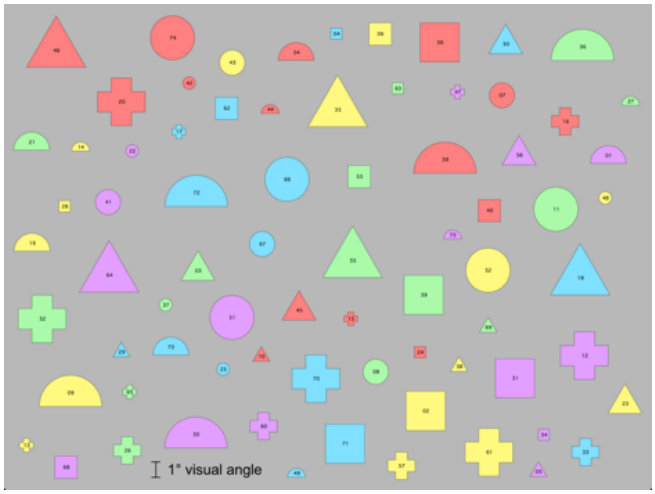


Figure 1. A sample search field used in the experiment.

Each successful trial proceeded as follows: (1) The precue appeared in the center of the display. (2) The participant moved the mouse and clicked on the XX. (3) The precue disappeared and the search field appeared. (4) The participant found the target. (5) The participant moved the mouse and clicked on the number in the target.

Participants were constrained to not move the mouse until they found the target by using a point-completion deadline (Hornof, 2001). Participants practiced until they were comfortable with the deadline.

Participants were rewarded for successful trials with a pleasant 170 ms "cha-ching" sound and penalized for error trials with a 350 ms buzzer. Participants were also financially rewarded. Each trial started with a bonus of five, twelve, or twenty-one cents, depending on the difficulty of the condition (for example, color was easiest) and the bonus diminished at a rate of 0.4, 0.3, or 0.15 cents per second until the participant clicked on the target (stopping at zero, and with faster rates for easier conditions). Errors resulted in no bonus plus a five-cent penalty. Accumulated bonuses were reported to the participants every twenty-four blocks.

Results

The fixations were identified using a dispersion-based algorithm with a maximum dispersion window size of 0.7º and a minimum fixation duration of 60 ms. The error in the eye tracking data was reduced using the method of required fixations, as described in Zhang & Hornof (2014), yielding a series of fixations for each trial by each subject, for a total of about 64 thousand fixations.

In each trial first, last, and any offscreen (and subsequent) fixations were discarded. Then the apparent target of each fixation was designated as the object on the display whose center was closest to the point of fixation; these were considered to be the fixated objects. Then the proportion of fixations in which the properties of the fixated object matched the cue properties were calculated. Similar calculations were made for other statistics, such as the *saccade distance* - the difference between the current and previous fixation. These statistics were accumulated for each subject in each condition, and means computed for each condition.

In all of the graphs shown here, the observed mean values are plotted with solid bars, and the predicted with open bars. Observed values are shown with 95% confidence intervals for the mean based on the values averaged over subjects.

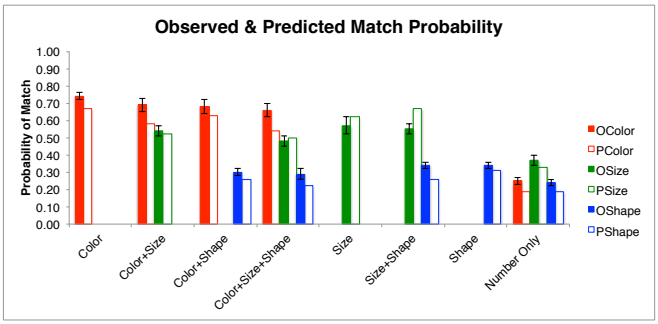


Figure 2. Proportion of fixations that match the cued properties.

The observed values will be discussed first; the predicted later in the context of the model presentation.

First, the results were consistent with those reported by Williams(1967). Figure 2 shows the proportion of fixations on objects that matched the cued properties. E.g., if the color was the only specified cue, about 74% of the fixations were on objects with the specified color. The color cue produces the highest proportion of matches, followed by object size, whereas object shape produces the lowest proportion of matches. The Number Only condition is shown for comparison; here a "match" just corresponds to whether the fixated object has the same property as the target object; the fact that the proportion of matches corresponds to their distribution in the display (five colors and shapes, three sizes) means that these fixations were basically random with regard to the color, size, or shape of the object.

These results replicate the Williams results quite well, showing that color is the most effective cue in guiding visual search, and shape is the least. But size appears to be more effective in these data compared to Williams, being similar to color, perhaps because there were only three different sizes, rather than four as in Williams that may have been difficult to discriminate.

To further compare with Williams (1967), Figure 3 shows the number of fixations required to complete the task for each cue type. The color cue requires the fewest fixations, followed by size, then shape, with the Number Only cue requiring the most. These effects also basically replicate the Williams results, but are more precise due to better eye tracking methodology.

A new effect in these data concerns the saccade distance. If a cue is more effective than another in guiding visual search, the corresponding property of an object should be visible at a greater eccentricity, meaning that saccades should be longer on the average for more effective cues. Figure 4 shows this effect; color cues produce the longest saccades, followed by size, then shape, then Number Only. The effect is fairly small, but reliable, as shown by the overlap relations of the confidence intervals. The small size of the effect could be due to averaging over both matching and mismatching objects in the effective cue conditions.

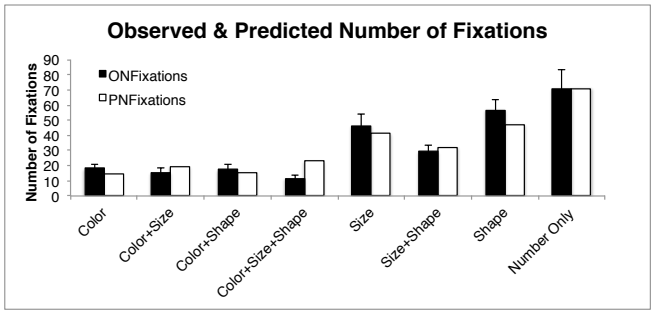


Figure 3. Mean number of fixations required to complete a trial.

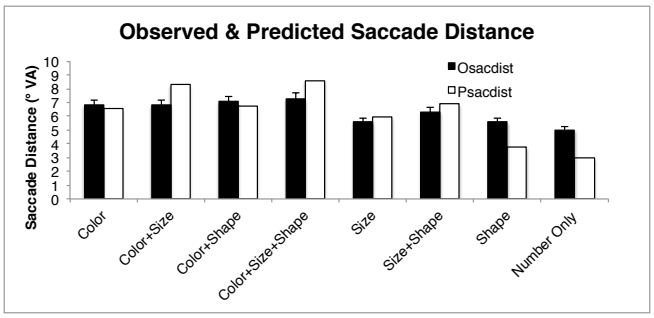


Figure 4. Mean saccade distance from the previous fixation to the fixated object.

Discussion

Visual Search and Active Vision

The empirical literature on visual search was dominated for a long time by studies that ruled out eye movements. But tasks in which the eye is free to move about a static display is more representative of the normal operation of the visual system and the role of attention in visual activity. This point was argued eloquently by Findlay & Gilchrist (2003) in presenting an *active vision* framework for understanding visual activity.

In active vision, a key process is choosing the next object for inspection. A variety of studies (see Findlay & Gilchrist, 2003, for a review) have shown that properties such as the color, shape, size, or orientation influences which object is chosen for the next fixation; the phenomenon is called *visual guidance*. These properties are available to some extent in extra-foveal or peripheral vision, meaning that visual attention, which is almost synonymous with where the eye is fixated, involves using extra-foveal information to select for detailed examination one of the objects currently perceived in the visual scene.

The importance of color in visual search is consistent with many results ranging from classic human factors studies (e.g. Sanders & McCormick, 1987). But in the active vision framework, color is not specially privileged in some way, but rather, various direct measurements show that the color of an object is visible over a wide range of eccentricity and object sizes (e.g. Gordon & Abramov, 1977), and so can often serve as an effective cue about where to look next. The relative ineffectiveness of shape is likewise not due to a fundamental problem with shape, but rather that in many cases, recognizing the shape requires resolving detailed features that can only be seen close to the fovea. As an extreme of shape recognition, recognizing the text label can require foveal vision.

Saccade accuracy

A general property of these data was that there are many fixations that appear to be between objects. The average distance between the fixation point and the center of the closest object averaged 0.99° and was similar across cue conditions. This inaccuracy could have three causes: (1) a deliberate strategy to collect information over a wider area than a precise fixation would allow, which seems unlikely given the density of the display; (2) measurement error, even after applying our error correction technique; (3) error and noise in the oculomotor system – saccades tend to fall short of the intended target and display some variability as well. In these data, there is no strong tendency for in-between fixations to be followed by a short corrective

saccade to precisely fixate an object. Rather, it seems likely that depending on the extent of extra-foveal vision, the eye can collect enough information at the "missed" fixation point that a person can usually correctly decide if an object in the vicinity is the target, and fixate another object if not.

Repeat fixations and memory failures

One overall feature of these results is that many more fixations are required than should be necessary if each object only received one fixation; for example, it should require no more than 37.5 fixations on average in the Number-Only condition to find the labeled object, but about 71 are performed. The data shows that 33% of the fixations are on a previously fixated object in this condition; in contrast, the four color-cue conditions have a lower repeat rate of 21%.

In contrast, some observations and modeling of repeat fixations (Peterson et al. 2001, Kieras & Marshall, 2006, Kieras, 2009, 2010) suggests that repeat fixations are relatively rare, around 5%, implying a good memory for previous fixations, and almost all are performed immediately, being due to recognition (encoding) failures rather than failures of the memory for previous fixations.

However, in these data, immediate repeats average about 14% with little variation between cue conditions, and the average lag between repeat fixations is about 2.4 in the color cue conditions, and much higher at 12.5 in the Number-Only condition (lag = 0 is an immediate refixation). So perhaps repeat fixations in these data are due to both factors: there are some encoding failures leading to immediate or almost immediate fixations, and some memory failures, especially in the conditions that take much longer.

The EPIC Cognitive Architecture

The EPIC architecture for human cognition and performance directly supports an active vision approach to visual search and provides a general framework for simulating a human interacting with an environment to accomplish a task. The reader is referred to Meyer & Kieras (1997) or Kieras (in press), for a more complete description of EPIC; here is only the necessary minimum description.

In the EPIC architecture, the *eye processor* contains *acuity functions* that specify whether each visual property of each object is currently visible as a function of the size of the object and its eccentricity. The currently available visual properties for each object are represented in the *sensory store*; the *perceptual processor* then encodes the properties of each object, possibly in relation to other objects, and passes the encoded representation on to the *perceptual store* where they are available to the cognitive processor to match the conditions of production rules. The perceptual store contains the current representation of the visual world that cognition can reason and make decisions about, including decisions about where to move the eyes next by commanding the *ocular motor processor*.

When the eyes move away from an object, the properties of the object persist for a short time (e.g. 200 ms) in the sensory store, and when lost, the perceptual processor notes that the corresponding property in the perceptual store no longer has sensory support. After a relatively long time, the property will then be lost from the perceptual store. But if the object disappears completely, it and all of its properties will be removed from the perceptual store fairly quickly. The notion that the representation persists for a considerable time as long as the scene is present is supported by studies summarized by Henderson & Castelhano (2005); memory for previously fixated objects was assessed in natural visual scenes, and retention times of at least several seconds were observed. Since this form of memory has not been studied extensively, its properties and duration must be chosen to fit the modeled data.

Model for the Search Task

The model is an instantiation of the active vision concept; constructing it requires a choice of (1) visual acuity functions and parameters, (2) a model of the "noise" in the eye movements, (3) a parameter for the persistence time of visual properties in the perceptual store that are no longer sensorily supported, and (4) a set of production rules that implement the visual search strategy. Each of these will be described, with emphasis on the new features in this work.

New acuity functions

The availability of a perceptual property in extra-foveal vision depends heavily on the *eccentricity* (the distance in degrees of visual angle from the center of gaze) of the object, normally referred to in degrees of visual angle, and on the *size* of the object (also measured in degrees of visual angle), and on the specific property involved. Despite the many decades of research on vision, the literature does not contain a comprehensive set of parametric data on acuity for different visual properties as a function of their eccentricity and size, especially for the density and properties typical of computer displays. Space limitations do not allow a review of the available data (see Findlay & Gilchrist, 2003).

Previously, EPIC used simple forms of acuity functions that were adequate to fit the limited data such as Williams(1967). The new work here was to anchor the acuity functions closer to the available psychophysical data. Of special interest are studies of "cortical magnification" which is based on the reasoning that a constant amount of visual cortex (presumably supporting a certain number of receptive fields) are required for performing discrimination at a certain level, and since anatomically, the density of cortical representation declines with distance from the fovea, the size of the stimulus must increase to involve the same amount of cortex. Such functions have been measured in psychophysical experiments; a typical result (e.g. Virsu & Rovamo, 1979) is that to maintain discriminability, the size of the object must increase as a cubic function of eccentricity; the required size increases linearly for a moderate eccentricity, and then quite sharply in the further periphery. A cubic function with a moderate linear coefficient, a zero quadratic coefficient and a very small cubic coefficient provides a good fit.Visual search studies such as Carrasco & Frieder (1996) show that if object size is constant, then targets at greater eccentricity are located more slowly, whereas if peripheral objects are magnified in size

according to the measured functions, search time becomes flat with eccentricity. However, it appears that magnification functions measured for individual objects greatly overestimate the acuity for objects in dense visual fields (e.g. see discussion in Anstis, 1974). To measure acuity in dense displays would be very difficult, and the literature does not contain useful parametric studies.

To deal with this non-definitive picture, a simple family of acuity functions are proposed, and their parameters determined by a combination of general constraints set by the literature and iterative maximization of fit in the models. A separate function was specified for each property: color, encoded size (small, medium, large), shape, and text label. The acuity function is a Gaussian detection function that gives the probability that the property will be detected (be available) for an object with size s at eccentricity e:

$$P(\text{detection}) = P(s > N(\mu, \sigma))$$

$$\mu = a + be + ce^2 + de^3,\ \sigma = a\ \text{constant}$$

The form for μ (which can be interpreted as the 50% threshold for object size) reflects the commonly fitted form of cortical magnification functions. The value of σ governs the steepness of the ogival detection function; smaller values of σ make it look more like an all-or-none threshold-like process.

In the preliminary predicted results presented here, the acuity parameters were determined by informal iterative fitting. The a term was held at 0.05, b was estimated as 0.2 for color, size, and shape, and 0.1 for text, c was held at 0, d was 0.0004 for color and size, .025 for shape, .05 for text, and σ was 0.5 for color, size, and shape, and 1.0 for text.

The availability for each property is independently resampled for all objects whenever the eye is moved. As the eye moves around, the available properties of the same object can fluctuate, and will not be reliably available from one fixation to the next. However, the information, once acquired, will remain for some time in the perceptual store.

New model of saccade accuracy

A variety of studies (see Harris, 1995 for a review) have shown that saccades tend to fall short of the actual fixation target, and the standard deviation of the saccade distance tends to be proportional to the distance. Following Harris (1995), the new oculomotor processor samples the distance for a saccade to an object at eccentricity e from a Gaussian distribution:

$$\text{saccade length} = N(\mu, \sigma))$$

$$\mu = g \cdot e,\ \sigma = s \cdot \mu$$

Typical values for g (gain) range from 0.85 - 0.95, and s (spread) is typically around 10%. The current preliminary fits use the values suggested by Harris as optimal, namely g=0.95, s=10%. Unlike previous EPIC models, this model thus often misses the object to be fixated, which decreases the probability that (e.g.) its text label will be available, meaning that the task strategy must either attempt to fixate the object again, or choose a entirely different object to fixate. On the other hand, if the acuity functions are such that most fixations are close enough, there may be little effect of inaccurate saccades.

Fixation memory

As summarized in the task strategy below, memory for previous fixations was implemented by only choosing objects to fixate whose relevant properties are currently unknown, either because the object was never fixated, its properties were not detected, or it was fixated a long time ago but the properties have been lost from the perceptual store. As mentioned below, the duration of properties of visible objects in perceptual store interacts with the acuity functions and model strategy in predicting the properties of repeat fixations. For the model predictions presented here, this duration was set at 15 s.

New task strategy

The visual search strategy in the model is a new variation of a basic strategy that has been used in several EPIC visual search models. There are now three concurrent threads of execution. In the first thread, *nomination rules* now continuously propose objects to fixate whose available visual properties match the cued properties. In the second thread, *choice rules* pick a single candidate from the nominated objects according to a priority scheme, and launch an eye movement to the chosen candidate. The priority scheme favors the more widely available attributes, and so chooses an object with a matching color over one with a matching size over one with a matching shape. If there are no nominations, a "guessed" object is chosen whose cued properties are currently unknown. Objects are only nominated or chosen if their text label property is currently unknown, which serves as a memory for fixations, and if more than one object qualifies, the closest one is chosen. The *response rules* in the third thread wait for the eye movement to the candidate to be complete and either click on the object if its text label matches the target label, or discard it if not, which enables the next choice of object to fixate. If the text label is not available (e.g. the saccade may have fallen short) the strategy waits for up to three additional cycles and then nominates the object for the next eye movement, which takes priority.

Model Results

Using the parameter values and task strategy described above, the model was run using the actual set of stimuli used in the experiment, which consisted of 2112 combinations of cue condition, search fields, and target object within that search field, with 4 repetitions of these stimuli, giving 1056 trials in each experimental condition, and the simulated eye movement and response time data were collected. These fits are preliminary, but encouraging; further work is in progress.

Figures 2-4 above show the observed and predicted statistics in each condition. As a summary measure of the goodness of fit of predicted to observed, r^2=0.89 for the proportions of matching fixations in Figure 2; r^2=0.92 for the number of fixations in Figure 3, and r^2=0.79 for the saccade distances in Figure 4, where it is clear that the distances for the weak cues of Shape and Number-Only are seriously under-predicted. In results not shown graphically, the predicted RTs are well correlated with the observed (r^2=0.92), but the model over-predicts them substantially,

probably due to suboptimal methods for disqualifying a fixated object. Most seriously, the overall predicted repeat rates are too high (41 vs 25% in the observed data), especially in the weak cue conditions; increasing the property decay time from 15 s to 20 s or more improves the fit, but then the number of fixations is under-predicted in the weak cue conditions.

Guided versus unguided fixation choices

An insight from this preliminary modeling work is that there are basically two kinds of fixation choices: a guided fixation when the object matches a cue, and an unguided fixation when there is no candidate that has a matching cue property. While unguided fixations dominate the Number-Only condition, they clearly play a role in the other cue conditions, because fixations to non-matching objects make up a quarter or more of the fixations in these conditions.

For unguided fixations, the model strategy must choose a next object on some basis; if this strategy is incorrect, then all of the summary statistics for a cue condition will be mis-predicted. Several strategies have been explored, but no clear winner is yet evident - for example, choosing a qualified candidate at random, rather than the closest, produces a different pattern of mis-predictions in the weak cue conditions. The next steps in this work will separate fixations on matching objects from non-matching, which should help characterize guided versus unguided fixation choice strategies.

Conclusion

This model represents a realization of the active vision concept in terms of a computational cognitive architecture whose components incorporate noisy saccades, size and eccentricity effects in perception, and a persistent visual perceptual store that represents the current visual situation and provides a memory for previous fixations. The task strategy implements visual guidance by using the supplied target properties and the information in the visual perceptual store to choose the next object to fixate. The adequacy of the architecture, and a deeper understanding of the functional properties of the visual system, will emerge as the model is developed to more closely account for the eye movement data.

Acknowledgements

This work was supported by the National Science Foundation, grant number IIS-1017593, and the Office of Naval Research, Cognitive Science Program, grant numbers N00014-10-1-0152 and N00014-13-1-0358. Sam Dodson assisted in running the visual search experiment.

References

Anstis, S.M. (1974). A chart demonstrating variations in acuity with retinal position. *Vision research*, 14, 589-592.

Carrasco, M., & Frieder, K.S. (1996). Cortical magnification neutralizes the eccentricity effect in visual search. *Vision Research*, 37, 63-82.

Findlay, J.M., & Gilchrist, I.D. (2003). *Active Vision.* Oxford: Oxford University Press.

Gordon, J., & Abramov, I. (1977). Color vision in the peripheral retina. II. Hue and saturation. *Journal of the Optical Society of America*, 67(2), 202-207.

Harris, C.M. (1995). Does saccadic undershoot minimize saccadic flight-time? A Monte-Carlo study. *Vision Research*, **35**, 691-701.

Henderson, J.M. & Castelhano, M.S. (2005). Eye movements and visual memory for scenes. In G. Underwood (Ed.), *Cognitive processes in eye guidance*. New York: Oxford University Press. 213-235.

Hornof, A. J. (2001). Visual search and mouse pointing in labeled versus unlabeled two-dimensional visual hierarchies. *ACM Transactions on Computer-Human Interaction*, 8(3), 171-197.

Kieras, D.E. (in press). A summary of the EPIC Cognitive Architecture. In S. Chipman (Ed.), *The Oxford Handbook of Cognitive Science*.

Kieras, D. (2009). The persistent visual store as the locus of fixation memory in visual search tasks. In A. Howes, D. Peebles, R. Cooper (Eds.), *9th International Conference on Cognitive Modeling – ICCM2009*, Manchester, UK

Kieras, D. (2010). Modeling Visual Search of Displays of Many Objects: The Role of Differential Acuity and Fixation Memory. T*he 10th International Conference on Cognitive Modeling – ICCM2010*, August 6-8, 2010, Philadelphia, PA.

Kieras, D.E & Hornof, A.J. (2014). Towards accurate and practical predictive models for active-vision-based visual search. In *Proceedings of CHI 2014: Human Factors in Computing Systems.* New York: ACM, Inc.

Kieras, D.E, & Marshall, S.P. (2006). Visual Availability and Fixation Memory in Modeling Visual Search using the EPIC Architecture. *Proceedings of the 28th Annual Conference of the Cognitive Science Society*, 423-428.

Meyer, D. E., & Kieras, D. E. (1997). A computational theory of executive cognitive processes and multiple-task performance: Part 1. Basic mechanisms. *Psychological Review*, **104**, 3-65.

Peterson, M.S., Kramer, A.F., Ranxiao, F.W., Irwin, D.E., & McCarley, J.S. (2001). Visual search has memory. *Psychological Science*, **12**, 287-292).

Sanders, M. S., & McCormick, E. J. (1987). *Human factors in engineering and design* (6th ed.). New York: McGraw-Hill.

Virsu, V. & Rovamo, J. (1979). Visual resolution, contrast sensitivity, and the cortical magnification factor. *Experimental Brain Research*, **37**, 475-494.

Williams, L.G. (1967). The effects of target specification on objects fixated during visual search. In A.F. Sanders (Ed.) *Attention and Performance*, North-Holland. 355-360.

Zhang, Y., & Hornof, A. J. (2014). Easy post-hoc spatial recalibration of eye tracking data. *Proceedings of ETRA 2014: Eye Tracking Research and Applications*, 95-98.

An Adaptable Implementation of ACT-R with Refraction in Constraint Handling Rules

Daniel Gall (daniel.gall@uni-ulm.de)
Institute of Software Engineering and Compiler Construction, Ulm University
89069 Ulm, Germany

Thom Frühwirth (thom.fruehwirth@uni-ulm.de)
Institute of Software Engineering and Compiler Construction, Ulm University
89069 Ulm, Germany

Abstract

ACT-R is a popular cognitive architecture. Although its psychological theory is well-investigated, it lacks a formal foundation. This inhibits computational analysis of cognitive models and leads to technical artifacts in ACT-R implementations.

In this paper we present an adaptable implementation of ACT-R derived from our formalization presented in previous work. We show how this formal approach supported by the use of the declarative programming language Constraint Handling Rules (CHR) leads to an implementation of the ACT-R close to the theory while maintaining interoperability. Due to the adaptability of our implementation we are able to extend the conflict resolution strategy of the system by production rule refraction in our implementation easily avoiding the problem of over-programming in some ACT-R models. The use of CHR facilitates the application of analytical methods from the CHR ecosystem paving the way for ACT-R model analysis.

Keywords: ACT-R, Constraint Handling Rules, conflict resolution, refraction, model analysis

Introduction

ACT-R is a popular cognitive architecture with lots of users and application models. The psychological theory is well-investigated and allows for studying of human-behavior by performing experiments both with humans and artificial ACT-R agents. However, from a computational point of view, ACT-R lacks a formal theory of its underlying computational concepts which inhibits analysis of the computational properties of a model. Like in every production rule system, properties like confluence (i.e. the order of rules does not matter for the result) and termination can be important to the soundness of a cognitive model. While confluence may be regained by a conflict resolution mechanism, unwanted or unexpected non-confluence could be the result of a programming error. Confluence analysis can help to detect all rules that could lead to undesired behavior and help the modeler to check the validity of his model.

In this paper, we present our translation of ACT-R models to the language Constraint Handling Rules (CHR). CHR is a declarative rule-based programming language which comes from the field of logic programming (Frühwirth, 2009). Due to the close relation of CHR to logic, a formalization of the ACT-R production rule system can be derived from the translation. This closes the gap between the formalization and execution of ACT-R models simplifying analysis. There are analysis methods and tools for CHR programs, e.g. for analysis of confluence, termination and operational equivalence which can be applied to ACT-R models. In (Gall & Frühwirth, 2014b) we have shown the first steps towards such an analysis toolbox for ACT-R by defining an abstract operational semantics of its procedural system and showing soundness and completeness of our translation with respect to the CHR very abstract semantics (Frühwirth, 2009). This result is crucial for lifting CHR results to ACT-R. Another benefit of our implementation is its adaptability. We exemplify this by adding refraction to the conflict resolution mechanism of ACT-R. Refraction inhibits rules to fire twice on the same state.

The contributions of this paper are a concise description of our adaptable implementation of ACT-R with CHR, the implementation of refraction as the first of its kind in ACT-R and its application to simplify an existing example model.

We concentrate on the symbolic parts of ACT-R in this paper, since we want to explain the formalization of the basic production rule system with a generalized conflict resolution. Nevertheless, in our implementation we have captured various conflict resolution mechanisms (Gall & Frühwirth, 2014a) and the declarative module with its sub-symbolic concepts (Gall, 2013). The restriction is also closer to our abstract semantics used for confluence analysis of ACT-R models: It is necessary to capture all possible transitions non-deterministically to find all transitions that could inhibit confluence. The abstraction can then be augmented by the details making our results applicable to actual ACT-R implementations in a next step.

Constraint Handling Rules

First, we give an informal introduction to the programming language Constraint Handling Rules which is the basis of our implementation. For a detailed description of the language we refer to (Frühwirth, 2009) and (Frühwirth, 1998). CHR programs consist of a set of *rules* operating on a *constraint store* comparable to the working memory in other production systems. Given an initial constraint store, matching rules are applied to the store to exhaustion. The data elements of the store are *(CHR) constraints* which are first-order predicates of the form $c(t_1,\ldots,t_n)$. For instance, `name(robert)`, `age_of(robert,75)` or `b` are constraints. The terms in the arguments of a constraint can also contain variables that are denoted by capital letters, e.g. `X`.

There are three types of rules in CHR – simplification, propagation and simpagation:

```
simplification @ Hr <=> G | B.
propagation    @ Hk ==> G | B.
simpagation    @ Hk \ Hr <=> G | B.
```

H_r, H_k and B are conjunctions of CHR constraints, whereas the so-called guard G consists of a conjunction of simple built-in tests like arithmetic comparisons, syntactic equality etc. comparable to the modifiers `-`, `<`, `>` in ACT-R production rules. Guards are optional and can be omitted.

Simplification rules match the constraints on the left-hand side of the rule with the store binding variables of the rule with the contents of the store. If matching constraints are found and the tests in the guard hold, the matching constraints are removed from the store and replaced with the constraints in the body B. An example simplification rule is `blue, yellow <=> green` modeling the mixture of the colors blue and yellow. If both colors are found in the store, they are replaced by the color green.

In contrast to simplification rules, propagation rules leave the matching constraints in the store and add the body. Simpagation rules are a mixture of both rules: The constraints in H_k are kept in the store, whereas the constraints in H_r are removed.

To execute CHR programs, an initial constraint store is specified by a so-called query and then rules are applied to exhaustion: E.g., a program consisting of the color-mixing rule from above would simplify the query `blue, blue, yellow` to `blue, green`. Since there are no more yellow – blue pairs in the store, the rule is not applicable anymore.

In implementations, CHR rules and queries are applied in textual order: from top to bottom and from left to right. Rules in CHR can be read as logical formulae giving programs a declarative semantics (Frühwirth, 2009). The logical reading of CHR programs together with analysis methods and tools for e.g. confluence or termination makes CHR suitable for program analysis.

The Basic Implementation of ACT-R in CHR

For an introduction of ACT-R we refer to (Anderson et al., 2004; Anderson & Lebiere, 1998; Taatgen, Lebiere, & Anderson, 2006). Our formalization and implementation of ACT-R and hence the following description of the implementation of ACT-R features is based on those sources together with the ACT-R reference manual (Bothell, n.d.).

Our implementation of ACT-R in CHR consists of two parts: a *compiler* and a *runtime environment*. The compiler takes a set of ACT-R production rules and translates it to CHR rules and the runtime environment implements the features necessary to execute the translated CHR rules according to ACT-R.

Basic Translation of ACT-R models to CHR

This section describes the work of the compiler translating ACT-R models to CHR rules. The description first explains how the state of the procedural system of ACT-R can be represented in terms of constraints. Based on these considerations, the actual translation of production rules is described.

States The state of the procedural module in ACT-R is represented by the chunks in the buffers. We can represent a chunk by the following constraints: a constraint `chunk(cname,type)` and for each slot-value pair of the chunk a `chunk_has_slot(cname,slot,value)` constraint. The buffers with their contents are represented by `buffer(bname,cname)`. There are some assumptions on the state, like for instance "There is at most one `chunk` constraint for each chunk name", "There is at most one `buffer` constraint for each buffer", "There is at most one `chunk_has_slot` constraint for each combination of chunk name and slot" and "Slots and types are consistent". Those assumptions follow from our formal description of ACT-R in (Gall, 2013) and (Gall & Frühwirth, 2014a). The assumptions can be checked by some simple CHR rules, if needed. They play an important role for model analysis.

Production Rules We translate each ACT-R rule to a CHR propagation rule, i.e. the tested constraints are left in the store. The buffer tests on the left-hand side of an ACT-R rule consist of a buffer name, usually a chunk type and a set of slot-value pairs. Such a test signifies that the specified buffer holds a chunk of the specified type with specified slot-value pairs. The values of the slot-value pairs can also be variables which are bound to the actual values of the slot in the buffer. The following rule shows the translation of such tests to a set of constraints. Actions are translated to special action constraints described later:

```
buffer(b,C), chunk(C,t), chunk_has_slot(C,s,v), ...
==> action(...), ... .
```

`C` is a fresh variable which will be bound to the name of the chunk in the specified buffer with name `b`. The other conditions expressed by constraints depend on the value of `C`, specifying that `C` has to have the type `t` and respective values in its slots (for each slot-value pair in the original rule, a `chunk_has_slot` constraint is in the head of the CHR rule). If the value of a slot-value pair is a variable it is also translated to a variable in the resulting rule.

The actions on the right-hand side of an ACT-R rule are translated to constraints `buffer_modification(b,CD)` for modifications, `buffer_request(b,CD)` for requests and `buffer_clearing(b)` for clearings. In those constraints `b` stands for the buffer name the action refers to and `CD` is a term `chunk(CName,Type,LSVP)` which describes the chunk defined by the action in the rule. If name `CName` or type `Type` are not defined they have no value (i.e. they remain an unbound variable). `LSVP` is a list of slot-value pairs taken directly from the original rule, e.g. `[(s1,v1), (s2,v2)]`. For each of the three action constraints there is a rule in the run-time environment which actually performs the action described in the next section.

In ACT-R, the procedural module blocks as soon as a rule has been selected to fire. Contrarily, CHR implementations execute the right-hand side of a rule depth-first from left to right. This leads to the effect that after the adding of the first constraint on the right-hand side of a CHR rule, the next rule

might fire directly, before the other constraints are added. The behavior of ACT-R can be modeled by two phases: a *match* phase and an *apply* phase. Those two phases are represented in CHR by a constraints `match`. The presence of this constraint in the store indicates that the procedural module is in the *match* phase. Each ACT-R production rule can be translated to a CHR rule as follows:

buffertests `\ match <=>` *bufferactions*`, match.`

If all buffer tests succeed and the program is in the *match* phase, i.e. a `match` constraint is in the store, then this constraint is removed prohibiting other rules from firing and the actions are performed. At the end, a new *match* constraint is added allowing other rules to fire.

Runtime Environment

The runtime environment is a framework which offers some features needed to actually execute the rules produced by the compiler.

Scheduler ACT-R implementations usually include a scheduling unit which takes track of the points in simulation time when a certain event is executed. Events in this context are triples `(T,P,C)` signifying that the constraint `C` is added to the store at simulation time `T` with priority `P`. In ACT-R, priority decides which event is executed first, if they are due at the same simulation time. The constraints added can be for example the phase controlling constraint `match` or the action of a production rule. The scheduler implements the following interface described in table 1.

Table 1: Interface of the scheduler.

Constraint present	Action
`get_time(T)`	`T` is bound to current simulation time
`add_event(T,P,C)`	event triple `(T,P,C)` is memorized
`next_event`	constraint from event with smallest time (and highest priority) is added to the store

Production Rule Actions In the last section, the production rule actions have been translated to some special constraints. In the following, we describe the rules in the runtime environment which perform the actions specified by these constraints. For details consult (Gall, 2013).

The rule for *buffer modifications* ignores any name or type in the chunk description, since they are not allowed to be modified (Bothell, n.d.). The symbol _ denotes an anonymous variable.

```
buffer(B,C) \ buffer_modification(B,chunk(_,_,SVP))
<=> modify_slots(C,SVP).
```

It adds a constraint `modify_slots` which takes care of the actual modification of the individual slots in the chunk of the specified buffer. The *buffer clearing* action is implemented similarly. However, for the *request* action, the implementation involves a module which returns a result. The modules have their individual constraint stores and simply have implement the following interface: They have a rule which reacts if a constraint `module_request(Request,Result,Time)` is added to their store. `Request` is the chunk description from the original production rule and it is the only argument with a value. The request action then binds the variable `Result` to a chunk description with the result chunk and the variable `Time` to the simulation time it takes to calculate the request. The request action is then implemented as follows:

```
buffer_request(B,Request)
<=> M:module_request(Request,Result,Time),
    get_time(Now),
    add_event(Now+Time,P,replace_chunk(B,Result)).
```

In this code `M` is the module associated with buffer `B`. The `replace_chunk` is a helper constraint which removes the old chunk from the buffer and adds the result chunk to the store. It is scheduled at the moment the request has finished, i.e. that the result of the request is only applied after the module has finished the request.

Modules As described before, a module can be added to the system by writing a new Prolog module which implements a rule reacting on a `module_request` constraints. Such Prolog modules have their own constraint store which does not interfere with the store from the procedural system.

Implementation of Advanced Concepts

By now we have seen how ACT-R rules can be translated to CHR and how the ACT-R framework can be built using CHR. However, we have ignored timings and therefore scheduling. We extend our translation scheme with those considerations and show the ACT-R main cycle. Furthermore, the basic implementation of conflict resolution is described.

Scheduling

To realize scheduling of production rule action, we translate each ACT-R rule to two CHR rule of the following form:

```
buffertests \ match
<=> get_time(Now),
    add_event(Now+0.05, 0,
      apply_rule(rule_name, buffertests).

buffertests \ apply_rule(rule_name, buffertests)
<=> schedule actions now,
    get_time(Now), add_event(Now, -10, match).
```

In the first rule, the `match` constraint is removed to end the match phase. Since in ACT-R the procedural module block for 50 ms simulation time, the rule application is postponed by that time by scheduling an `apply_rule` event 50 ms from the current time with the rule name as argument. Additionally, the buffer tests, i.e. the resulting variable bindings, are memorized in a second argument making sure that the conditions still match at the rule application time.

The second rule can be applied after the scheduler has added the corresponding `apply_rule` constraint and the memorized variable bindings in the buffer test still apply. Then the actions are scheduled at the current time with priorities as defined in (Bothell, n.d.). Finally, a `match` constraint

is scheduled at the current time with low priority to make sure that the actions are performed before the next rule can match. Note that for requests first are only scheduled to be stated at the current simulation time. The resulting changes are applied after a time offset defined by the requested module.

ACT-R Main Cycle

With the scheduler and the rules modified for scheduling, the ACT-R main cycle can be built: First, the initialization code is executed (adding chunks, ...), then the initial time is set to 0 and an initial event `match` is added to the scheduling queue. As soon as there are no more events in the queue, computation is stopped.

Conflict Resolution

The current considerations have ignored the case when more than one production rule is applicable. Like other production rule systems, ACT-R resolves such conflicts by a certain conflict resolution strategy. In the CHR implementation described so far, only the first matching rule will be applied, since CHR tries rules in textual order.

To implement conflict resolution, we exchange the first rule of our translation scheme for production rules by the following rule scheme:

```
buffertests, match
  ==> conflict_set(rule(rule_name, buffertests)).
```

Since we have a propagation rule, the `match` constraint stays in the store, even if a matching rule has been found. Instead of scheduling the rule application directly, the rule is added to the so-called *conflict set* which collects all matching rules by adding a corresponding `conflict_set` constraint memorizing the rule and its matching variable binding.

Since the `match` constraint is still present afterwards, another rule can match again. In the end, the constraint store is filled with `conflict_set` constraints of all matching rules. As a last rule, we remove the `match` constraint and start the *select* phase which selects the rule being applied according to a certain strategy: `match <=> select.`

In the runtime environment, we can simply add a rule which reacts on the presence of a `select` constraint and prunes the conflict set. For instance, the rule could simply select on arbitrary rule and discard all other rules without defining an order on the rules:

```
select, conflict_set(R1) \ conflict_set(R2) <=> true.
select, conflict_set(R)
<=> get_time(Now),
    add_event(Now+0.05, 0, apply_rule(R)).
```

The first of the two rules will remove `conflict_set` constraints repeatedly, as long as there is more than one such constraint in the store. As soon as only one `conflict_set` constraint is left, the second rule can be applied which simply schedules the rule application event of that rule.

In practice, the conflict resolution depends on some properties of the rules. This only needs a slight adjustment of the conflict set pruning rule. For example, production utilities can be taken into account. In (Gall & Frühwirth, 2014a) we have shown that our implementation allows to exchange the conflict resolution mechanism by simply exchanging the reaction on the select constraint. Since the rules for conflict resolution are split into a separate module, it suffices to exchange this file to modify conflict resolution. We refer to the original paper for details.

Implementation of Refraction

In (Young, 2003) the question is raised, if ACT-R should include rule refraction. Refraction is a concept introduced in (McDermott & Forgy, 1977) as a possible conflict resolution strategy for production rule systems. It inhibits production rules from firing twice on the exact same instance.

Young argues that the lack of refraction can lead to (as he calls it) *over-programming*, i.e. determining the order of rule applications in advance. This aspect destroys declarativity of ACT-R models, since it follows a more imperative – i.e. step-by-step/state-by-state – thinking. This seems to be inelegant and leads to the problem that adjustments of one production rule result in changing every production rule (Young, 2003). Furthermore, the question if such state-aware production rules explain human cognition can also be raised. Although it has been discussed in the community, to the best of our knowledge, refraction has not been included in the ACT-R reference implementation by now.

In the following, we describe how refraction can be included with our CHR implementation and exemplify again how easy it is to exchange fundamental parts of our implementation due to the power of CHR and logic programming. First of all, we memorize the instantiation of a rule that is being applied in an `instantiation` constraint. The instantiation of a rule is a list of all constraints with their matching values. Hence, we can use a propagation rule which reacts on the presence of an `apply_rule` constraint: `apply_rule(R) ==> instantiation(R).`

Before the actual conflict set pruning of an arbitrary conflict resolution strategy (as described before), we add a rule which removes a production rule from the conflict set if it is present in an `instantiation` constraint from the store:

```
instantiation(R) \ conflict_set(R) <=> true.
```

With those two rules the basic refraction mechanism is implemented. Rules should fire again on the same instantiation, if there were changes in one of the involved parts of the instantiation. In this case, if the instantiation has been "touched" intermediately, the `instantiation` constraint should be removed from the store to allow the rule to fire again on this instantiation if it occurs again later. For ACT-R this is the case if a modification or a request have changed the content of one of the buffers of the instantiation in the meantime. Hence, we add two rules which detect changes in `buffer` or `chunk_has_slot` constraints:

```
buffer(B,C1) \ instantiation(rule(_,Hk,_))
 <=> member(buffer(B,C2),Hk), C1 \== C2 | true.
chunk_has_slot(C,S,V1) \ instantiation(rule(_,Hk,_))
 <=> member(chunk_has_slot(C,S,V2),Hk), V1 \== V2 |
      true.
```

Those two rules react if new `buffer` or `chunk_has_slot` constraints enter the store. The guard with the `member` check tests if there is a constraint referring to the same buffer or chunk but having different values. In this case obviously a modification or request occurred and hence this particular instantiation can be removed from the history. Note that the use of refraction in our implementation is optional and can be exchanged and even combined with other conflict resolution strategies (Gall & Frühwirth, 2014a).

Evaluation

We show in an example model, how refraction can simplify the rules of a cognitive model and make them less imperative, i.e. defined from state to state. Our example is derived from the semantic model from ACT-R tutorial unit 1 (*The ACT-R 6.0 Tutorial*, 2012). This model implements a taxonomy of some animals and adds information about some of their properties. For example, it categorizes animals in categories like *fish* and *birds*. Additionally, properties like *swims* or *dangerous* are annotated to categories and representatives. We shortly describe the subset of the model which is the objective of our example.

The knowledge is organized as chunks of type `property` with slots `object` for the name of the object, e.g. *shark*, `attribute` for an attribute of the object, e.g. *dangerous* or *category*, and `value` for the value of the attribute, e.g. *true* in case of the *dangerous* attribute of the shark or *fish* for *category*. In the following, we concentrate on chunks with the attribute category. The goal of the model is to judge if a certain object is member of a category. Such a goal is encoded in a chunk of type `is-member` with slots `object` for the object to judge, `category` for the category, and `judgment` for the result of the query encoded by this chunk.

In a first initialization step, the model requests a chunk from declarative memory which refers to the object the judgment refers to and which has the attribute category, since it wants to deduce the membership of the category (and not something about other properties of the objects) (see the rule `initial-retrieve` in figure 1). To judge the membership of an object in a category, the model can verify the membership directly, if the retrieved chunk already contains the information that the object is a member of the queried category (rule `direct-verify`). Otherwise, it can chain through the categories, i.e. take the category of the found object and check if it is a subcategory of the queried category (rule `chain-category`). Note that in our description the rule to deduce failure has been omitted due to space reasons.

The rules are over-programmed in the sense that the judgment slot always gets a value determining the state of the derivation. For instance, in the beginning, the judgment is expected to be `nil` and is then changed to `pending`. This prevents the first rule from firing repeatedly leading to an endless loop. The rules `direct-verify` and `chain-category` check if the judgment is pending in the beginning. This shows the imperative thinking behind such rules: Those two rules are always meant to fire after the initialization. This is ensured by an artificial state slot in the goal chunk.

```
(P initial-retrieve
  =goal>
     ISA      is-member
     object   =obj
     category =cat
     judgment nil
 ==>
  =goal>
     judgment pending
  +retrieval>
     ISA      property
     object   =obj
     attribute category)
```

```
(P direct-verify
  =goal>
     ISA      is-member
     object   =obj
     category =cat
     judgment pending
  =retrieval>
     ISA      property
     object   =obj
     attribute category
     value    =cat
 ==>
  =goal>
     judgment yes)
```

```
(P chain-category
  =goal>
     ISA       is-member     object     =obj1
     category  =cat          judgment   pending
  =retrieval>
     ISA       property      object     =obj1
     attribute category      value      =obj2
                           - value      =cat
 ==>
  =goal>
     object    =obj2
  +retrieval>
     ISA       property      object     =obj2
     attribute category)
```

Figure 1: Rules of the semantic model

With refraction, we can simplify the rules as follows: `direct-verify` and `chain-category` do not have to check for the `judgement` slot in their conditions and the rule `initial-retrieve` is not required to change the judgment slot of the goal in its actions to `pending` because it never fires again on the same goal. You can find the translation of the model to CHR together with commented example derivations with and without refraction on our homepage[1]. It can be seen that without refraction, the model ends in an infinite loop due to the repeated application of the initialization rule. With refraction, the model derives the correct judgments. For the initialization rule it seems legitimate to check for a `nil` judgment, since this is encodes that the goal has not yet been achieved. It could also check for values other than `yes` or `no` or check if the retrieval buffer is empty or does not contain a suitable chunk for the problem. This might further increase the declarativity of the model.

Related Work

As mentioned before, (Young, 2003) has raised the question if production rule refraction should be included in ACT-R. (Lebiere & Best, 2009) refer to this idea and discuss how the lack of refraction can lead to pathological behavior of a model like infinite looping. The authors also point out that the traditional strategies to avoid such behavior are difficult to model

[1] `http://www.uni-ulm.de/?id=59460`

and sometimes also lack cognitive plausibility. The paper also mentions strategies to inhibit repeated retrieval of declarative memory addressing another architectural problem that could be included in our implementation of declarative memory.

ACT-R has also been implemented in Python (Stewart & West, 2006, 2007) and Java (jACT-R, n.d.; Salvucci, n.d.). These implementations do not concentrate on formalization and analysis. We thank the reviewers for pointers to work on the formalization of cognitive modeling in general (Cooper & Fox, 1998; Howes, Vera, Lewis, & McCurdy, 2004). We plan to investigate how those approaches relate to our work.

Conclusion

In this paper we have presented our implementation of ACT-R in CHR including the translation of ACT-R models to CHR rules and the embedding of the basic ACT-R framework in CHR. We then explained the implementation of some more specific concepts of ACT-R like conflict resolution and finally refraction. To the best of our knowledge, our implementation is the first to include refraction in ACT-R.

It can be seen that the fundamental ideas of ACT-R – rules, chunks, scheduling and conflict resolution – can be captured concisely and elegantly in CHR. By the implementation of refraction and the previous work in (Gall & Frühwirth, 2014a), we have shown the adaptability of our implementation.

Due to the conciseness and declarativity of the rules needed to describe the ACT-R in terms of Constraint Handling Rules, a formalization of ACT-R can be derived from our implementation. We have shown parts of this formalization in (Gall, 2013; Gall & Frühwirth, 2014a, 2014b). The formalization together with the analysis tools of the CHR world pave the way for an ACT-R analysis toolbox. In (Gall & Frühwirth, 2014b), we have taken the first steps towards ACT-R analysis by formulating and abstract operational semantics which is sound and complete with respect to the very abstract semantics of CHR.

Although we have concentrated on the procedural system of ACT-R in this particular work, we want to emphasize that we also have implemented other components like the declarative module with concepts like chunk activation. For details, we refer to (Gall, 2013).

For the future, we want to extend our ACT-R implementation by more features known from the ACT-R world. We also want to investigate how CHR analysis tools in detail can be applied to ACT-R models. Additionally, we plan to extend our abstract operational semantics with more details from ACT-R to investigate how they relate. Finally, the transfer of concepts, methods and ideas from the ACT-R production rule system like activation or reinforcement learning based conflict resolution could be included in an extension of CHR.

References

The ACT-R 6.0 tutorial. (2012). Retrieved from `http://act-r.psy.cmu.edu/actr6/units.zip`

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological Review*, *111*(4), 1036–1060.

Anderson, J. R., & Lebiere, C. (1998). *The atomic components of thought*. Lawrence Erlbaum Associates, Inc.

Bothell, D. (n.d.). Act-r 6.0 reference manual – working draft [Computer software manual]. Pittsburgh, Pennsylvania 15213.

Cooper, R., & Fox, J. (1998). Cogent: A visual design environment for cognitive modeling. *Behavior Research Methods, Instruments, & Computers*, *30*(4), 553–564.

Frühwirth, T. (1998). Theory and practice of constraint handling rules. *The Journal of Logic Programming*, *37*(1–3), 95–138.

Frühwirth, T. (2009). *Constraint Handling Rules*. Cambridge University Press.

Gall, D. (2013). A rule-based implementation of ACT-R using Constraint Handling Rules. *Master Thesis, Ulm University*.

Gall, D., & Frühwirth, T. (2014a). Exchanging conflict resolution in an adaptable implementation of ACT-R. *Theory and Practice of Logic Programming*, *14*, 525–538.

Gall, D., & Frühwirth, T. (2014b). *A Formal Semantics for the Cognitive Architecture ACT-R.* (Tech. Rep.).

The Homepage of jACT-R. (n.d.). Retrieved from `http://jactr.org/`

Howes, A., Vera, A., Lewis, R. L., & McCurdy, M. (2004). Cognitive constraint modeling: A formal approach to supporting reasoning about behavior. In *Proc. cognitive science society* (pp. 595–600).

Lebiere, C., & Best, B. J. (2009). Balancing long-term reinforcement and short-term inhibition. In *Proceedings of the 31st annual conference of the cognitive science society* (pp. 2378–2383).

McDermott, J., & Forgy, C. (1977, June). Production system conflict resolution strategies. *SIGART Bull.*(63), 37–37.

Salvucci, D. (n.d.). *ACT-R: The Java Simulation & Development Environment – Homepage.* Retrieved from `http://cog.cs.drexel.edu/act-r/`

Stewart, T. C., & West, R. L. (2006). Deconstructing ACT-R. In *Proceedings of the seventh international conference on cognitive modeling* (p. 298303).

Stewart, T. C., & West, R. L. (2007, September). Deconstructing and reconstructing ACT-R: exploring the architectural space. *Cognitive Systems Research*, *8*(3), 227–236.

Taatgen, N. A., Lebiere, C., & Anderson, J. (2006). Modeling paradigms in ACT-R. In *Cognition and multi-agent interaction: From cognitive modeling to social simulation.* (pp. 29–52). Cambridge University Press.

Young, R. M. (2003). Should ACT-R include production refraction? In *Proceedings of 10th Annual ACT-R Workshop.*

Supraarchitectural Capability Integration: From Soar to Sigma

Paul S. Rosenbloom (Rosenbloom@USC.Edu)
Institute for Creative Technologies & Department of Computer Science, University of Southern California
12015 Waterfront Dr.
Playa Vista, CA 90094 USA

Abstract

Integration across capabilities, both architectural and supraarchitectural, is critical for cognitive architectures. Here we revisit a classic failure of supraarchitectural capability integration in Soar, involving *data chunking*, to understand better both its source and how it and related integration issues can be overcome via three general extensions in Sigma.

Keywords: Cognitive architecture; integration; declarative learning; Soar; Sigma.

Many of the most important early results from Soar concerned how integration across a small general set of architectural mechanisms, plus appropriate knowledge above the architecture, could yield a wide variety of problem solving and learning capabilities (Laird, Newell & Rosenbloom, 1987). Because these capabilities all intrinsically involved forms of knowledge above the architecture, in addition to mechanisms within the architecture, they are on the whole most appropriately considered *supraarchitectural*; i.e., above the architecture.

Some supraarchitectural capabilities, such as lookahead search across metalevels, became part of the toolkit available for routine use in more comprehensive systems – in this case via a set of default rules that were loaded whenever Soar was initialized and usable whenever a tie occurred among operators proposed for selection. However, others of these capabilities – such as declarative learning via what came to be called *data chunking* (Rosenbloom, Newell & Laird, 1991) – proved impossible to deploy routinely in combination with other capabilities, and thus never amounted to more than standalone demonstrations.

Such failures in *supraarchitectural capability integration* loomed over Soar for years as one of its most significant flaws. In the case of declarative learning, the inability to integrate it routinely with other capabilities was ultimately accepted as a fundamental limitation in Soar, triggering a dramatic shift to an approach in which new declarative memory and learning modules were implemented in Soar 9 for routine use in conjunction with other capabilities (Laird, 2012). This move then helped trigger an even broader shift in Soar from its early emphasis on uniformity to its more diverse present state, while also aligning it more closely with ACT-R's long-term approach (Anderson *et al*., 2004).

Sigma (Rosenbloom, 2013) is a more recent architecture that is based on combining what has been learned from over three decades of separate work in cognitive architectures – Soar in particular – and graphical models (Koller & Friedman, 2009). One of the three key desiderata driving the development of Sigma – *functional elegance* – is a reformulation of Soar's earlier notion of uniformity. Sigma maintains many of the high level concepts from Soar, yet it has revealed an ability both to embody a wider variety of supraarchitectural capabilities and to integrate them together routinely. Here we analyze what has enabled Sigma to overcome this earlier fundamental limitation in Soar.

The key to integration of supraarchitectural capabilities is to fit them naturally within the system's overall processing and control structure, which for both Soar and Sigma can range from reactive to deliberative to reflective. Reactive processing can be thought of as parallel, memory driven, automatized, or System 1. It may include basic forms of perception, memory access, reasoning and decisions, but it is limited to what can be accomplished within a single cognitive cycle; i.e., ~50 msec in people. Deliberative processing can be thought of as algorithmic, knowledge intensive, or controlled. It comprises routine sequential behavior based on sufficient expertise to always know what to do. Reflective processing deals with situations that are problematic – yielding impasses and metalevels – and can be thought of as search driven or System 2.

Both of the supraarchitectural capabilities mentioned earlier – lookahead search and data chunking – are implemented reflectively in Soar; that is, an impasse must occur that halts normal processing before the metalevel processing necessary for the capabilities can proceed. In the former case, hypothetical reasoning about the future occurs, as necessary, across metalevels. In the latter case, declarative knowledge structures must be explicitly assembled within a metalevel in order for chunking – Soar's sole learning mechanism at the time – to learn new rules from them. Although chunking can occur each decision, it is an inherently reflective learning mechanism because it learns from traces of rules that fire in metalevels.

Reflective integration is unproblematic for lookahead search because an impasse has already brought normal processing to a halt. However, normal processing could continue in the absence of data chunking, and an artificially induced impasse is in fact required to enable it. Thus, reflective integration is natural in the former case, but both artificial and intrusive in the latter case, where reflection is in service of learning for the future rather than solving the current problem. This is not to say that deliberate reflective learning can't be appropriate or natural, as in after-action review or post-problem metacognition, or that reflective learning can't occur naturally as a side effect of metalevel problem solving – as with chunking – but it can be inappropriate and intrusive when pursued deliberately during task performance.

Yet, declarative learning – both semantic and episodic – must be able to occur continually on a routine basis. Since data chunking could not achieve this, and Soar's formulation of supraarchitectural reactivity – in terms of parallel knowledge access (i.e., rule firing) until quiescence, followed by a decision – could support no other approaches, distinct semantic and episodic memories and learning mechanisms were added in the Soar 9 architecture to enable declarative learning to proceed reactively and in parallel.

Sigma, in contrast, succeeds because of three general extensions to the reactive level. The first extension is to support a more general form of knowledge structure – one that is *hybrid* (discrete + continuous) and *mixed* (symbolic + probabilistic) – and thus also a more general form of reactive reasoning. This enables Sigma not only to perform symbolic reasoning in parallel – as was supported by Soar's parallel rule system – but also probabilistic reasoning and signal processing. Data chunking was a purely symbolic approach to declarative memory, but declarative memory in Soar 9, and in ACT-R before it (Anderson *et al*., 2004), has a strong activation-based subsymbolic component. This aspect is provided in Sigma's supraarchitectural declarative memories via reactive probabilistic reasoning.

The second extension is that, instead of only making decisions about which action to perform next, Sigma can in parallel make decisions about any of the values in working memory. This enables not only reactive retrieval of distributions from declarative memory, but also reactive selection of the best choices from these distributions. In Sigma, declarative retrieval is thus inherently reactive, with deliberative retrieval arising only as necessary (through explicit manipulation of cues across decisions). Declarative retrieval in Soar has traditionally been deliberative, even in Soar 9, although a more reactive mode has recently been introduced (Li & Laird, 2015).

The third extension is the inclusion of a reactive learning mechanism based on *gradient descent* that updates parameters everywhere in long-term memory once per cognitive cycle (Rosenbloom, Demski, Han & Ustun, 2013). Instead of embodying one general reflective learning mechanism, as in the early days of Soar, or this plus multiple memory-specific reactive learning mechanisms, as in Soar 9, Sigma supports a single general reactive learning mechanism. This is adequate for learning not only the contents of semantic and episodic memory (Rosenbloom, 2014), but it can also acquire: Q functions in reinforcement learning; models of actions that are experienced; maps (as part of SLAM); and perceptual and transition functions in speech recognition (Joshi, Rosenbloom & Ustun, 2014). Moreover, because it operates in parallel over all parameters in Sigma's long-term memory, multiple reactive supraarchitectural learning capabilities can proceed without interference with each other or with other capabilities.

These three extensions together enable a full reactive path from perception through memory access, reasoning, decisions and learning (and, hopefully, ultimately affect and motor control as well). In the process they yield a major expansion of what can occur in general via parallel reactive processing, in moving from Soar to Sigma, and thus which reactive supraarchitectural capabilities can be implemented and integrated together in a routine manner. Declarative – semantic and episodic – memory and learning provide compelling examples, but so do perceptual memory and learning – as in speech recognition and parameter learning – and imagery (e.g., mental map) memory and learning.

None of this implies that Sigma will not eventually need additional learning mechanisms – such as for acquiring new types of memory structures rather than just new instances of existing types – but it does imply that a suitably general reactive cycle can support much broader supraarchitectural capability integration than was previously thought.

Acknowledgments

This work has been sponsored by the U.S. Army. Statements and opinions expressed do not necessarily reflect the position or the policy of the United States Government, and no official endorsement should be inferred. Thanks are also due to John Laird for helpful comments on this article.

References

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C. & Qin, Y. (2004). An integrated theory of the mind. *Psychological Review*, *111*, 1036-1060.

Joshi, H., Rosenbloom, P. S. & Ustun, V. (2014). Isolated word recognition in the Sigma cognitive architecture. *Biologically Inspired Cognitive Architectures*. In press.

Koller, D. & Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. Cambridge, MA: MIT Press.

Laird, J. E. (2012). *The Soar Cognitive Architecture*. Cambridge, MA: MIT Press.

Laird, J. E., Newell, A. & Rosenbloom, P. S. (1987). Soar: An architecture for general intelligence. *Artificial Intelligence, 33*, 1-64.

Li, J. & Laird, J. (2015). Spontaneous retrieval from long-term memory for a cognitive architecture. *Proceedings of the 29th AAAI Conference on Artificial Intelligence.*

Rosenbloom, P. S. (2013). The Sigma cognitive architecture and system. *AISB Quarterly*, *136*, 4-13.

Rosenbloom, P. S., (2014). Deconstructing episodic learning and memory in Sigma. *Proceedings of the 36th Annual Conference of the Cognitive Science Society* (pp. 1317-1322). Quebec City, Canada.

Rosenbloom, P. S., Demski, A., Han, T. & Ustun, V. (2013). Learning via gradient descent in Sigma. *Proceedings of the 12th International Conference on Cognitive Modeling* (pp. 35-40). Ottawa, Canada.

Rosenbloom, P. S., Newell, A. & Laird, J. E. (1991). Towards the knowledge level in Soar: The role of the architecture in the use of knowledge. In K. VanLehn (Ed.), *Architectures for Intelligence* (pp. 75-111). Hillsdale, NJ: Erlbaum.

Populating ACT-R's Declarative Memory with Internet Statistics

Daniela Link (Daniela.Link@unil.ch) and Julian N. Marewski (Julian.Marewski@unil.ch)
Department of Organizational Behavior, UNIL-Internef
1015 Lausanne Switzerland

Keywords: memory; accessibility; ACT-R; simple heuristics; decision making

Introduction

Memory processes drawing on declarative knowledge play an important role in many cognitive models, for example in models of decision making. Within the fast-and-frugal heuristics research program, several strategies have been proposed that describe how people infer unknown criteria using knowledge associated with these criteria as *cues*. Much of the success of fast-and-frugal heuristics lies in their *ecological rationality*, or fit to regularities in the environment. Ecological rational decision strategies exploit regularities in the structure of the environment as they are reflected in basic cognitive capacities, such as memory. However, little research has looked at how environmental structures are mapped into mental representations. The ACT-R architecture offers a quantitative theory about how patterns of occurrences and co-occurrences of information in the environment are reflected in the memory activation of corresponding chunks. In this poster, we propose an ACT-R based ecological memory model representing objects and associated knowledge contingent on environmental frequencies of information encoded in the corresponding memory chunks. Based on internet statistics, we predict retrieval probabilities and retrieval latencies for associative knowledge, which will serve, for example, as input for simulating the selection and performance of knowledge-based decision strategies. A corresponding model could provide the missing link explaining how the interplay between the environment and the cognitive system promotes ecologically rational decision making.

Modeling Associative Memory in ACT-R

The basic unit of knowledge in ACT-R's declarative memory is the *chunk*. New declarative knowledge is added to memory by encoding representations of objects that are attended in the environment. A chunk can encode discrete elements of information as well as associations between elements being attended at the same time. The type of pattern encoded in the chunk is given in a *isa* slot, whereas other slots indicate the relationship between the elements of information that is being configured together. The knowledge that Berlin has an airport, for example, can be represented in a chunk with the following structure:

BERLIN-AIRPORT	
ISA	CITY_FACT
CITY	BERLIN
FACT	AIRPORT

In addition to symbolic information, each chunk encodes subsymbolic information about the likelihood that the chunk will be needed to reach one of the system's processing goals -the chunk's *activation*. The likely usefulness is a Bayesian estimate of posterior need odds derived from the past usefulness of the chunk (prior odds, or *history factor*) as well as from the current context (likelihood ratio, or *context factor*). ACT-R's theory of human associative memory offers a set of equations to calculate a chunk's activation from these two factors. Specifically, the activation, A_i, of a chunk i is determined by the base-level activation, B_i, plus the spreading activation the chunk receives from each of the j elements in the current context:

$$A_i = B_i + \sum_{j=1}^{m} W_j S_{ji} \qquad \text{(Activation)}.$$

Assuming approximately equal spacing of encounters of a chunk since its time of creation L, the base-level activation of a chunk can be approximated by (Anderson, 1993):

$$B_i = \ln n/(1-d) - d \ln L \qquad \text{(History Factor)},$$

where d is a decay parameter and n is the number of encounters of the object or relation encoded by the chunk.

In addition to the base-level activation which reflects the prior use of the chunk itself, a chunk receives spreading activation from related chunks currently attended in the current context. The amount of spreading activation a chunk receives depends on the associative strength, S_{ji}, between elements j stored in the buffers and chunk i as well as on the weight W_j given to each source of activation. The associative strength factor S_{ji}, can be calculated from environmental frequencies of occurrences and co-occurrences of chunk i and elements j according to the following equation (Schooler & Anderson, 1997):

$$S_{ji} = \ln \frac{P(i|j)}{P(i)} \qquad \text{(Context Factor)},$$

where $P(i|j)$ is an estimate of the probability of i occurring when j is present and $P(i)$ is the base rate of i occurring. Source activation is typically divided equally among the number of sources of activation, m, and sums to a constant, W, which implies that

$$W_j = W/m \qquad \text{(AttentionWeighting)},$$

In ACT-R, only chunks that exceed a certain amount of activation A_i, as defined by the retrieval threshold, τ, can be retrieved. Because of the stochastic volatility in momentary activation levels, chunks exceed this threshold with a certain probability. The retrieval probability, p, for chunk i, is a logistic function of the chunk's activation:

$$p_i = \frac{1}{1+e^{\frac{-(A_i-\tau)}{s}}} \qquad \text{(Retrieval Probability)},$$

where s is a scale parameter representing noise in the retrieval process. Given a chunk is successfully retrieved, the retrieval time can be expressed as an exponential function of the chunk's activation:

$$T_i = Fe^{-A_i} \quad \text{(Retrieval Time).}$$

The above equations describe how patterns of occurrences and co-occurrences of objects in the environment are reflected in subsymbolic properties related to the activations and associative strengths of chunks in ACT-R's declarative memory. Importantly, the chunk's activations, in turn, allow for behavioral predictions about retrieval probabilities and retrieval times for the corresponding memories. We use observed retrieval probabilities and retrieval time distributions for knowledge about cities to calibrate our memory model for chunks encoding associative knowledge about these cities. We then use our model to predict people's knowledge of and retrieval speed for cities and associated facts from frequency statistics obtained from the internet.

Behavioral Data

One-hundred twenty-eight students (54 female; mean age 20 years, SD = 2.23) took part in the experiment. Participants received a fixed payment of 5 CHF (5.39 US$) supplemented by a performance bonus of up to 33 CHF (35.56 US$) depending on the coherence of their responses in the main task with responses given in a later control task where similar knowledge was tested. The stimuli for the cue-knowledge task (see Figure 1) consisted of 95 European cities and eight cues. Cue-knowledge tested was whether the city had an *airport*, a *university*, a premier league *soccer* team, the headquarters of a *company* listed on the stock market, a *cathedral*, a *subway*, a *harbor*, and whether it was served by a high-speed *train* line. Each city was paired with each of the cues, so that the items consisted of a total of 760 city-cue pairs.

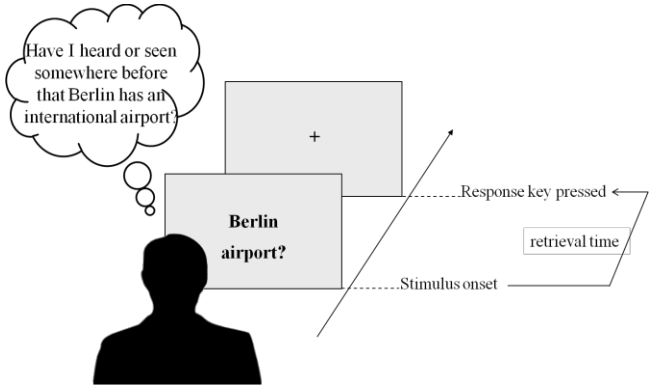


Figure 1: Illustration of the cue-knowledge task.

Participants were presented with city-attribute pairs one at a time in and were asked to respond with either "yes" if they could remember having seen or heard of the city possessing such an attribute or "no" if they could not remember having heard of this before. Responses were made by pressing keys on the right and left side of the keyboard. The order of presentation of items was randomized. All trials were preceded by a small fixation cross for 1,000 msec and participants were instructed to fixate the cross until it disappeared and to respond as quickly and accurately as possible upon stimulus onset.

Predicting Accessibility from Internet Statistics

We approximate memory activation A_i resulting from encounters with certain information in a person's environment by the activation $A_{i,web}$ estimated from *web counts*, the number of entries for this information in the knowledge base Wikipedia.

$$A_i = c + b\, A_{i,web}.$$

The parameters c and b serve as scaling parameters describing the unknown relation between how often we encounter an object in our environment and the web frequency of the corresponding search term.

We calibrated the memory model to the log odds of retrieval of cue-knowledge. Subsequently, we calibrated the model to the observed retrieval times for retrieved cue-knowledge. Activations for chunks encoding cue-knowledge estimated from web counts were then used to predict observed retrieval probabilities and retrieval time distributions.

Conclusion

Comparisons between observed and predicted retrieval probabilities, and observed and predicted retrieval time distributions show that our memory model is able to capture how the probability of retrieval and the accessibility of cue-knowledge depends on the distribution of relevant information in the environment.

Our work extends the ecological approach for populating the contents of declarative memory in ACT-R (e.g., Marewski & Schooler, 2011). Possible applications include the simulation of performance of and selection between knowledge-based inference strategies and could be used for any model interested in mirroring the statistical structure of the environment outside the laboratory.

Acknowledgments

This work was supported by a grant from the Swiss National Science Foundation [Grant number 100014_144413] to Julian Marewski.

References

Anderson, J. R. (1993). *Rules of the mind*. Hillsdale, NJ: Erlbaum.

Marewski, J. N., & Schooler, L. J. (2011). Cognitive niches: an ecological model of strategy selection. *Psychological Review*, *118*, 393.

Schooler, L. J., & Anderson, J. R. (1997). The role of process in the rational analysis of memory. *Cognitive Psychology, 32,* 219–250.

Tracking memory processes during ambiguous symptom processing in sequential diagnostic reasoning

Agnes Scholz (agnes.scholz@psychologie.tu-chemnitz.de)[a]

Josef F. Krems (josef.krems@psychologie.tu-chemnitz.de)[a]

Georg Jahn (jahn@imis.uni-luebeck.de)[b]

[a]Cognitive and Engineering Psychology, Wilhelm-Raabe-Str. 43, 09120 Chemnitz, Germany

[b]Institute for Multimedia and Interactive Systems, Ratzeburger Allee 160, 23562 Lübeck, Germany

Keywords: eye movements; process tracing; memory indexing; diagnostic reasoning; belief updating

In sequential diagnostic reasoning multiple pieces of information have to be combined to find a best explanation for observed symptoms (e.g., Johnson & Krems, 2001). Tracking memory processes involved in reasoning proves difficult because they proceed without accompanying actions towards the environment. However, this is important in order to build and test cognitive models (Schulte-Mecklenbeck, Kühberger, & Ranyard, 2011). Memory indexing is a novel method to study the time course of information processing in memory during reasoning and decision making (Jahn & Braatz, 2014; Renkewitz & Jahn, 2012) by recording eye movements. The basic principle underlying memory indexing is that people look at an emptied spatial location when retrieving information that has been associated with the spatial location during encoding (e.g., Richardson & Spivey, 2000). We use memory indexing to reveal memory dynamics in sequential diagnostic reasoning in order to test process assumptions derived from cognitive models on reasoning and belief updating.

We study sequences of symptoms, for which more than one diagnosis is possible. Reasoners strive for a coherent interpretation of symptoms (Kostopoulou, Russo, Keenan, Delaney, & Douiri, 2012). When two diagnoses compete, coherence can be achieved by *biased interpretation of symptoms* that increases the belief in one hypothesis while decreasing the belief in alternatives (Holyoak & Simon, 1999; Mehlhorn & Jahn, 2009). Maximizing coherence often favors the initially leading hypothesis. But it can strengthen an alternative when stronger evidence for an alternative hypothesis has accumulated. Then, a *hypothesis change* takes place.

In this study, we test process assumptions of coherence maximization, i.e. biased symptom processing towards the leading hypothesis or hypothesis change, by applying memory indexing and presenting participants with ambiguous symptom sequences, for which coherence maximization over the course of symptom processing can favor one or the other diagnosis. The biases in symptom processing preventing or inducing a hypothesis change should be revealed by participants' gaze behavior.

Method

The study consisted of a learning phase and a subsequent reasoning phase. During the learning phase, participants acquired the knowledge needed for the reasoning phase. The reasoning task was to determine the most likely cause of a patient's symptoms. The patients were workers in a chemical plant that produces four chemicals and each worker was affected by exactly one of those chemicals (Mehlhorn, Taatgen, Lebiere, & Krems, 2011).

Participants

Thirty-two students (21 female, M_{age} = 22.4, range: 19-39 years) from Technische Universität Chemnitz participated in the study.

Apparatus and Material

An SMI RED remote eye tracker sampled data of the right eye at 120 Hz in a laboratory setting.

Four chemicals were assigned to screen quadrants (Fig. 1). Each quadrant enclosed three rectangular frames, which contained

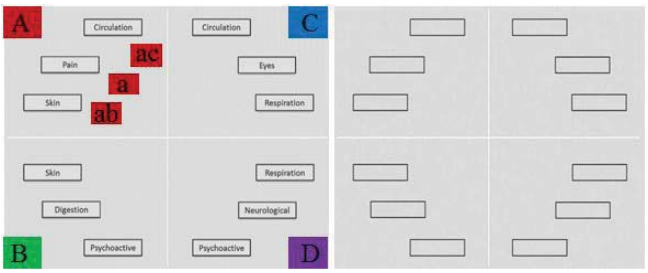


Figure 1: Left: Spatial arrangement of chemicals (A, B, C, D) and symptom classes (e.g., a, ac, ab) as presented during learning. Right: Emptied spatial arrangement during the reasoning phase.

three symptom classes that the respective chemical could cause. For example, the chemical at the top left caused symptoms from the symptom classes circulation, pain, and skin. One symptom class was unique (pain for the top left chemical, denoted with single small letters, e.g., a) and two symptom classes were shared with other chemicals (denoted with two small letters, e.g., ab).

Procedure

In the learning phase, participants first learned how symptoms were assigned to symptom classes and second how symptom

classes related to chemicals. Associations between symptom classes and chemicals were established by presenting symptom classes in rectangular frames in the screen quadrants that each represented one chemical (Fig. 1, left side). During reasoning, symptoms were presented auditorily while participants only saw the emptied rectangular frames (Fig. 1, right side). Eye movements were recorded throughout the reasoning phase. The diagnostic decision was collected at the end of the reasoning trial.

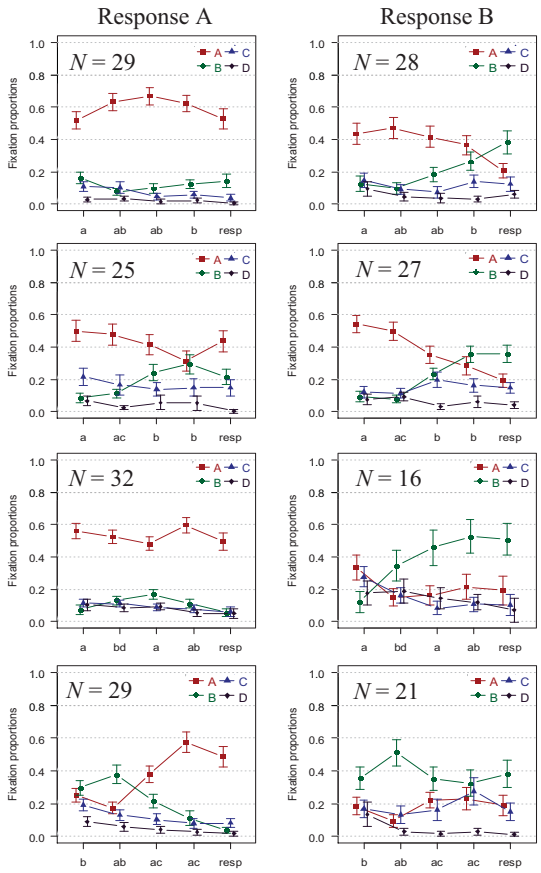


Figure 2: Mean proportion of fixation times in each interval that fell upon the A-, B-, C-, or D-quadrants for four ambiguous symptom sequences and A- and B-responses. Error bars represent standard errors.

Results and Discussion

For the analyses of eye movements, four areas of interest (AOIs) were defined corresponding to the four quadrants representing the four chemicals. The AOIs were denoted A, B, C, and D according to the four chemical roles. Each trial was divided in five time intervals defined by the onsets of each of the four symptom presentations and the response interval. For each of the five intervals and each AOI, we computed the proportion of total fixation time in the four AOIs separately for each symptom sequence, each participant, and by diagnostic response.

Fig. 2 shows plots of mean fixation proportions of four exemplary symptom sequences. There are separate plots for trials with A-, and B-responses. The sequences in Fig. 2 are ordered from top to bottom according to the number of consecutive symptoms that supported the A-hypothesis from the beginning of the sequence onward.

Fixation proportions after the *first symptom presentation* reflected which hypothesis was supported. During *subsequent symptom intervals* fixation proportions increased towards the most likely hypothesis given the subjective interpretation of symptoms in the symptom sequence. After strong evidence for an alternative hypothesis had accumulated, fixation proportions revealed a hypothesis change for B-responses in sequences starting with an a-symptom and for A-responses in the sequence starting with a B-symptom. Fixation proportions during the *response interval* reflected which hypothesis was chosen.

Eye movements as revealed by applying memory indexing to the study of sequential diagnostic reasoning of ambiguous symptom sequences reflect the reasoners' tendency to strive for coherence in interpreting new information. Studying eye movement behavior will inform existing computational models on reasoning and decision making and enhance the understanding even of memory-based reasoning processes.

Acknowledgments

This research was supported by German Research Foundation (DFG) grants KR 1057/17-1 and JA 1761/7-1. The authors would like to thank Ricarda Fröde and Claudia Dietzel for their help in conducting the experiment.

References

Holyoak, K., & Simon, D. (1999). Bidirectional reasoning in decision making by constraint satisfaction. *Journal of Experimental Psychology: General*, *31*, 3–31.

Jahn, G., & Braatz, J. (2014). Memory indexing of sequential symptom processing in diagnostic reasoning. *Cognitive Psychology, 68,* 59–97.

Johnson, T. R., & Krems, J. F. (2001). Use of current explanations in multicausal abductive reasoning. *Cognitive Science, 25*, 903-939.

Kostopoulou, O., Russo, J. E., Keenan, G., Delaney, B. C., & Douiri, A. (2012). Information distortion in physicians' diagnostic judgments. *Medical Decision Making, 32*, 831–839.

Mehlhorn, K., & Jahn, G. (2009). Modeling sequential information integration with parallel constraint satisfaction. *Proceedings of the 31st Annual Conference of the Cognitive Science Society*, 2469- 2474.

Mehlhorn, K., Taatgen, N. A., Lebiere, C., & Krems, J. F. (2011). Memory activation and the availability of explanations in sequential diagnostic reasoning. *Journal of Experimental Psychology: Learning, Memory and Cognition, 37*, 1391-1411.

Renkewitz, F., & Jahn, G. (2012). Memory Indexing: A novel method for tracing memory processes in complex cognitive tasks. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *38*, 1622–1639.

Richardson, D. C., & Spivey, M. J. (2000). Representation, space and Hollywood Squares: Looking at things that aren't there anymore. *Cognition*, *76*, 269–295.

Schulte-Mecklenbeck, M., Kühberger, A., & Ranyard, R. (2011). The role of process data in the development and testing of process models of judgment and decision making. *Judgment and Decision Making*, *6*(8), 733–739.

Mathematical modeling of cognitive learning and memory

Vipin Srivastava[1,2] **(vpssp@uohyd.ernet.in)** & **Suchitra Sampath**[1] **(cc09pc01@uohyd.ernet.in)**
[1]Centre for Neural and Cognitive Sciences & [2]School of Physics, University of Hyderabad, India.

Abstract

We demonstrate how basic cognitive functions of learning and memory can be modeled mathematically and how such models are first built from a bare minimum of essential information and then developed systematically in a step by step manner to include more and more realistic features.

Keywords: Cognitive learning and memory; Hopfield model; orthogonalization; attractor neural network; LTP; LTD; spin glass

Introduction

Learning and memory are amongst the basic cognitive attributes of our brain, yet we are only beginning to understand the physiological mechanisms underlying them. Whatever little success we have achieved in recent years in this direction can be attributed, to some extent, to mathematical and computational modelling of these and related phenomena. We offer a glimpse of how one approaches this problem through mathematical modelling.

Developing mathematical models

Learning and memory are related in that we first learn, and, if what we learn stays in the brain and can be recalled then we say that we are able to memorise. A major break in understanding "learning" was given by Donald Hebb in 1949 (Hebb, 1949), who pointed out that the synapse connecting the neurons are plastic in nature and that their strength can change in an irreversible manner. These changes, termed as long-term potentiation (LTP) and long-term depression (LTD), are manifestations of 'learning'. Leon Cooper (Cooper, 1973) cast the Hebbian hypothesis in the following mathematical form,

$$J_{ij} = \frac{1}{N}\sum_{\mu=1}^{p} \xi_i^{(\mu)}\xi_j^{(\mu)}, \tag{1}$$

where, in a fully connected network of N neurons, J_{ij} represent the synaptic strengths between neurons i and j, and $\xi_i{}^{(\mu)}$ represents the activity of the i^{th} neuron, which is taken as 1 if the neuron fires and -1 otherwise; μ is index for a pattern/vector of ± 1s. The J_{ij} thus depends on the activities of the neurons that are connected by it as was hypothesized by Hebb (Hebb, 1949). It changes cumulatively as new patterns μ are presented successively to the network.

John Hopfield (Hopfield, 1982) used mathematical framework of a physics system called 'spin glass' (Edwards & Anderson, 1975) and incorporated this prescription for learning in a simple model of firing/not firing (i.e. ± 1) neurons to account for numerous memories that the brain can accommodate at the same time. In particular, it helped us understand 'content addressability' or 'associative recall' in which if the network encounters a pattern that is similar to but not the same as an imprinted pattern, then it can associate the new pattern with the imprinted one. This accounts for a common feature of cognitive memory in which we can identify a familiar (or memorized) object from its partial, or obscured, or noisy appearance. In fact we can imagine around each imprinted pattern a collection of patterns that bear similarity with the imprinted pattern in varying degrees. This region in the configuration space is called "basin of attraction" and the Hopfield network is called attractor neural network (ANN). A noisy version of an imprinted pattern falling within a certain range around it, if presented to the ANN, will by and by converge to the imprinted pattern following the retrieval/recall prescription,

$$h_i^{(\nu)} = \frac{1}{2}\sum_{\substack{j=1 \\ j\neq i}}^{N} J_{ij}\xi_j^{(\nu)}, \tag{2}$$

where $h_i^{(\nu)}$ is the local field (or post-synaptic potential) on neuron i due to activities on all the other $(N-1)$ neurons (in an arbitrary pattern ν) projecting onto i via J_{ij}'s. If ν^{th} pattern is not one of the the imprinted patterns then the condition,

$$h_i^{(\nu)}\xi_i^{(\nu)} > 0 \tag{3}$$

will not be met for all i's. In that case $\{1\ \mathrm{sgn}(h_i^{(\nu)})\}$ are fed on the right hand side of equation (2) as $\{\xi_i^{(\nu)}\}$ and condition (3) is checked with the new set $\{h_i^{(\nu)}\}$. After a few iterations the ν^{th} pattern converges to the imprinted pattern in whose basin of attraction the ν^{th} pattern happens to fall. In physics terms this means that the imprinted pattern, say μ, corresponds to a minimum of the following total energy function (or Hamiltonian),

$$H = \frac{1}{2}\sum_{\substack{j=1 \\ (j\neq i)}}^{N} J_{ij}\xi_i^{(\mu)}\xi_j^{(\mu)}, \tag{4}$$

This energy function is akin to that of spin-glass (Edwards & Anderson, 1975). What makes it useful as a model for memory is that it has an exponentially large number of minima, which correspond to different configurations of up and down spins or ± 1, being fed in through (1).

Random sets of $\{\xi_i^{(\mu)}\}$ minimize H as long as the number of imprinted patterns does not exceed a critical limit (Amit, Gutfreund, & Sompolinsky, 1985). As new patterns are imprinted according to (1) noise builds up in the system and beyond a stage $(p/N > 0.14)$ the noise submerges the signals and we end up in a situation where none of the imprinted patterns is retrieved. This catastrophic loss of memory is cognitively unrealistic.

In figure (1) we show a simulation of the Hopfield model. It shows the variation of the number of patterns that are retrieved with 100% accuracy as a function of the number of stored patterns, p normalised by N. Note that beyond p/N=0.1, the fraction of stored patterns that are retrieved accurately begins to reduce, and drops rather steeply for p/N >0.15. Close to p/N=0.3 hardly any of the stored patterns is retrieved. This marks the memory catastrophe.

Going beyond, with corrections

To remove the above hurdle we have improved the Hopfield model to eliminate the noise from the system, which is produced by "cross-talks" between the imprinted patterns. Our hypothesis is that when an information comes to be recorded, it is first "orthogonalized" with respect to all the information in the memory, and then the orthogonalized version is stored in the memory following the Hebbian hypothesis (1). Orthogonalization is a mathematical transformation that converts a set of vectors into a mutually perpendicular set. Orthogonalization amounts to identifying similarities and differences that the new pattern may have with all those in the memory and then storing these similarities and differences in the synapses. While the mathematical details can be found in (Srivastava & Edwards, 2000), we will highlight here a curiously interesting aspect of our hypothesis.

Suppose a set of vectors $\{\xi^{(\mu)}\}$ is to be stored in the Hopfield like neural network. In the orthogonalization hypothesis $\vec{\xi}^{(\mu)}$'s will be orthogonalized sequentially (for μ=1,2,3...p) following Gram-Schmidt's procedure (Srivastava & Edwards, 2000). This will give us a set $\{\vec{\eta}^{(\mu)}\}$, which will be inscribed/stored in the network following (1) using $\{\eta^{(\mu)}\}$ instead of $\{\xi^{(\mu)}\}$. However, we find that we can study the retrieval, or recall, of the original vectors $\vec{\xi}^{(\mu)}$'s from the network. Most significantly N $\vec{\xi}^{(\mu)}$'s in a network of N are retrieved efficiently, i.e. with 100% accuracy in a single iteration of prescription (2).

The red plot in figure (1) displays that the fraction of p stored patterns that can be retrieved perfectly stays at 1 for all values of p upto p=N when the orthogonalized versions of the given p patterns are stored. The orthogonalization scheme gives new insight into the basins of attraction of $\vec{\xi}^{(\mu)}$'s and their stability conditions (Sampath & Srivastava, manuscript in preparation).

In sum we have shown that mathematical modeling plays a crucial role in understanding the mechanisms of cognitive functions. Such models not only provide quantitative results which can be substantiated by experiments, but also have almost indefinite scope for improvement and generalization to include new parameters, and relax approximations and simplifying assumptions to make the model more and more biologically realistic. In the present model, for instance, we need to (a) dilute the connectivities between neurons (a neuron is typically connected to 15% or less of other neurons), (b) take into account the fact that J_{ij} need not be equal to J_{ji}, and (c) treat a multinary neuron rather than binary to account for the fact that a neuron may fire at different rates, etc. Also, an incoming new information need not be orthogonalized with respect to all the previously stored information; it should be orthogonalized with respect to a selected set of old and stored information - i.e., the memory ought to have a hierarchical structure. Moreover, we should generalize the model to go beyond sequential learning symbolized by Gram-Schmidt orthogonalization and include, e.g. episodal memories.

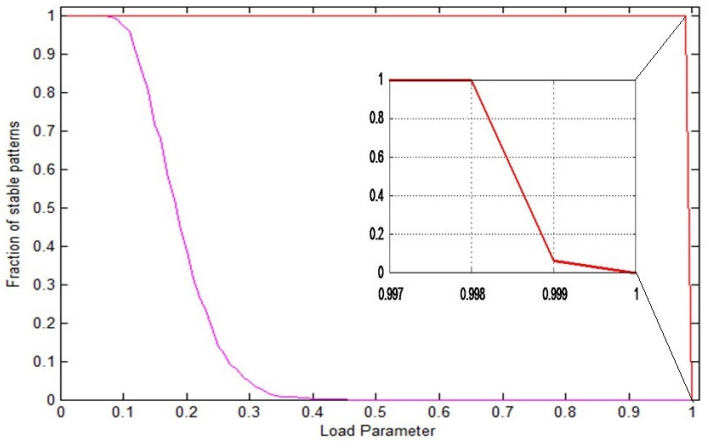


Figure 1: Fraction of perfect retrieval vs load parameter (p/N), in conventional Hopfield model (magenta) and after introducing orthogonalization for learning (red), (N=1000) .

Acknowledgements : This work is supported by the Royal Society (London, UK) and the cognitive science research initiative of the Department of Science and Technology, Government of India.

References

Amit, D., Gutfreund, H., & Sompolinsky, H. (1985). Storing infinite numbers of patterns in a spin-glass model of neural networks. *Physical Review Letters*, *55*, 1530–1533.

Cooper, L. (1973). A possible organization of animal memory and learning. In *Nobel symposium on collective properties of physical systems* (pp. 62–84). Aspensagaerden, Sweden: The Nobel Foundation.

Edwards, S., & Anderson, P. (1975). Theory of spin glasses. *Journal of Physics F: Metal Physics*, *5*, 965–974.

Hebb, D. (1949). *Organization of behaviour*. New York: Wiley.

Hopfield, J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, *79*, 25542558.

Sampath, S., & Srivastava, V. (manuscript in preparation). New results from basins of attraction in attractor neural networks.

Srivastava, V., & Edwards, S. (2000). A model of how the brain discriminates and categorises. *Physica A: Statistical Mechanics and its Applications*, *276*, 352–358.

Modeling Choices at the Individual Level in Decisions from Experience

Neha Sharma (neha724@gmail.com)
Applied Cognitive Science Laboratory, Indian Institute of Technology, Mandi, India

Varun Dutt (varun@iitmandi.ac.in)
Applied Cognitive Science Laboratory, Indian Institute of Technology, Mandi, India

Abstract

Decisions from Experience (DFE) involve situations where decision makers sample information before making a final choice. Trying clothes before choosing a garment and enquiring about jobs before opting for one are some examples involving such situations. In DFE research, conventionally, the final choice that is made after sampling information is aggregated over all participants and problems in a given dataset. However, this aggregation does not explain the individual choices made by participants. In this paper, we test the ability of computational models of aggregate choice to explain choices at the individual level. Top three DFE models of aggregate choices are evaluated on how these models account for individual choices using the maximization criterion. A Primed-Sampler (PS) model, a Natural-Mean Heuristic (NMH) model, and an Instance-Based Learning (IBL) model are calibrated to explain individual choices (maximizing or non-maximizing) in the Technion Prediction Tournament (the largest publically available DFE dataset) and the generalization SC Problems dataset. Our results reveal that all the three DFE models of aggregate choices perform average to explain individual choices. Although the IBL model performs slightly better than PS and NMH models; all the three models are able to account for all individuals in both the calibration and generalization datasets. We conclude by drawing implications for computational cognitive models in explaining individual choices in DFE research.

Keywords: Aggregate choice, individual choice, experience, sampling, computational models, maximization.

Introduction

The steep rise in number of smartphones has given an ample choice to consumers to experiment with (Emarketer, 2014). As a result, a customer now has the privilege of choosing between a wide range of smartphones. To buy the best, one must sample information about the various handsets before making one final choice for one's preferred smartphone. The act of making choices based upon sampled information, however, is not limited to choosing between smartphones rather, it is a very common exercise involving people in different facets of their daily life (choosing clothes, choosing jobs etc.). In fact, information search before a choice constitutes an integral part of Decisions from Experience (DFE) research, where the focus is on explaining human maximizing decisions based upon one's experience with sampled information (Hertwig & Erev, 2009).

In order to study people's search and choice behaviors in the laboratory, DFE research has proposed a "sampling paradigm" (Hertwig & Erev, 2009). In the sampling paradigm, people are presented with two or more options to choose between. These options are represented as blank buttons on a computer screen. People are first asked to sample as many outcomes as they wish from different button options (information search). Once people are satisfied with their sampling of the options, they decide from which option to make a single final choice for real.

Computational cognitive models of human choice behavior have thus far predicted choices at an aggregate level in the sampling paradigm, i.e., when people's final choices are averaged over a large number of participants models (Busemeyer & Wang, 2000; Gonzalez & Dutt, 2012; Lejarraga, Dutt, & Gonzalez, 2012). For example, the Primed-Sampler (PS) model, the Natural-Mean Heuristic (NMH) model, and the Instance-Based Learning (IBL) model are popular DFE algorithms for explaining aggregate choices (Erev et al., 2010; Gonzalez & Dutt, 2011). The PS model depends upon the recency of sampled information, where the model looks back a few samples on each option before making a final choice (Gonzalez & Dutt, 2011). On the other hand, the NMH model is a generic case of the PS model. In this model, one calculates the natural mean of outcomes observed on each sampled option, and using the same for making a final choice (Hertwig, 2011). Similarly, the IBL model (Gonzalez & Dutt, 2011) consists of experiences (called instances) stored in memory. Each instance's activation is a function of the frequency and recency of the corresponding outcomes observed during sampling in different options. These activations are used to calculate the blended values for each option, thereby helping the model make a final choice. The IBL model rely on ACT-R framework for its functioning (Anderson & Lebiere, 1998).

Prior DFE research has shown that, at the aggregate level, the PS, NMH, and IBL models exhibit superior performance compared to other computational models in the sampling paradigm (Erev et al., 2010; Gonzalez & Dutt, 2011). For evaluating these models at the aggregate level, a comparison is made between a model's data and human data from the Technion Prediction Tournament (TPT) dataset (TPT being the largest publically known DFE dataset). However, up to now, none of the three DFE models have been evaluated in their ability to account for maximizing choice behavior at the individual participant level (i.e., in explaining the maximizing final choice of each human participant playing a problem in a dataset). If these models are able to account for maximizing choices at the aggregate level, then one expects that they might also be able to account for maximizing choices at the individual level. However, given that there are sources of noise in both the sampling data as well as in these

models, it is likely that these models are no better than random chance in explaining choices at the individual level. Furthermore, as models developed at the aggregate level have different number of free parameters, it is important to use generalization as a test of a model's ability to account for data at the individual level (Busemeyer & Wang, 2000).

In this paper, our main goal is to evaluate how models, which explain choice behavior at the aggregate level, perform at the individual level. In order to evaluate models at the level of an individual participant, we use the largest publically available TPT dataset in the sampling paradigm (Erev et al., 2010) to calibrate the models. Next, we use the SC Problems six-problem dataset (Hertwig et al., 2004) to generalize the calibrated models and test them at the individual level. In what follows, we detail the dataset used and the working of the three models described above. Then, we discuss the methodology of calibrating these models at the individual participant level so as to capture the maximizing final choice. Next, we present the results of models' evaluation at the individual level both during calibration and during generalization. Finally, we close the paper by discussing the implications of our results for models of aggregate choice.

The Technion Calibration Dataset

The Technion Prediction Tournament (TPT) (Erev et al., 2010) was a competition in which several participants were subjected to an experimental setup, the "e-sampling condition." In this condition, participants sampled the two blank button options in a binary-choice problem before making a final choice for one of the options. During sampling, participants were free to click both button options one-by-one and observe the resulting outcome. Participants were asked to press the "choice stage" key when they felt that they had sampled enough (but not before sampling at least once from each option). The outcome of each sample was determined by the structure of the relevant problem. One option corresponded to a safe choice: Each sample provided a medium (M) outcome. The other option corresponded to the payoff distribution of a risky choice: Each sample provided a High (H) payoff with some probability (pH) or a low (L) payoff with the complementary probability (1 - pH). At the choice stage, participants were asked to select once between the two options. Their choice yielded a random draw of one outcome from the selected option and this outcome was considered at the end of the experiment to determine the final payoff. Competing models submitted to TPT were evaluated following the generalization criterion method (Busemeyer &Wang, 2000), by which models were fitted to choices made by participants in 60 problems (the estimation set) and later tested in a new set of 60 problems (the test set) with the parameters obtained in the estimation set. The 120 problems consisted of choice between a safe option and a risky option as described above. The M, H, pH, and L in a problem were generated randomly, and a selection algorithm was used so that the 60 problems in each set differed in its M, H, pH, and L from other problems. In all the models described here, we have considered an individual human or model participant playing a problem in a dataset (competition or estimation) as an "observation." Also, all model parameters have been calibrated by using the entire TPT dataset that consisted of 120 problems and 2,370 observations. For more details about the TPT, please refer to Erev et al. (2010).

In this section, we detail the working of three popular DFE models that have been used to evaluate human choices at the aggregate level.

Prime Sampler (PS) Model

The PS model (Hertwig, 2011) employs a simple choice rule. In this model, participants are expected to take a sample of k draws from each option. The exact value of k differs between observations (an observation is defined as a participant playing a problem in a dataset). The PS model assumes that the exact value of k for an observation is uniformly drawn as an integer between 1 and N, where N is a free parameter that is calibrated in the model. The final choice for each observation is determined by the following choice probability:

$$Prob\ (Option\ X) = EXP(S_{Mean}X)/(EXP(S_{Mean}X) + EXP(S_{Mean}Y)) \quad \ldots (1)$$

Where, $S_{Mean}X$ and $S_{Mean}Y$ are the samples means of the two options and $Prob\ (Option\ X)$ is the probability of choosing $Option\ X$. For each model observation, the $Prob\ (Option\ X)$ is compared with a random number U (0, 1) to make a choice for one of the two options. If the value of random number is less than or equal to $Prob\ (Option\ X)$, then a choice is made for *Option X*. According to literature, the PS model has performed very accurately at predicting aggregated human choices in the sampling paradigm (Erev, Glozman, & Hertwig, 2008).

Natural Mean Heuristic (NMH) Model

The NMH model (Hertwig & Pleskac, 2010) involves the following steps:

Step 1. Calculate the natural mean of observed outcomes for each option by summing, separately for each option, all n experienced outcomes and then dividing by n.

Step 2. Apply equation 1, where the sample mean is replaced by natural mean.

Thus, the NMH model is a special case of the PS model, where k = an observation's sample size. There are no free parameters in the NMH model. Like the PS model, this model has also performed very accurately at predicting aggregated human choices in the sampling paradigm (Hertwig & Pleskac, 2010).

Instance Based Learning (IBL) Model

The IBL model is based upon the ACT-R framework (Gonzalez & Dutt, 2011; 2012) and this model is known to predict human aggregate choices better than several DFE

models that include assumptions similar to those made in the PS and NMH models (Gonzalez & Dutt, 2011). In this model, every occurrence of an outcome on an option is stored in the form of an *instance* in memory. An instance is made up of the following structure: SDU, here S is the current situation (a number of blank option buttons on a computer screen), D is the decision made in the current situation (choice for one of the option buttons), and U is the goodness (utility) of the made decision (the outcome obtained upon making a choice). When a decision choice needs to be made, instances belonging to each option are retrieved from memory and blended together. Blended values are a function of activation of instances being blended. Activation is a function of the frequency and recency of observed outcomes that occur on choosing options during sampling. In binary choice, the IBL model chooses one of two options by selecting the one having a value greater than a random variable (Gonzalez & Dutt, 2011; 2012). The blended value of option *j* (e.g., a gamble that pays \$4 with .8 probability or \$0) at any trial *t* is defined as

$$V_{j,t} = \sum_{i=1}^{n} p_{i,t} x_{i,t} \qquad \cdots (2)$$

where $x_{i,t}$ is the value of the U part of an instance *i* (e.g., either \$4 or \$0, in the previous example) at trial *t* and $p_{i,t}$ is the probability of retrieval of that instance from memory at the same trial [10]. Because $x_{i,t}$ is the values of the U part of an instance *I* at trial *t*, the number of terms in the summation changes when new outcomes are observed within an option *j* (and new instances corresponding to observed outcomes are created in memory). Thus, *n*=1 if *j* is a safe option with one possible outcome. If *j* is a risky option with two possible outcomes, then *n*=1 when one of the outcomes has been observed on an option (i.e., one instance is created in memory) and *n*=2 when both outcomes have been observed (i.e., two instances are created in memory).

At any trial *t*, the probability of retrieval of an instance *i* is a function of the activation of that instance relative to the activation of all instances created within that option, given by

$$p_{i,t} = \frac{e^{A_{i,t}/\tau}}{\sum_j e^{A_{j,t}/\tau}} \qquad \cdots (3)$$

Where τ, is random noise defined as $=\sigma.\sqrt{2}$ and σ is a free noise parameter. Noise in Equation (2) captures the imprecision of recalling past experiences from memory. The activation of an instance corresponding to an observed outcome in a given trial is a function of the frequency of the outcome's past occurrences and the recency of the outcome's past occurrences (as done in ACT-R). At each trial *t*, activation A of an instance *i* is

$$A_{i,t} = \sigma \ln\left(\frac{1-\gamma_{i,t}}{\gamma_{i,t}}\right) + \ln \sum_{t_i \varepsilon \{1,\ldots,t-1\}}^{n} (t-t_i)^{-d} \quad \ldots (4)$$

where *d* is a free decay parameter; $\gamma_{i,t}$ is a random draw from a uniform distribution bounded between 0 and 1; and t_i is each of the previous trial indexes in which the outcome corresponding to instance *i* was observed. The IBL model has two free parameters that need to be calibrated: *d* and σ. The *d* parameter controls the reliance on recent or distant sampled information. Thus, when *d* is large (> 1.0), then the model gives more weight to recently observed outcomes in computing instance activations compared to when *d* is small (< 1.0). The σ parameter helps to account for the sample-to-sample variability in an instance's activation. For each model observation, the model applies equation 1 to make a choice for one of the two options (for this purpose, the sample mean is replaced by blended values, $V_{j,t}$ for each option).

The Coin Toss (CT) Model

The CT model is used as a baseline model and it represents chance performance. In this model, we compare the value of a random number between [0, 1] with probability = 0.5. In a binary-choice task, if the random number value < 0.5, then the model chooses the final choice as one option; otherwise, the model chooses a final choice as the other option. When simulated, for a binary-choice task, this model is expected to produce close to 50% accuracy in explaining participants' individual choices. As the probability is fixed at 0.5, this model contains no free parameters.

Method

Model Execution

Models submitted to the TPT were evaluated only according to their ability to account for aggregate choice behavior (i.e., the proportion of choices for the option with H and L outcomes were aggregated across participants and problems) (Erev et al., 2010). In this paper, we account for the choice at the individual participant level. For this purpose, a choice made by a model observation is evaluated against a choice made by a corresponding human observation. In order to compare human and model choices for each observation, we evaluate an "error ratio" (i.e., the ratio of incorrectly classified final choices between model and human observations divided by the total number of observations). Firstly, for each observation in human data, we determine the final choice whether maximizing or non-maximizing. In the TPT dataset, a choice is classified as maximizing if the expected value of an option (based upon given problem) with high or low outcome is greater than expected value of an option with medium outcome. Those cases for which the aforementioned criteria fails are termed as having non maximizing choice. Furthermore, human final choice is then compared with the theoretical maximizing or non-maximizing choices to obtain maximizing human final choice per observation. A similar final maximizing choice is then derived for a model observation and this derived choice is compared to the maximizing choice made by the corresponding human observation. The final choices from each of the three models are compared to 2,370 human

observations, i.e., the total number of participant-problems-set combinations in TPT dataset. For a model, the error-ratio is calculated as:

Error Ratio = $(M_HN_M + N_HM_M)/ (M_HN_M + N_HM_M + N_HN_M + M_HM_M)$ ⋯ (5)

Where, M_HN_M was the number of observations where the human made a maximizing choice but the model predicted a non-maximizing choice. N_HM_M was the number of observations where human made a non-maximizing choice but the model predicted a maximizing choice. Similarly, the M_HM_M and N_HN_M were the number of observations, where a human observation made the same choice (maximizing or non-maximizing) as predicted by the model. The smaller the value of the error ratio, the more accurate is a model in accounting for maximizing individual choices of human participants. For some observations, a model could be equally likely to choose either of the options. Such cases were discarded from the error ratio calculation and were termed as uncategorized (UN) cases, separately. Thus, more are the number of UN cases, the poorer is the corresponding model's algorithm in accounting for complete human data.

In the PS model, an integer value was drawn between 1 and *N*. For the purposes of model calibration, a maximum value of *N* was assumed to be 216. This choice of maximum value was justifiable as 216 is the maximum sample size in the TPT dataset (Erev et al., 2010). For each new value of *N* from 1 to 216, the PS model was run 5 times over the set of 2,370 observations (i.e., for each run, 2,370 observations were used in the model). Error ratio was computed for each of the 5 runs and these five 5 ratios were then averaged for calculating the average error ratio. The *N* value for which the average error ratio was minimized was taken as the calibrated *N* value. The choice of 5 runs of the model is because it enabled us to account for the randomness present in the model due to equation 1.

The NMH model did not possess any free parameters. The model's calibration involved only the natural mean computation for each observation based on the outcomes observed on both the op options during sampling. In NMH model, the final maximizing choice was made for an option based upon equation 1. This model was run 5 times across 2,370 observations since the computation of final choice involved random decisions.

The IBL model described here has two free parameters *d* and σ that were calibrated using a genetic algorithm program. The genetic algorithm repeatedly modified a population of individual parameter tuples in order to find the tuple that minimized the error ratio. In each generation, the genetic algorithm selected individual parameter tuples randomly from a population to become parents and used these parents to select children for the next generation. Over successive generations, the population evolved toward an optimal solution. The population size used here was a set of 20 randomly-selected parameter tuples in a generation (each parameter tuple was a particular value of *d* and σ). The mutation and crossover fractions were set at 0.1 and 0.8, respectively, for an optimization over 150 generations. For each parameter tuple, the IBL model was run 5 times across 2,370 observations. Across the 5 runs, the model's average error ratio was computed by averaging the error ratios from each run. The parameter tuple that minimized the average error ratio across 150 generations were reported as the calibrated parameters for the IBL model.

Results

In the PS model, the best average error ratio across 5 runs was found to be 0.40. This error ratio occurred at *N*=18. Figure 1 shows the average error ratio results obtained from the PS model for different values of *N*. As shown in the figure, for the first few values of *N*, the error ratio reduced rapidly as the size of *N* increased (up to *N* = 18 samples). However, the error-ratio value saturated to a smaller proportion after increasing *N* further. Thus, there was not much variation in the error ratio from *N*=18 to *N*=216.

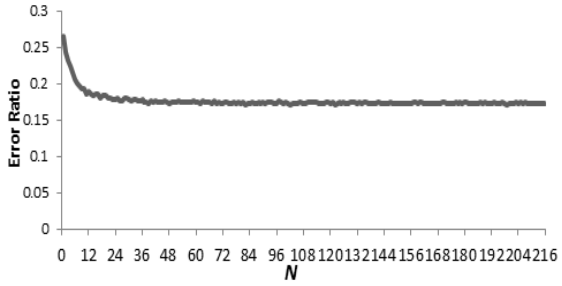


Figure 1. Results from the PS model. The value of parameter *N* was varied between 1 and 216. The best average error ratio = 0.40 for *N*=18.

Table 1 shows the individual-level results from the PS model for the best value of *N*=18. As shown in Table 1, the N_HN_M combinations constituted 24.8% of total combinations. This N_HN_M proportion was second best amongst other combinations (M_HM_M, M_HN_M, N_HM_M, and UN). The M_HM_M combinations had the highest value of 34.3%. The combined average of both the combinations formed about 59.1% of correctly predicted choices by the model. In contrast, the erroneous M_HN_M and N_HM_M ratios were at 21.2% and 19.5% respectively. There were no uncategorized (UN) observations out of 2,370 observations. The PS model's best average error ratio is almost 10% better than the average error ratio of 50% resulting from the CT model. Thus, the PS model explained certain proportion of human dataset fairly accurately.

Table 1. Results from the calibrated PS model. The error ratio = 0.40 for N = 18.

Combinations from Human Data and Model h/m	Number of Observations	Percentage of 2370 Observations
N_HN_M	588.2	24.8
M_HM_M	814	34.3
N_HM_M	463.8	19.5
M_HN_M	504	21.2
UN	0	0

Next, we investigated the performance of the NMH model. Table 2 shows the results from this model. The model's average error ratio equaled 0.44 across 2,370 observations. In the NMH model, the N_HN_M and M_HM_M values accounted for 22.8% and 32.5% of all 2,370 observations, respectively. The best value of error ratio was for the M_HM_M combinations, where about 32.5 % of the 2,370 human observations were accounted by the model. The erroneous M_HN_M and N_HM_M combinations from the model were 23.08% and 21.5% of the 2,370 human observations, respectively. The second best value accounted by the model was that for the M_HN_M combinations. The error ratio obtained from the NMH model is marginally higher than that obtained from the PS model However, the NMH model too, classified all observations. Thus, like the PS model, the NMH model was able to explain a large dataset. The model showed an improvement of 5% over the CT model and performed efficiently at the individual level.

Table 2. The high level results from the NMH model. The average error ratio = 0.44.

Combinations from Human Data and Model H/M	Average Number of Observations	Percentage of 2370 Observations
N_HN_M	542.4	22.8
M_HM_M	771	32.5
N_HM_M	509.6	21.5
M_HN_M	547	23.08
UN	0	0

Next, we evaluated the IBL model's ability to account for individual final choices in the TPT dataset. The results from IBL model are presented in Table 3. The best calibrated values of *d* and σ in the IBL model were found to be 13.6 and 0.22, respectively. The large *d* value exhibited reliance on recency during sampling resulting in maximization. Also, the small σ value exhibited lesser sample-to-sample variability in instance activations. The calibrated IBL model produced 37.6% of N_HN_M combinations and 26.12% of M_HM_M combinations, respectively. Having a total of comparatively higher values for the N_HN_M and M_HM_M combinations increases the accuracy of the IBL model compared to the NMH and PS models. In contrast, the erroneous N_HM_M and M_HN_M combinations were 17.9% and 18.2% respectively from the IBL model and both these percentages were slightly less than those obtained in the NMH and PS models. Thus, the human choices were predicted more correctly by the model for about 11.1% of total observations. This erroneous classification is about 2% higher than the same erroneous classification from the NMH model. Based upon above statistics, the IBL model's performance was better than the PS and NMH models. Also, the number of uncategorized (UN) cases resulting from the IBL model was zero. Furthermore, as the average error ratio from the IBL model was 0.36, the model shows 14% superior performance compared to the CT model.

Table 3. Results from the calibrated IBL model. The calibrated value of parameters d=13.6 and σ=0.22. The average error ratio= 0.36.

Choice Combinations from Human Data and Model	Average Number of Observations across 5 Runs	Percentage of 2370 Observations
N_HN_M	892.6	37.6
M_HM_M	619.2	26.1
N_HM_M	432.8	18.2
M_HN_M	425.4	17.9
UN	0	0

Table 7 shows the results summary from the PS, NMH, and IBL models in the calibration TPT dataset. The PS model gave an error ratio of 0.43. The NMH model gave an error ratio of 0.44, which was close to value of error ratio from the PS model. However, the IBL model gave an error ratio of 0.36, which was less than that of the other two models. The NMH model considers the complete sample size of each observation as opposed to the PS model, where the PS model takes into consideration only last few samples of each observation. In this regard, we can conclude that the PS model is more efficient than the NMH model as it uses a much smaller proportion of samples for about the same error ratio compared to the NMH model. The IBL model's error ratio is lower than the other two models; also, the IBL model does not have any UN observations.

Table 4. Summary of results from the three DFE models on TPT dataset.

Model	Parameters	UN Observations	Error ratio
PS	$N = 18$	0	0.40
NMH	-	0	0.44
IBL	d=13.6,σ=0.22	0	0.36

As the three models have different number of free parameters, we verified the results from these models upon a generalization to a different dataset (Busemeyer & Wang, 2000). For generalization, we used the SC Problems data set. This dataset has 6 problems, out of which 4 are identical to those in the TPT dataset (one option risky and the other safe) and 2 problems are different from the TPT dataset (both options are risky). Table 5 shows generalization results from the PS, NMH, and IBL models (models were run with the parameters derived in the TPT dataset). The IBL model gave the best error ratio of 0.32. Also, the IBL model did not have any UN observations.

Table 5. Summary of results from the three DFE models (SC Problems dataset).

Model	UN Observations	Error ratio
PS	0	0.34
NMH	0	0.37
IBL	0	0.32

Discussions & Conclusion

So far, literature in judgment and decision making had compared models by evaluating their maximizing performance at the aggregate level (Gonzalez & Dutt, 2011; 2012). In such comparisons, the average performance from a model was compared to the average performance from human data (the average was computed across several participants). However, in this paper, we compared a model's maximizing performance at the individual participant level. We used three popular and competing models of aggregate human choice and evaluated their abilities in explaining individual human choices over maximization criterion. Overall, in the TPT dataset, our results reveal that all the three models of aggregate choice performed average at the individual level (error ratio <44%) in both the calibration and generalization datasets. The IBL model's strength is in its ability to account for higher number of maximizing human choices across a large number of observations.

Furthermore, it was found that the two models (PS and NMH) find it easier to explain individual maximizing choices compared to individual non-maximizing choices (there were greater proportion of M_HM_M combinations compared to N_HN_M combinations across the two models). However, the IBL model manages to explain the N_HN_M combinations better than the M_HM_M combinations. Also there were no UN cases reported across the three models.

In addition, the PS and NMH also report a higher proportion of erroneous M_HN_M combinations compared to the erroneous N_HM_M combinations as opposed to the IBL model. Thus, both the models were unable to predict a maximizing final choice for human observations, while the IBL was able to do the same and had a fairly lower error ratio. This finding could also be due to the fact that the one of the options contained two outcomes compared to the single outcome in the other option. Thus, the greater variability experienced in one of the option drove a model to make choices that maximized over a constant value; whereas, the same variability drove humans to maximize upon recency. We generalized the models on another dataset to test this reasoning. In this dataset, again the IBL model performs best thereby proving the stability to the model.

We believe that this paper augurs a beginning of a larger research program that plans to launch an in-depth investigation of the presence of N_HM_M and M_HN_M cases among influential models of experiential choice. As part of future research, we would like to investigate problems where there are multiple options rather than current problems where one options were more of binary kind. Such problems with two options would help us investigate the role of variability in affecting contradictory human and model choices as depicted by the M_HM_M cases. Furthermore, in this paper, we took three competing models of experiential choice; however, as part of future research, we plan to extend this investigation to a larger set of DFE models and other application areas that cover other theoretical ideas.

Acknowledgments

The authors are grateful to Dr. Ido Erev, Technion - Israel Institute of Technology, and Dr. Ralph Hertwig, Max Planck Institute for Human Development, for providing datasets for this paper. Also, the authors would like to thank Tata Consultancy Services' research fellowship to Neha Sharma for supporting this project.

References

Anderson, J. R., & Lebiere, C. (1998). *The atomic components of thought*. Hillsdale, NJ: Lawrence Erlbaum Associates. ISBN 0-8058-2817-6

Busemeyer, J. R., & Wang, Y. (2000).Model comparisons and model selections based on the generalization criterion methodology. *Journal of Mathematical Psychology*, 44, 171–189.

Erev, I., Glozman, I., & Hertwig, R. (2008). What impacts the impact of rare events. *Journal of Risk Uncertainty* 36:153–177.

Erev, I., Ert, E., Roth, A. E., Haruvy, E., Herzog, S. M., & Hau, R.(2010). A choice prediction competition: Choices from experience and from description. *Journal of Behavioral Decision Mak*ing, 23, 15–47.

Gonzalez, C.,& Dutt, V. (2012). Refuting data aggregation arguments and how the instance-based learning model stands criticism: A reply to Hills and Hertwig *Psychological Review*, *Vol* 119(4),893-898.

Gonzalez, C., & Dutt, V. (2011). Instance-Based Learning: Integrating Sampling and Repeated Decisions From Experience. *Psychological Review*,118(4), 523-551.

Hertwig, R., & Erev, I.(2009). The description-experience gap in risky choice. *Trends in Cognitive Sciences*, 13, 517-523.

Hertwig, R. (2012). The psychology and rationality of decisions from experience. *Synthese*, 187, 269-292.

Hertwig, R., & Pleskac, T. J. (2010). Decisions from experience: Why small samples*?*. *Cognition*, 115,225–237.

Lebiere, C. (1999). Blending: An ACT–R mechanism for Aggregate retrievals. *Paper presented at the 6th Annual ACT–R Workshop at George Mason University*. Fairfax County, VA.

Lejarraga, T. & Dutt, V. & Gonzalez, C. (2012). Instance-Based Learning: A general model of repeated binary choice. *Journal of Behavioral Decision Making*,25: 143-153.

Smartphone Users Worldwide Will Total 1.75 Billion in 2014. (2014, Jan 16). Retrieved from http://www.emarketer.com/Article/Smartphone-Users-Worldwide-Will-Total-175-Billion-2014/1010536

Expectations in the Ultimatum Game

Peter Vavra (peter.vavra@donders.ru.nl)
Behavioural Science Institute, Montessorilaan 3
Donders Institue for Brain, Cognition and Behaviour, Kapittelweg 29
Nijmegen, 6525 HR The Netherlands

Luke J. Chang (luke.chang@colorado.edu)
Institute of Cognitive Science, 344 UCB
Boulder, CO 80309 USA

Alan G. Sanfey (alan.sanfey@donders.ru.nl)
Behavioural Science Institute, Montessorilaan 3
Donders Institue for Brain, Cognition and Behaviour, Kapittelweg 29
Nijmegen, 6525 HR The Netherlands

Keywords: social decision-making; Ultimatum Game; social expectations; social norms

Introduction

Being treated fairly by others is an important social need. Experimentally, fairness can be studied using the Ultimatum Game in which the decision to reject a low, but non-zero, offer is seen as a way to punish the other player for an unacceptable division. The canonical explanation of such behavior is inequity aversion: people prefer equal outcomes over personal gains (Fehr, Schmidt, 1999). However, there is abundant evidence that the decision to reject a low offer can be changed by both contextual factors and emotional state, which cannot be satisfactorily explained by the inequity aversion model.

A recent alternative explanation proposes that the main driving force behind the decision to reject is that of deviation from expectations: the larger the difference between the actual offer and the expected offer, the more likely one is to reject the offer (Chang, Sanfey, 2013). We tested and extended this idea by providing participants with explicit information on what kind of offers to expect. Crucially, we independently manipulated both the mean and the variance of expected offers.

Methods

Each participant played as the responder in the Ultimatum Game and made a series of decisions to either accept or reject monetary offers. Participants were provided with information as to what kind of offers to expect in form of histograms, indicating what the current group of partners supposedly offered in a previous experiment. The critical manipulation was of both the mean and the variance of the histograms. First, behavioral data were analyzed using a logistic mixed-model analysis. Second, we fitted and compared different, previously proposed utility models. A second group of participants also underwent scanning using fMRI.

Results

As expected, we found that the decision to accept or reject a certain offer was dependent on the information provided. Importantly, we found that the mean and variance of expected offers differentially affected this decision. Specifically, changing the mean expected offer shifts the threshold for acceptance. In contrast, changing the variance alters how strictly this threshold is adhered to. A model comparison showed that the expectation model outperforms the inequity aversion model.

Conclusions

These results demonstrate the complex nature of social expectations, which might be better conceptualized as distributions instead of simple mean expected values, and how they influence considerations of fairness. Follow-up work is examining the neural bases of these expectations.

References

Fehr, E., & Schmidt, K. M. (1999). A Theory of Fairness, Competition, and Cooperation. *The Quarterly Journal of Economics*, 114(3), 817–868.

Chang, L. J., & Sanfey, A. G. (2013). Great expectations: neural computations underlying the use of social norms in decision-making. *Social Cognitive and Affective Neuroscience*, 8(3), 277–84.

Quantifying Simplicity: How to Measure Sub-Processes and Bottlenecks of Decision Strategies Using a Cognitive Architecture

Hanna B. Fechner (hfechner@mpib-berlin.mpg.de)
Max Planck Institute for Human Development, Adaptive Behavior and Cognition, Lentzeallee 94,
14195 Berlin, Germany

Lael J. Schooler (lschoole@syr.edu)
Syracuse University, Psychology Department,
Syracuse, NY 13244 USA

Thorsten Pachur (pachur@mpib-berlin.mpg.de)
Max Planck Institute for Human Development, Adaptive Rationality, Lentzeallee 94,
14195 Berlin, Germany

Keywords: decision strategies; take-the-best; working memory; cognitive architectures; ACT-R.

Introduction

What makes a decision strategy simple or complex? In this project, we investigated the time costs for cognitive sub-processes and bottlenecks of decision strategies. In order to gauge these time costs, we formally implemented two prominent decision strategies as well as a working memory (WM) load manipulation within the cognitive architecture ACT-R (Anderson, 2007) and compared the performance of the strategies under varying degrees of WM load. We tested the simulation results from this analysis in an empirical study.

The first tested strategy is *tallying* (TALLY), which integrates across several attributes to make a decision (Gigerenzer & Goldstein, 1996). The second strategy is the *take-the-best* heuristic (TTB), which relies on one best attribute (or reason) and considers further attributes only if the decision alternatives do not differ on that reason. As TTB does not integrate across multiple attributes and often searches only part of the information, it is considered a relatively "simple" strategy (Gigerenzer, Todd, & the ABC Research Group, 1999).

Using TALLY and TTB as paradigmatic examples of integrative and one-reason decision making strategies, respectively, we evaluated the hypothesis that integrative strategies induce higher cognitive costs than one-reason strategies—as indicated, for instance, by longer response times and lower execution accuracy when set under WM load (cf. Payne, Bettman, & Johnson, 1993).

Methods

In a dual-task paradigm, one group of participants were instructed to make decisions using TALLY and another group using TTB while being set under WM load by a concurrent tone-counting task.

Decision Task and Strategy Instruction

The decision task was to infer which of two animals has a longer lifespan based on five biological attributes displayed as visual symbols. One group of participants (n = 42) was instructed to use TALLY, which examines all attributes and integrates the attributes in favor of each alternative; the other group (n = 42) was instructed to use TTB—the strategy that justifies decisions with one attribute and only examines more attributes when the decision alternatives do not differ on the current attribute.

To investigate the behavior of TTB in face of varying search requirements, we varied the number of attributes that needed to be searched before a difference between the alternatives could be detected (1-5 attributes).

Working Memory Load

In the concurrent tone-counting task, participants counted the number of times bird voices were played. To induce increasing amounts of WM load, there were either none, one, two or even three different kinds of bird voices. The tones started before the decision information became available and continued until a decision was made in the decision task. Thereafter, it had to be indicated how often the voice of each bird species had been played.

Computational Models in ACT-R

We implemented computational models for the decision strategies and the concurrent counting task in ACT-R 6.0. The models pursue the goals for the decision and tone-counting task, handling multi-tasking demands using Threaded Cognition (Salvucci & Taatgen, 2008). Specifically, the models for the decision strategies (i.e., TALLY and TTB) and for the tone-counting task all rely on ACT-R's problem state and retrieval module, as well as on the procedural module to progress through the course of their tasks (cf. Borst, Taatgen, & van Rijn, 2010).

The models store the information relevant for the two tasks in separate chunks that are either in the problem state (i.e., the buffer of the imaginal module) or need to be retrieved from declarative memory (i.e., when the other task was using the problem state). We modeled effects on reaction times and accuracy in terms of retrieval activity in both tasks using a combination of (a) decay in base-level

activation, (b) spreading activation, and (c) chunk confusion via noise.

From the models we extracted time costs for (a) cognitive sub-processes (i.e., the amount of time that the constituent models rely on a specific cognitive resource corresponding to an ACT-R module), and (b) the bottlenecks they impose on cognitive processing (i.e., the amount of time that the models rely on a specific resource and no other process is executed in parallel).

Results

The modeling results showed that TTB produced faster response times than TALLY when WM load was not present or low and only little information had to be searched to make a decision. When WM load was higher and TTB had to inspect more attributes, however, the decision times of TTB increased strongly and exceeded those of TALLY. Both strategies could be applied with high accuracy that only decreased slightly under WM load. The concurrent task was also executed with high accuracy, although accuracy slightly declined for TTB when more attributes were inspected. Importantly, these patterns of results also emerged in the empirical study.

The computational models made it possible to attribute the performance differences of the strategies to the component cognitive sub-processes in both tasks, revealing bottlenecks to processing: When WM load is high and many attributes have to be inspected, the TTB model spends more time retrieving information from memory, updating WM, and coordinating the mental actions for decisions and the concurrent task compared to the TALLY model.

Another difference between the strategies is that TTB performs an ordered visual search for the next best attribute when the previous attribute did not discriminate between the alternatives; TALLY, by contrast, requires no ordered search as it examines all attributes irrespective of their order. Indeed, visual processing demands imposed time costs and bottlenecks to both models and increased for the TTB model as TTB's search requirements increased. However, these time costs for visual processes did not account for the extended decision times under WM load in any of the models.

Discussion

The results point to differential cognitive costs of paradigmatic examples of one-reason (i.e., TTB) and integrative (i.e., TALLY) strategies. These differences become most evident when more information needs to be searched and the resources for WM are occupied by another task (i.e., tone-counting), revealing conditions under which TTB imposes greater demands on WM than does TALLY.

In this project we used computational models to reveal the cognitive costs of decision strategies under varying internal (i.e., WM load) and external (i.e., search requirements) circumstances. We conceptualized the complexity of decision strategies in terms of time costs for cognitive sub-processes and bottlenecks for cognitive processing. In doing so, we gain novel insights into the cognitive complexity (or simplicity) of strategies. The project therefore extends the concept of building blocks of decision strategies (e.g., Gigerenzer et al., 1999), allowing to quantify the simplicity of "simple heuristics" and to compare it across different decision strategies.

Acknowledgments

This work was supported by the Center for Adaptive Behavior and Cognition (ABC) and Center for Adaptive Rationality (ARC) at the Max Planck Institute for Human Development and the International Max Planck Research Network on Aging (MaxNetAging).

References

Anderson, J. R. (2007). *How can the human mind occur in the physical universe?* New York, NY: Oxford University Press.

Borst, J. P., Taatgen, N. A., & van Rijn, H. (2010). The problem state: A cognitive bottleneck in multitasking. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *36*, 363–82.

Gigerenzer, G., & Goldstein, D. G. (1996). Reasoning the fast and frugal way: Models of bounded rationality. *Psychological Review, 103*, 650–69.

Gigerenzer, G., Todd, P. M., & the ABC Research Group (Eds.) (1999). *Simple heuristics that make us smart*. New York, NY: Oxford University Press.

Payne, J. W., Bettman, J. R., & Johnson, E. J. (1993). *The adaptive decision maker*. New York, NY: Cambridge University Press.

Salvucci, D. D., & Taatgen, N. A. (2008). Threaded cognition: An integrated theory of concurrent multitasking. *Psychological Review*, *115*, 101–30.

Reducing the Attentional Blink by Training: Testing Model Predictions Using EEG.

Trudy Buwalda (t.a.buwalda@rug.nl), Jelmer Borst (j.p.borst@rug.nl), Marieke van Vugt (m.k.van.vugt@rug.nl), Niels Taatgen (n.a.taatgen@rug.nl)
Institute of Artificial Intelligence, University of Groningen, the Netherlands

Keywords: Attentional Blink; EEG; Working Memory.

Introduction

The attentional blink (AB; Raymond, Shapiro, & Arnell, 1992) is a phenomenon that captures people's limited ability to process stimuli presented in quick succession. When a second target item (T2) in a stream of distracting items is presented 200-400ms after the first target (T1), the accuracy of reporting T2 will be decreased as compared to when T2 is presented outside of this time window.

It has long been thought that the AB is a structural capacity limitation, insusceptible for interventions aimed at reducing or removing the AB (e.g. Taatgen, Juvina, Schipper, Borst, & Martens, 2009). However, recently several training studies have shown independently that it is possible to improve recognition of T2 in the impaired time frame (Choi, Chang, Shibata, Sasaki, & Watanabe, 2012; Damsma, 2014). This indicates that the AB is more likely to be caused by the use of a disruptive cognitive strategy than by a structural limitation of the brain.

To explain the reduced AB from Damsma (2014), a cognitive model based on an earlier attentional blink model (Taatgen et al., 2009) was developed to take the training from Damsma into account. The model predicts that consolidation of the first target in memory will be delayed after training, so that the first and second target can be combined into one memory chunk. The goal in the current study is to test this prediction.

We repeated the experiment by Damsma using electro-encephalography (EEG) to focus on an event-related potential (ERP) component that has previously been found to reflect memory consolidation processes, the P300 (e.g., Donchin & Coles, 1988).

If working memory consolidation is delayed after training, as predicted by the model, this should be reflected by a delay in the P300 as well. We therefore expect the onset of the P300 to be later after the training on the letter-mask task then before the training.

Study

Methods

Behavioral Fourteen people (age: 18-27, mean: 22.3; 10 female) performed three parts of the experiment: an AB pretest, an AB posttest, and in between a training using the letter-mask task from Damsma (2014). All three parts of the experiment were performed in one session, with short breaks between the tasks.

In the AB task, participants were presented with zero to two target letters in a stream of 22 numbers. Each item was presented for 100ms. T1 was the fifth item in the stream. In the case of two targets, T2 appeared either 100ms (lag 1), 300ms (lag 3), or 800ms (lag 8) after T1. No feedback was given. Both parts contained 320 trials.

In the letter-mask task a letter was presented on the screen, followed by a mask ('#'). Participants had to recognize and report the letter that was presented as fast as possible after the mask disappeared. The presentation time of the letter was variable and depended on the accuracy of the participant. Presentation times varied between (16 and 91ms). Feedback was given in the form of points for speed and accuracy. The training consisted of 520 trials.

In total, the experiment took approximately 1.5 hours to complete.

EEG EEG activity was recorded from 128 locations according to the ABC electrode system. Data were re-referenced offline to the grand average of all electrodes. Artifact rejection was performed using the FASTER method (Nolan, Whelan, & Reilly, 2010) in combination with visual inspection. The signal was calculated relative to a 200ms pre-stream baseline. ERPs were measured at electrode A19 (Pz).

Results

Behavioral Accuracy results replicate previous experiments; a dip in accuracy is observed on lag 3 compared to lag 1 and 8. The data were analyzed with a t-test on blink size between pre- and posttest. The size of the blink is defined as the difference between the mean accuracy on lag 1 and 8 (no-blink trials) and the accuracy on lag 3 (blink-trials). The size of the blink is smaller after training, compared to before training ($t(13) = -2.03, p = 0.063$).[1]

EEG Only participants showing a training effect are included (11 out of 14).[2] One person was removed due to problems with EEG recording.

The EEG latencies were analyzed using mixed effect models. The latency of the P300 peak was determined by taking the latency of the maximum peak 200-600ms after the onset of the target. We hypothesized that the latency of T1 would be delayed; which could account for the difference in behavioral results.

[1] We use a liberal p-value threshold of 0.1 here, because this small study replicates an effect that has been confirmed in other studies (e.g. Damsma, 2014).

[2] This does not influence the results of this analysis.

Figure 1 shows the grand averages of A19 on lag 3, both targets correct. P300 Latencies on T1 are very similar for other lags. There is no effect of part ($t < 1, p > .1$), nor is there a difference in latency due to lag ($t < 1, p > .1$) or accuracy ($t < 1, p > .1$).

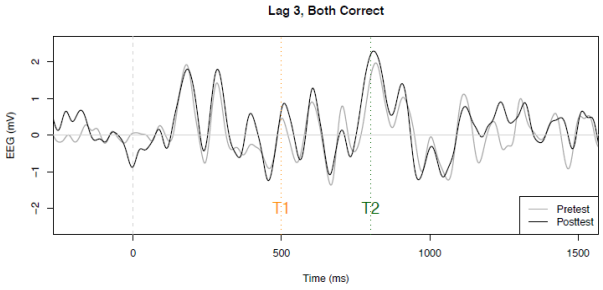


Figure 1: ERP on Lag 3 trials with both targets correctly reported. Band pass filtered from 1 to 30 Hz.

Discussion

The model of Taatgen et al. (2009) predicted that working memory consolidation would be delayed after training. In combination with the relationship between working memory consolidation and the P300 (Donchin & Coles, 1988), we hypothesized that the P300 will occur later after training, as compared to before training.

Although the behavioral results confirm results from previous studies – training on the letter-mask task decreases the attentional blink – we did not find any evidence of a shift in latency of the P300. There are several possible explanations for this. First, the latency of the P300 might not reflect the change in working memory strategy. Although the P300 has been related to working memory consolidation (Donchin & Coles, 1988), the effects of memory on the P300 are most prominent in its amplitude, instead of the latency (Polich, 2007). Other analyses, focused on the amplitude instead of the latency of the P300, are necessary to specify whether there is a relationship between the P300 and the decrease in the attentional blink.

Second, although the model predicts a shift in working memory consolidation after training, this does not necessarily have to explain the improvement after training. Currently, new experiments are exploring alternatives, such as a speed-up in target processing.

From the current analysis, we can conclude that if there is a change in memory strategy following the letter mask training, this change is not reflected in the latency of the P300.

References

Braun, J. (1998). Vision and attention: The role of training. *Nature*, *393*(6684), 424–425.

Choi, H., Chang, L.-H., Shibata, K., Sasaki, Y., & Watanabe, T. (2012). Resetting capacity limitations revealed by long-lasting elimination of attentional blink through training. *Proceedings of the National Academy of Sciences*, *109*(30), 12242–12247.

Damsma, A. (2014). *Overcoming the blink: The effect of training on the attentional blink* (Unpublished master's thesis). University of Groningen.

Donchin, E., & Coles, M. G. (1988). Is the P300 component a manifestation of context updating? *Behavioral and Brain Sciences*, *11*(03), 357–374.

Nolan, H., Whelan, R., & Reilly, R. (2010). FASTER: fully automated statistical thresholding for EEG artifact rejection. *Journal of Neuroscience Methods*, *192*(1), 152–162.

Polich, J. (2007). Updating P300: An integrative theory of P3a and P3b. *Clinical Neurophysiology*, *118*(10), 2128–2148.

Raymond, J. E., Shapiro, K. L., & Arnell, K. M. (1992). Temporary suppression of visual processing in an RSVP task: An attentional blink? *Journal of Experimental Psychology: Human Perception and Performance*, *18*(3), 849.

Taatgen, N. A., Juvina, I., Schipper, M., Borst, J. P., & Martens, S. (2009). Too much control can hurt: A threaded cognition model of the attentional blink. *Cognitive psychology*, *59*(1), 1–29.

Explaining Eye Movements in Program Comprehension using jACT-R

Sebastian Lohmeier (sl@monochromata.de)
TU Berlin, Germany, M.Sc. Informatik degree course

Nele Russwinkel (nele.russwinkel@tu-berlin.de)
TU Berlin, FG Kognitive Modellierung in dynamischen MMS
Sekr. MAR 3-2, Marchstraße 23, 10587 Berlin, Germany

Abstract

We propose that experimentally recorded sequences of eye movements are input into a cognitive model. By removing the need to model decisions on where to look next during a complex task, modelling long-term activation effects in real-world data becomes conceivable. Eye movement records from experiments on program comprehension shall be used because object-oriented source code provides knowledge structures required by a cognitive model of comprehension. We introduce a tool that supports this new approach. The tool is based on an implementation of the ACT-R cognitive architecture written in the Java programming language and could therefore attract Java developers to the cognitive modelling community.

Keywords: tools; ACT-R; Java; eye movements; program comprehension; cohesion; activation

Motivation

Eye movements of programmers reading computer programs can be explained by models written in a cognitive architecture like ACT-R (Anderson et al., 2004). This is comparable to, but may be easier to achieve, than explaining eye movements in natural language comprehension. Modelling program comprehension in a Java-based implementation of ACT-R could, in addition, attract programmers to ACT-R that are capable of contributing to ACT-R as a cognitive architecture.

Psycholinguists like Garrod and Terras (2000) have studied coherence effects (i.e. effects of semantic relations expressed in a text) in reading using eye tracking and provided explanations for their findings that are suitable for cognitive modelling. While it would be possible to model the results of these experiments in ACT-R, there are a number of open problems for such models. E.g. representations of lexical and conceptual knowledge required for these tasks are not readily available (Lohmeier & Russwinkel, 2013).

ACT-R has already been implemented in Java (see jACT-R[1] and Java ACT-R[2]) and can therefore be integrated directly into the development environments used by Java programmers, making ACT-R development possible in a well-known programming language. This would also allow ACT-R tools, e.g. for visualisation, to be developed using a larger set of existing application programming interfaces and frameworks than is available for Lisp.

Eye Movements in Program Comprehension

Mandel (1984) compared measures of eye movements during reading to predictions generated by a computational implementation of a theory from cognitive linguistics. Burkhardt, Détienne, and Wiedenbeck (1997) used a related theory to explain findings on the comprehension of the source of object-oriented computer programs. It has also been suggested that computational cognitive models be used to model program comprehension (Hansen, Lumsdaine, & Goldstone, 2012). For object-oriented programming languages provide knowledge structures required for models from cognitive linguistics (Lohmeier, 2014), eye movements of programmers are furthermore suitable for cognitive modelling based on theories from cognitive linguistics: Recorded eye movements could be used to model activation of conceptual knowledge represented in the source code. Because the source code can be parsed automatically, the conceptual knowledge expressed in the source code can be made available to the cognitive model and can be used to compute activation values to explain coherence effects in source code that are comparable to those found in studies like Garrod and Terras (2000).

Eye Tracking and jACT-R in Eclipse

We are developing a plug-in that controls an eye tracking device and sends eye movement data to a cognitive model.

Data capture and analysis

Similar to iTrace (Walters, Falcone, Shibble, & Sharif, 2013), our plug-in enables the Eclipse IDE[3] to calibrate an eye tracker, to receive data from it, to save the data to disk and to assign the data to user interface elements and words displayed at locations fixated by the user. Both iTrace and our plug-in support different eye trackers and user interface elements and can be extended to support additional ones. Our plug-in has been used to connect to eye tracking devices of SMI and The Eye Tribe and is able to map fixations to Text, StyledText, Label and Button user interface elements. Our plug-in implements complex event processing through an extensible chain of filters through which eye tracking events are passed. There are different modes, e.g. for tracking, replay and batch replay so that visualising filters can render fixations on screen during replay, but not in tracking or batch replay mode.

Interfacing eye movements and model

Preparation of data for the jACT-R model is implemented as a filter that saves saccades, fixations, and words assigned to fixations, to a log file. While tools like those of Salvucci (2000) and Heinath, Dzaack, Wiesner, and Urbas (2007) compare records of eye movements obtained experimentally to

[1] http://jact-r.org/
[2] http://cog.cs.drexel.edu/act-r/
[3] https://www.eclipse.org/

eye movements generated by a model, the model we are developing does not compute durations and targets of saccades but receives this information from the log file recorded during an experiment. The model executes cognitive processes that yield activation of knowledge representations and fixation durations. The jACT-R model is launched as a separate process within the Eclipse IDE. While data capture, analyis and submission to the model could all happen on-line, analysis and input into the model are currently separated to be able to verify the data before executing the model.

Our model does not generate saccades, but follows Salvucci (2001) in how it distinguishes but couples visual attention and the onset time of saccades, whose durations are read from the log file. For each saccade a duration is provided. Fixations come with a duration and (multiple) words within foveal and parafoveal vision. The model generates fixation durations. In addition, activation values are provided by the model. If a passage of source code has low cohesion that leads to regressive saccades in the experimental data, the model can explain this behavioural effect by means of low activation of chunks in memory that makes memory requests for a chunk referred to by a recently fixated word fail.

Discussion

The reconstructive model we aim at differs significantly from typical ACT-R models that *generate* behaviour in a *goal-directed* way. The reconstructive model attempts to execute low-level cognitive processes that fit the given fixations. There will be situations in which a sequence of fixations will end in a way that makes clear to a human analyst of the model that the model failed to execute cognitive processes that explain the behavioural data in a correct or plausible way. The better a model fits the data, the fewer of such occasions will occur. Depending on the context-dependence of the cognitive processes implemented, it may also be required to test different mutually exclusive sequences of cognitive processes and select the one that fits the behavioural data best. That would require the state of the cognitive architecture be replicated for each of such sequence and is beyond our current implementation which aims at an initial model of longer-lasting activation scenarios e.g. based on 40 minutes of eye movements.

Conclusion

While not the only possible application of ACT-R for explaining records of eye movements, eye tracking studies in program comprehension are well suited for this kind of modelling because the source code provides knowledge structures that can be used in the cognitive model. Restricting the model to low-level cognitive processes as used to establish coherence during reading should permit the implementation of a model that is able to explain rather long-term phenomena compared to models that generate goal-oriented behaviour themselves. Implementing the model in jACT-R could attact further developers to cognitive architectures.

Acknowledgments

We thank Anthony Harrison for implementing jACT-R and for helping us to get started with it.

References

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of mind. *Psychological Review*, *111*(4), 1036–1060.

Burkhardt, J.-M., Détienne, F., & Wiedenbeck, S. (1997). Mental representations constructed by experts and novices in object-oriented program comprehension. In S. Howard, J. Hammond, & G. Lingaard (Eds.), *Human-computer interaction: INTERACT '97*. London: Chapman & Hall.

Garrod, S., & Terras, M. (2000). The contribution of lexical and situational knowledge to resolving discourse roles: Bonding and resolution. *Journal of Memory and Language*, *42*, 526–544.

Hansen, M. E., Lumsdaine, A., & Goldstone, R. L. (2012). Cognitive architectures: A way forward for the psychology of programming. In *Onward! 2012: Proceedings of the ACM international symposium on New ideas, new paradigms, and reflections on programming and software* (pp. 27–37).

Heinath, M., Dzaack, J., Wiesner, A., & Urbas, L. (2007). Simplifying the development and the analysis of cognitive models. In S. Vosniadou, D. Kayser, & A. Protopapas (Eds.), *Proceedings of EuroCogSci07*. Hove: Erlbaum.

Lohmeier, S. (2014). Computational linguistics vice versa. In B. du Boulay & J. Good (Eds.), *Proceedings of the psychology of programming interest group annual conference 2014* (pp. 191–196).

Lohmeier, S., & Russwinkel, N. (2013). Issues in implementing three-level semantics with ACT-R. In *Proceedings of the 12th international conference on cognitive modeling (ICCM)*.

Mandel, T. S. (1984). *An eye movement investigation of a process model of comprehension* (Tech. Rep. No. 84-132). University of Colorado, Institute of Cognitive Science.

Salvucci, D. D. (2000). An interactive model-based environment for eye-movement protocol analysis and visualization. In *Proceedings of the 2000 symposium on eye tracking research & applications* (pp. 57–63). New York: ACM.

Salvucci, D. D. (2001). An integrated model of eye movements and visual encoding. *Cognitive Systems Research*, *1*(4), 201–220.

Walters, B., Falcone, M., Shibble, A., & Sharif, B. (2013). Towards an eye-tracking enabled IDE for software traceability tasks. In *International workshop on traceability in emerging forms of software engineering (TEFSE), 2013* (pp. 51–54).

Affordances based k-TR Common Coding Pathways for Mirror and Anti-Mirror Neuron System Models

Karthik Mahesh Varadarajan (kv@acin.tuwien.ac.at)
Technical University of Vienna
Austria

Abstract

The *k*-TR theory to visual perception is a recent psychophysical theory that explains the processing of visual perception in humans towards the goal of object detection, recognition, grasping and manipulation through the notion of functional and grasp affordances. In this work, we postulate a new Brain Operating Model based on the k-TR theory and supported by various neuro-biological studies to accompany the observed psychophysics. The key components of the model include the Mirror Neuron System (MNS) and the Anti-Mirror Neuron System (AMNS).

Keywords: Affordances; brain operating model, schema theory

Introduction

In this paper, we build a Brain Operating Model to explain affordance-related perceptual processing in the brain. We also design a schema theory based on PHG anti-mirror neurons and the known affordance coding linkages (as noted above) of the PHG substrate, calling it the Anti-Mirror Neuron System (AMNS) and use it to explain various aspects of the k-TR Common Coding Theory. The various sub-schemas of AMNS such as the Object/Hand Perception Schema, Reach/Grasp Schema, Core Anti-Mirror circuit along with the various biological units catering to the schemas in terms of sub-tasks such as object affordance extraction, motor execution, hand motion detection etc. and their contributions are analyzed through simulations and validated through psychophysical tests that involve subject recall of concrete nouns based on observation of affordance executions with/without the target object, self-actuation without visual perception and solely based on touch and move-assist by an external agent, along with control tests and negative affordance coding linkages. These psychophysical tests demonstrate the contribution of various affordance features (observed/imagined, as well as both visual coding and touch/motor coding) with respect to object recognition or object identity label (concrete noun) association.

Hypotheses Proposals

Hypothesis 1: There exist mirror neuronal reverse linkages from dorsal to ventral pathway for decoding sensorimotor affordance cues

Hypothesis 2: The sensorimotor encodings (action-perception common code) carry affordance features as described by the *k*-TR theory (Varadarajan 2011)

Hypothesis 3: Suggestive recruitment and modulation of dorsal and ventral feature faculties occurs in an iterative refinement process

Hypothesis 4: The linkage between perception affordance features and motor affordance (observed/ego-actuated) features is an independent process (Molyneux's problem has a negative response) and key to visual object perception

Corollary 1: Linguistic association in Object Recognition depends on affordance features from visual perception

Hypothesis 5: Anti-mirror neurons act as interfaces or junctions through which physical/manipulation affordance features extracted from perception (observed/simulated) in the dorsal pathway is filtered to remove motor features and integrated with material affordances, affordance semantics and other local instance features leading to retrieval of both object category and object instance from memory.

Corollary 2: Anti-mirror neurons are responsible for generation of perceptual affordance encodings of objects.

Dorsal Pathway and Ventral Pathway

The dorsal stream essentially represents the 'Affordance' stream or the '*k*' layer stream, while the ventral stream largely represents 'TR' feature extraction. These two pathways are physiologically described as the ventral and dorsal streams in the brain. The ventral stream can be expected to processes fine-grained texture information (TR). Early models of this Two-Stream Hypothesis portrayed the dorsal and ventral pathways as being independent of each other. Newer models show multiple interconnections between the two. Yet these have largely focused on the dorsal pathway receiving input from the ventral stream and not other way around (esp v3).

Mirror Neurons

Mirror neurons have been proposed as fundamental mechanism for learning new actions through imitation of observed action-affordances. Since mirror neurons respond to both ego-action/affordance execution as well as perception (visual or other sensory) of execution of action/affordances by an external agent, they provide the physiological mechanism for perception/action-affordance coupling. Mirror neurons are found along both the dorsal as well as the ventral pathways.

Affordance based Brain Operating Model and Object Recognition/ Manipulation Schema

We hypothesize in this work that there exist unique reverse linkages from the dorsal stream to the ventral stream and an iterative exchange of information along the forward and

reverse linkages is responsible for increase in granularity and fidelity of output from the two streams. In other words, the theory on "how" an object is/is to be afforded (through grasping and task driven or functional manipulation and typically based on geometric shape and spatial analysis - key processing done in the dorsal pathway) provides cues to recruit layers that enhance the extraction of object relevant features and parameters that determine "what" the object is. We further hypothesize that these cues represent sensorimotor encodings that contain affordance features (from *k*-TR) theory. For example, the first iteration from the dorsal to the ventral pathway could encode for attentional modulation based on motion, bottom-up affordance aberration saliencies and task driven top down affordance modulation. On the other hand the first iteration from the ventral to the dorsal pathway could encode material (frequency content analysis) affordances. We hypothesize here that the encodings are sensorimotor in nature and depend on observed/actuated motor, as well as affordance features from perception (primarily visual) along with the transformation from perception features to motor features playing a key role. A corollary from the above hypothesis is that linguistic association or object category labeling, which is an essential sub-process in object recognition should also employ affordance features from visual perception, since these are most dominant discriminative features for recognition. Direct support for the corollary comes from the recent breakthrough work from Just et al. (2010). The developed Brain Operating Model along with the sub-regions of the brain that compose the affordance extraction schema as well as the forward-reverse connections between the dorsal and ventral pathways are demonstrated in figure 1a. The various schemas associated with the brain regions interacting with the model are shown in the figure 1b.

Anti-Mirror Neurons and Anti-Mirror Neuron System (AMNS)

Besides the canonical F5 neurons that respond to ego-affordance execution and mirror neurons that respond to both ego and exo-affordance execution, there are also anti-mirror neurons that are in a excitatory state only in response to exo-affordance execution and accompanying linguistic association in the form of entities involved in the affordance execution. Anti-mirror neurons are found in the Parahippocampal Gyrus and additional neurons with similar function in the Entorhinal cortex. It can be hypothesized here that anti-mirror neurons might be interface or junction through which physical/manipulation affordance features extracted from perception (observed/simulated) in the dorsal pathway is filtered to remove motor features and integrated with material affordances, affordance semantics and other instance features leading to retrieval of both object category and object instance from memory.

The validity of the proposed Brain Object Perception Model is also supported by Anti-Mirror Neuron System (AMNS) Psychophysical tests (Varadarajan 2013). The contributions of the various modules in AMNS were analyzed through simulations and validated through psychophysical tests that involve subject recall of "concrete nouns" based on observation of affordance executions with/without the target object, self-actuation without visual perception and solely based on touch and move-assist by an external agent, along with control tests and negative affordance coding linkages. These psychophysical tests demonstrated the contribution of various affordance features (observed/imagined, as well as both visual coding and touch/motor coding) with respect to object recognition or object identity label (concrete noun) association. On tests for recall of 20 samples from 50 categories across 17 test subjects, direct recall yielded an accuracy of 34%, while affordance aided recall yielded and negative affordance primed recall yielded rates of 62% and 21% with object and 60% and 2% without object. Hence, affordances were found to be key to recognition. The difference between observed (exo-) and self (ego-) affordance execution were minimal from the standpoint of recognition of objects in simulated affordance sequences.

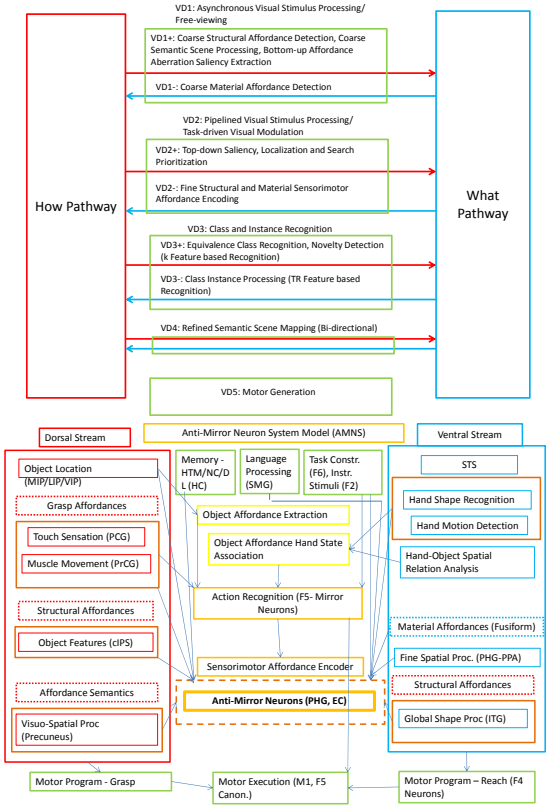


Figure 1a (top). Brain Operating Model and 1b (bottom). Schemas

References

Just, M. A., et al. (2010). *A neurosemantic theory of concrete noun representation based on the underlying brain codes*. PloS one, 5(1), e8622.

Varadarajan, K. M. (2011). *k-TR: Karmic Tabula Rasa–A Theory of Visual Perception*. Fechner Day, 2011.

Varadarajan, K. M. (2013). *Anti-mirror neuron system model for affordance based k-TR Common Coding Theory*. Fechner Day 2013, 93. IEEE.

Functional Cognitive Models of Malware Identification

Christian Lebiere, Stefano Bennati, Robert Thomson ({cl, sbennati, thomsonr}@andrew.cmu.edu)
Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA 15213 USA

Paulo Shakarian, Eric Nunes ({shak, eric.nunes}@asu.edu)
Arizona State University
699 S. Mill Avenue, Tempe, AZ 85281 USA

Abstract

An important source of constraints on unified theories of cognition is their ability to perform complex tasks that are challenging for humans. Malware reverse-engineering is an important type of analysis in the domain of cyber-security. Rapidly identifying the tasks that a piece of malware is designed to perform is an important part of reverse engineering that is manually performed in practice as it relies heavily on human intuition. We present an automated approach to malware task identification using two different approaches using ACT-R cognitive models. Against a real-world malware dataset, these cognitive models significantly out-perform baseline approaches while demonstrating key cognitive characteristics such as the ability to generalize to new categories and to quickly adapt to a change of environment. Finally, we discuss the implications of our approach for applying cognitive models to complex tasks.

Keywords: functional cognitive models, ACT-R, Bayesian models, decision trees, malware detection.

Introduction

Malware reverse-engineering is an important type of analysis in the domain of cyber-security. Rapidly identifying the tasks that a piece of malware is designed to perform is an important part of reverse engineering and is manually performed in practice as it relies heavily on human intuition (Sikorski & Honig, 2012). The difficulty of this task increases substantially when historically studied malware samples are significantly different (i.e. members of a different malware family). Cognitive architectures such as ACT-R (Anderson, Bothell, Byrne, et al., 2004) have previously been shown to effectively model human cognition on a variety of decision-making (Lebiere, Gonzalez, & Martin, 2007) and general intelligences tasks (Lebiere, Gonzalez, & Warwick 2009), including complex domains such as intelligence analysis (Lebiere, Pirolli, Thomson, et al., 2013). Further, they have been shown to perform well on reasoning tasks where historical knowledge is sparse, limited, or dissimilar to the current context (Taatgen, Lebiere, & Anderson, 2006). However, models have occasionally had to abstract from some of the details of the high-fidelity framework that cannot be constrained by data in order to scale to complex tasks involving substantial human expertise (e.g., Sanner et al, 2000). Our work fits into that approach by selectively using some features of the cognitive architecture while temporarily ignoring others.

Malware Identification

In this paper, we leverage such models to identify the tasks associated with a piece of malware. Using a real-world malware dataset (Mandiant Corp, 2013), these cognitive models identify sets of tasks with an unbiased F1 measure of 0.94 – significantly out-performing baseline approaches. Even when trained on historical datasets of malware samples from different families, our ACT-R cognitive models still maintain the precision of baseline methods while providing a significant improvement to recall by identifying over 60% of malware tasks.

Existing work on malware classification falls into two general categories: (1) determining if a given binary is malicious (Tamersoy, Roundy & Horng 2014; Firdausi, Lim Erwin, & Nugroho, 2010) and (2) classifying malware by family (Bayer, Comparette, Hlauschek, et al., 2011; Kinable & Kostakis, 2011; Kong & Yan, 2013). The problem of identifying whether a binary is malware is complementary to this effort (a "first step") – as an analyst must first identify malware before then determining what it does. Our work substantially differs from malware family classification as we look to directly infer the tasks that a malware was created to perform whereas malware family classification is mainly used to help guide an analyst into identifying tasks by first identifying a family. It is noteworthy that we were able to train our classifiers on data of malware of *different* families than the malware we are attempting to classify and were still able to obtain a set of tasks with over 60% recall on the best-performing cognitive models. Further, malware family classification has suffered from two primary draw-backs: (1) disagreement about malware family "ground truth" as different analysts (i.e. Symantec and MacAfee) cluster malware into families differently; and (2) previous work has shown that some of these approaches mainly succeed in "easy to classify" samples (Perdisci, 2012; Li, Liu, Gai & Reiter, 2010) – where an "easy to classify" family is a family that is typically agreed upon by multiple malware analysis firms. By inferring malware tasks directly, we avoid both of these pitfalls. Further, as a side-effect, we create a probability distribution over malware families as part of an intermediate step – though the ultimate inference of malware tasks is independent of *how* the historical malware families are classified by family.

ACT-R Models

The models are built using the mechanisms of the ACT-R cognitive architecture and learn to recognize malware samples based upon a limited training schedule similar to the actual experience of a human analyst. Given a malware sample, the model generates a probability distribution over a set of malware families then infers a set of likely malware intents based upon that distribution. The models primarily leverage the subsymbolic (statistical) mechanisms of the ACT-R architecture, especially the activation calculus underlying retrieval from long-term declarative memory. Each sample is represented by its set of static and dynamic attributes. The model operates in two stages: first by family, then by intent. To assign family, the model generates a probability distribution over the set of possible malware families from the activation in declarative memory of the chunks representing those families. To assign intents in a second pass, the model combines the probability distribution over families with a representation linking each malware family to known intents. Two distinct models were created that leveraged separate parts of the activation calculus.

ACT-R Rule-Based Model

This ACT-R model is based on the Bayesian components of the activation calculus, specifically the base-level and spreading activation components. Given a malware training sample with its set of attributes, along with the ground truth family, we derive a pair of conditional probabilities $p(a/f)$ and $p(a/\neg f)$ for attribute a belonging (or not) to family f. Those probabilities are used to set the strengths of association from each attribute a to each family chunk f. Similarly, Bayesian priors $p(f)$ are used to set the base-level of each family. Given the attributes of the current malware held in the goal buffer context, a retrieval for family chunks (the "rules", not to be confused with production rules) computes their activation and sets the probability of each family according to the Boltzmann (softmax) equation. Intents are then determined by summing up the probability of the families associated with a given intent, with an appropriately set threshold (50%).

ACT-R instance-Based Model

This model follows the instance-based learning theory (IBL; Gonzalez, Lerch, and Lebiere, 2003) that is particularly relevant to modeling naturalistic decision making in complex dynamic situations. The instance-based approach is an iterative learning method that reflects the cognitive process of accumulating experiences and using them to make decisions. In this case a chunk is created for each malware instance associating the set of attributes of that malware with its family. When a new malware is encountered, a retrieval for past chunk instances is triggered with the purpose of inferring their family. The retrieval primarily uses the base-level and partial matching components of the activation equation. The base-level reflects the recency and frequency of each instance according to the power law of learning and decay, while the similarity measure used in partial matching is computed as the overlap (dot product) between the attribute vector of the current malware and each sample in memory. A probability distribution over families is generated by the blending mechanism that sums up the evidence supporting each family from the individual instance chunks (Lebiere, 1999; Wallach & Lebiere, 2003). The same process is used for generating intent judgments, this time partial matching the family probability distribution of this malware instance against those of past instances. The intent chunks that reach the activation threshold are given as answers.

Experiment

We created a dataset from 132 malware samples used by the APT1 cyber espionage group as identified by the popular report by Mandiant Inc (Mandiant, 2013). Dynamic malware analysis was performed using the ANUBIS sandbox which generates an XML-formatted report for each malware. From the XML data, a total of 1740 malware attributes were identified (see Table 1).

Table 1: Sample attributes from Anubis malware sandbox

ATTRIBUTES	INTUITION
hasDynAttrib	Malware has a generic attribute determined in the analysis
usesDll(***X***)	Malware uses a library ***X***
regAct	Malware conducts an activity in the registry
fileAct	Malware conducts an activity on a certain file
proAct	Malware initiates or terminates a process

Each malware sample belonged to one of 15 families (e.g., BISCUIT). Based on malware family description, we associated a set of tasks with each family that each malware in that family was designed to perform. In total, 30 malware tasks were identified for the given malwares (see Table 2). On average, each family performed 9 tasks.

Table 2. Sample of malware tasks.

TASK	INTUITION
beacon	Beacons back to the adversary's system
enumFiles	Designed to enumerate files on the target
ServieManip	Manipulates services running on the target
takeScreenShots	Takes screen shots
upload	Designed to upload files from the target

Decision Tree

We implemented a decision tree as a baseline approach. This hierarchical algorithm is widely used for classification problems (Alpaydin, 2007). We used information gain to find the best split at a node. The gain was calculated using malware attributes. In order to avoid over-fitting, the

terminating criteria was set to less than 5% of total samples. Note that labels are not used for terminating the tree, hence the leaf nodes may or may not be pure, generating a probability distribution over the malware families.

Results

We compared the decision tree (DT) approach to implementations of the rule-based and instance-based ACT-R models (ACTR-R and ACTR-IB respectively). Precision, recall and F1 values were computed for the inferred adversarial tasks. On average, each sample was associated with 9 tasks out of 30 different tasks in total. DT predicted 9 tasks per sample, ACTR-R 9 tasks, and ACTR-IB 10 tasks.

Leave One Out Cross-Validation (LOOCV)

In leave one out cross validation, for N malware samples, we train on N-1 samples and test on the remaining one. This procedure was repeated procedure for all samples and the results were averaged (see Figure 1).

ACTR-IB outperformed both the DT and ACTR-R models; average F1 = 0.94 vs .81 ($t(132) = 5.77$, $p = 5e\text{-}8$), and .82 ($t(132) = 5.35$, $p=3.83\text{-}7$) respectively. The Bayesian nature of the ACT-R model dominates because it is trained on a stable, almost complete set of statistics. The IBL model is superior because it uses the full pattern of the probability distribution over families rather than just a sum.

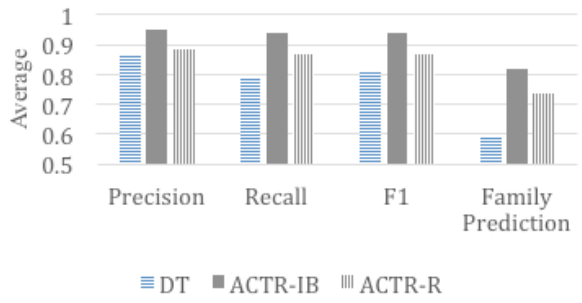


Figure 1. Average Precision, Recall, F1 and Family prediction comparisons for DT, ACTR-IB and ACTR-R.

These three approaches were also evaluated with respect to predicting the correct family (before the tasks were determined). Both the ACTR-IB and ACTR-R cognitive models outperform DT to predict the correct malware family. ACTR-IB has an average family prediction accuracy of 0.82, outperforming the DT model's accuracy of 0.6, $t(132) = 5.35$, $p = 3.8e\text{-}7$. ACTR-R also outperformed with prediction accuracy of 0.72 vs 0.6, $t(132) = 3.23$, $p = 1e\text{-}3$. Figure 2 (below) shows family-wise performance for LOOCV. This gives an unbiased estimation regarding predictions for different malware families, giving insight as to which families are difficult to predict.

ACTR-IB outperforms DT in 9 out of 15 malware families with an average F1 difference of 0.3 with at least 99% confidence, $t(132) = 4$, $p = 0.01$. DT performs qualitatively better than ACTR-IB in 4 out of 15 malware families with an average F1 difference of 0.05, but this difference is not statistically significant, $t(132) = 0.76$, $p = 0.49$. Similarly, ACTR-R outperforms DT in 7 out of 15 malware families with an average F1 difference of 0.27, while DT does not perform significantly better than ACTR-R, $t(132) = 0.28$, $p = 0.786$.

Among the cognitive models ACTR-IB performs qualitatively better than ACTR-R in 12 out of 15 families with average F1 difference of 0.08, but this difference is not statically significant, $t(132)= 1.58$, $p = 0.19$.

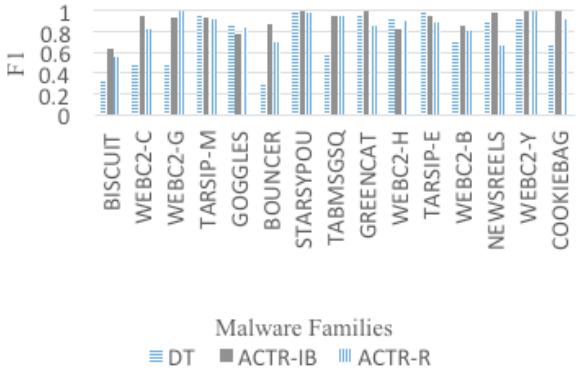
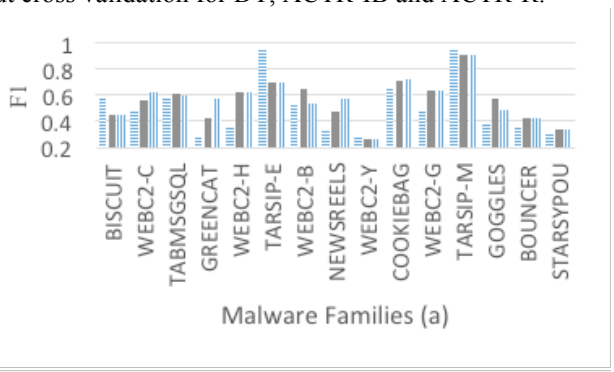


Figure 2. F1 measure by malware families for leave one out cross validation for DT, ACTR-IB and ACTR-R.



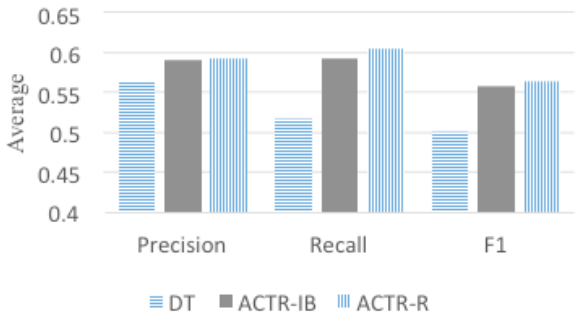


Figure 3. Average F1 values for 15 malware families (a) and the average precision, recall and F1 across all families (above) for DT, ACTR-IB and ACTR-R.

Leave One Family Out Cross-Validation

To see how the models generalize to unseen malware families, we performed a leave-one-family-out comparison,

where we test the models against one previously unseen malware family. Both the ACTR models significantly outperform the decision tree in terms of precision, recall and F1 (see Figure 3). This is primarily due to the statistical nature of the classification performed by the cognitive models over the logical classification of the decision tree.

Despite the overall higher performance for the cognitive models over the decision-tree, there were certain families where the decision tree performed particularly well when compared to the cognitive model. In the F1 comparison the decision tree peaks for TARSIP-Eclipse and TARSIP-Moon. Both these families are variants of the same malware profile. TARSIP-Eclipse performs 12 tasks, while TARSIP-Moon performs 13. They have 12 tasks in common hence during testing one family gets incorrectly predicted as the other while still getting almost all their tasks correct.

90/10 Training/Testing

Finally, we randomly divided the data into 90% training and 10% testing. This measure was then divided into 10 phases, where in the first phase the models were trained with 10% of the total training data (which was 90% of the dataset) and then an additional 10% of the training data was added for each subsequent phase. Note that each phase gets tested on the same test data. This allows us to observe the performance of the decision tree and cognitive models for incremental learning across the training data. In the real world, humans need to be able to learn from small partial samples and adapt quickly to changes in the underlying distribution. As shown in Figure 4, it is clear that the ACT-R models outperform the decision tree in precision, recall and F1 measures. This is particularly true of the instance-based model, which uses the dynamic nature of the blending mechanism to generalize over the entire space from just a few instances.

An important point to note is that the cognitive models achieve the best performance against the decision tree with only 40% of the training data. T-tests were computed for each fraction of training data comparing each of the ACT-IB, ACTR-R, and DT models against each other. The results of both ACTR models statistically outperformed decision tree (all $p < .001$) except for when 30% of the training data was used ($p = 0.46$). We hypothesized that our random sample for the 30% training data phase may have underrepresented the population of malware samples where the decision tree performs poorly. By examining the family-wise performance for leave one out cross validation (see Figure 2) we determined that decision tree has difficulty predicting malware tasks from families BISCUIT, WEBC2-CSON, WEBC2-GREENCAT, TABMSGSQL, COOKIEBAG and NEWSREELS (difference in F1 measure is greater than 0.3 as when compared to ACTR-IB or ACTR-R). The overall fraction of malware samples belonging to these families is 0.36 and in all phases, except it is 0.31 in the 30% phase, thus relatively increasing the performance of decision tree at that point.

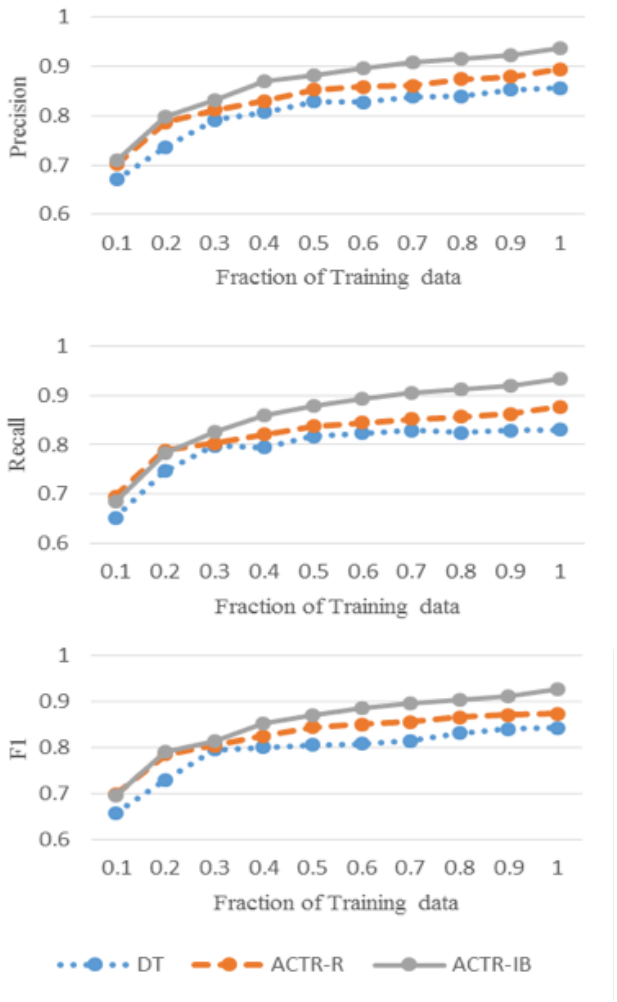


Figure 4. Average Precision, Recall and F1 for fraction of training data of 200 trials for DT, ACTR-IB and ACTR-R.

Discussion

These ACT-R models are not full-fledged high-fidelity models in that, while they make sole use of cognitive mechanisms, they do not use all aspects of the architecture, nor are they directly compared to human data. The primary reasons for this approach are three-fold: (1) because of the challenging nature of the task, we decided to focus on the functional aspects of the model; (2) we did not believe that the unmodeled aspects of the task would significantly impact the performance of the model; (3) we did not have human performance data with which to assess the model.

Regarding (1), we believe that there is a valid use of cognitive architectures for artificial intelligence that makes use of basic cognitive mechanisms while not necessarily making use of all constraints of the architecture. In that case, the model has to be evaluated on functional grounds, which is the approach that we took. However, we also discuss in the concluding section which aspects of the model were currently not cognitively plausible, such as the lack of working memory constraints, and how they could be remedied, perhaps by improving current deficiencies of the

architecture. In general, artificial intelligence constraints, such as high performance on complex tasks, can serve a valuable purpose in driving the development of cognitive architectures. Conversely, constraints on unified theories of cognition can be used to design more useful benchmark tests of artificial intelligence (Lebiere et al., 2015).

Regarding (2), Reitter & Lebiere (2010) introduced a modeling methodology called accountable modeling that recognizes that not every aspect of a cognitive model is reflected in measurable performance, and thus that human performance data cannot constrain all aspects of a model. In that case, it is arguably better to specifically state which aspects of the model are not constrained by data, and rather than mock up those aspects in plausible but impossible to validate manner, simply treat them as unmodeled processes. This approach results in simpler models with a clear link between mechanisms used and results accounted for, rather than being obscured by complex but irrelevant machinery.

Regarding (3), we are exploring obtaining human data through empirical studies using expert malware analysts to provide the kind of data that can be directly compared against performance provided by a full ACT-R model.

Conclusion

We present two cognitive models of malware intent classification. Those models are both based on the ACT-R cognitive architecture but leverage separate mechanisms and have distinct advantages. The rule-based model leverages the Bayesian memory activation mechanisms. The representation is more compact, with a single memory chunk for each family whose associations abstract the various instances belonging to that category, but those associations need to be computed and do not involve time discounting and other adaptive features (Thomson & Lebiere, 2013). The instance-based model is based on a more direct, incremental learning that accumulates malware instances in long-term memory and leverages neurally plausible pattern matching processes such as partial matching and blending (Lebiere et al., 2013) but is less parsimonious with storage and thus has potential scalability issues for large data sets.

A number of further model developments can address those and other issues. The first computational efficiency issue is the size of the feature set, which can easily number in the hundreds for a given malware. It is also an issue of cognitive plausibility since feature set size is associated to working memory, usually assumed in humans to be about seven or so (Miller, 1956). Reducing feature set size could also potentially improve generalization by removing features that are only misleadingly associated with specific intents and focusing on those that are causally related to malware function. One potential approach is to choose among features those that most contribute to correct performance. This can be implemented in ACT-R by relying on the production utility reinforcement learning mechanism to sequentially select specific features (Rutledge-Taylor et al., 2011). Given the limited working memory constraint in the form of a fixed spreading activation parameter, this can benefit performance both in eliminating spurious features and focusing limited attentional resources on the most diagnostic features.

Another approach to reducing feature set size is to build higher order features with which to represent malware instances. This process is similar to the concept of chunking in expertise-driven domains such as chess playing (Chase & Simon, 1973). When those higher-order features are known, they can be directly incorporated in the model and have been shown to improve learning performance by orders of magnitude (Sanner et al., 2000). Alternatively, a deep learning algorithm could be used to infer those features in a manner similar to past efforts combining ACT-R with neural learning mechanisms (e.g., Jilk et al., 2008, Vinokurov et al, 2011), illustrating the benefits of combining symbolic and neural architectures.

A second model development to improve computational efficiency for the instance-based learning model would be to reduce the size of the instance set in long-term memory. One possibility is to reinforce the most similar malware chunk(s) already in memory instead of creating a new one, which has already been shown to preserve generalization while sharply reducing memory requirements and retrieval process demands (Sanner et al, 2000). This approach results in the emergence of prototypes that can be seen as a middle ground between the single-chunk representation of categories in the rule-based model and the pure instance-based approach of the IBL mode.

Another approach to reducing the size of the feature set would be to include an ontology of malware functionality (e.g., Mateos et al., 2012) that would allow the model to reason over the association of features and intents and prune the representation (e.g., Oltramari et al., 2014). This process of combining symbolic reasoning to guide statistical learning is one of the main advantages of integrated cognitive architectures.

Acknowledgements

This work is supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of the Interior (DOI) contract number D10PC20021. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained hereon are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DOI, or the U.S. Government.

References

Alpaydin, E. 2007. Introduction to Machine Learning. *Massachusetts Institute of Technology*.

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. 2004. An integrated theory of mind. Psychological Review, 11(4), 1036-1060.

Bayer, U., Comparetti, P.M., Hlauschek, C., Krügel, C., Kirda, E. 2009. Scalable, behavior- based malware clustering. In *NDSS*.

Chase, W. G., & Simon, H. A. 1973. Perception in Chess. *Cognitive Psychology*, 4, 55-61.

Firdausi, I; Lim, C.; Erwin, A; & Nugroho, AS. 2010. Analysis of Machine learning Techniques Used in Behavior-Based Malware Detection. *In Proceedings of Second Annual Conference of Advances in Computing, Control and Telecommunication Technologies*. 201-203.

Gonzalez, C., Lerch, J. F., & Lebiere, C. 2003. Instance-based learning in dynamic decision making. *Cognitive Science, 27*, 591-635.

Jilk, D. J., Lebiere, C., O'Reilly, R. C., & Anderson, J. R. 2008. SAL: An explicitly pluralistic cognitive architecture. *Journal of Experimental and Theoretical Artificial Intelligence*, 20(3), 197-218.

Kinable, J., Kostakis, O. 2011. Malware classification based on call graph clustering. *J. Comput. Virol*. 7(4), 233–245. DOI 10.1007/s11416-011-0151-y.

Kong, D., & Yan, G. 2013. Discriminant malware distance learning on structural information for automated malware classification. In: *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining,* 1357–1365. ACM, New York, NY, USA. DOI 10.1145/2487575.2488219.

Lebiere, C. 1999. The dynamics of cognition: An ACT-R model of cognitive arithmetic. *Kognitionswissenschaft.*, 8 (1), 5-19.

Lebiere, C., Gonzalez, C., & Martin, M. 2007. Instance-based decision making model of repeated binary choice. In *Proceedings of the 8th International Conference on Cognitive Modeling*. Ann Arbor, Michigan, USA.

Lebiere, C., Gonzalez, C., & Warwick, W. 2009. A Comparative Approach to Understanding General Intelligence: Predicting Cognitive Performance in an Open-ended Dynamic Task. In Proceedings of the Second Artificial General Intelligence Conference (AGI-09). Amsterdam-Paris: Atlantis Press.

Lebiere, C., Pirolli, P., Thomson, R., Paik, J., Rutledge-Taylor, M., Staszewski, J., & Anderson, J. R. 2013. A functional model of sensemaking in a neurocognitive architecture. *Computational Intelligence & Neuroscience.*

Lebiere, C, Bothell, D., Morrison, D., Oltramari, A., Martin, M., Romero, O., Thomson, R., & Vinokurov, J. (2015). Strong Cogsci: Guidance from cognitive science on the design of a test of Artificial Intelligence. Proceedings of the *Beyond the Turing Test Workshop, AAAI-2015*.

Li, P., Liu, L., Gao, D., & Reiter, M. K. 2010. On Challenges in Evaluating Malware Clustering. *Proceedings of the 13th International Symposium on Recent Advances in Intrusion Detection*, Ottawa, Canada.

Mandiant. 2013. APT1: Exposing One of China's Cyber Espionage Units. Mandiant Corp. URL: http://intelreport.mandiant.com/ retrieved 1/21/2014.

Mateos, V., Villagrá, V. A., Romero, F., & Berrocal, J. 2012. Definition of response metrics for an ontology-based Automated Intrusion Response Systems. *Computers & Electrical Engineering,* 38(5), 1102-1114.

Miller, G. A. 1956. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review,* 63 (2), 81–97.

Oltramari, A., Vinokurov, Y., Lebiere, C., Oh, J., & Stentz, A. 2014. Ontology-Based Cognitive System for Contextual Reasoning in Robot Architectures. *In 2014 AAAI Spring Symposium Series.*

Perdisci, P., & ManChon, U. 2012. VAMO: Towards a Fully Automated Malware Clustering Validity Analysis. In *Proceedings of the 28th Annual Computer Security Applications Conference.*

Reitter, D., & Lebiere, C. (2010). Accountable Modeling in ACT-UP, a Scalable, Rapid-Prototyping ACT-R Implementation. In *Proceedings of the 2010 International Conference on Cognitive Modeling.*

Rutledge-Taylor, M., Lebiere, C., Vinokurov, Y., Staszewski, J., & Anderson, J. R. 2011. Bridging the gap: A neurally plausible functional model of sensemaking. In Proceedings of *Biologically Inspired Cognitive Architectures,* 331-340.

Sanner, S., Anderson, J. R., Lebiere, C., & Lovett, M. 2000. Achieving efficient and cognitively plausible learning in backgammon. *In Proceedings of the Seventeenth International Conference on Machine Learning,* 823-830. San Francisco: Morgan Kaufmann.

Sikorski, M., Honig, A. 2012. Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software, 1st edn. No Starch Press, San Francisco, CA, USA.

Taatgen, N., Lebiere, C. & Anderson, J.R. 2006. Modeling paradigms in ACT-R. In Sun, R. (Ed) *Cognition and Multi-Agent Interaction: From Cognitive Modeling to Social Simulation.* NY, NY: Cambridge University Press.

Tamersoy, A., Roundy, K. A., & Horng, D. P. 2014. Guilt By Association: Large Scale Malware Detection by Mining File-Relation Graphs". In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD) 2014.* New York City, NY.

Thomson, R. & Lebiere, C. 2013. Constraining Bayesian Inference with Cognitive Architectures: An Updated Associative Learning Mechanism in ACT-R. In *Proceedings of the 35th Annual Conference of the Cognitive Science Society*. Berlin, Germany.

Vinokurov, Y., Lebiere, C., Herd, S. A., & O'Reilly, R. C. 2011. A Metacognitive Classifier Using a Hybrid ACT-R/Leabra Architecture. *Lifelong Learning AAAI Workshop*.

Wallach, D. & Lebiere, C. 2003. Conscious and unconscious knowledge: Mapping to the symbolic and subsymbolic levels of a hybrid architecture. In Jimenez, L. (Ed*.) Attention and Implicit Learning.* Amsterdam, Netherlands: John Benjamins Publishing Company.

The value of time: Dovetailing dynamic modeling and dynamic empirical measures to conceptualize the processes underlying delay discounting decisions.

Stefan Scherbaum (stefan.scherbaum@tu-dresden.de)

Simon Frisch (frisch@psychologie.tu-dresden.de)

Maja Dshemuchadse (maja@psychologie.tu-dresden.de)

Department of Psychology, Technische Universität Dresden

Keywords: dynamic modeling, delay discounting, decision making, cognitive processes

How do people come to a decision when facing conflicting options? In the case of delay discounting (e.g. Frederick, Loewenstein, & O'Donoghue, 2002) – the choice between an immediate but small reward and a delayed but large one – research generated a multitude of descriptive models pinpointing precisely how people devalue and choose rewards across delays (Doyle, 2013). This descriptive and outcome-centered perspective has been complemented recently by models focusing on the process dynamics leading to delay discounting decisions (Rodriguez, Turner, & McClure, 2014). We present work that embraces and adds to this dynamic approach in two ways. First, we demonstrate how dovetailing continuous dynamic modeling and continuous empirical measures (Dshemuchadse, Scherbaum, & Goschke, 2012; Spivey, Grosjean, & Knoblich, 2005) constrains the conceptualization of the processes underlying a decision. Second, we extend the dynamic approach from the intra-trial time scale of single decisions to the inter-trial time scale of sequences of decisions (compare Duran & Dale, 2014) to explore their interacting effects on behavior (Scherbaum, Dshemuchadse, & Kalis, 2008; Scherbaum, Dshemuchadse, Leiberg, & Goschke, 2013). We present a dynamic computational model of delay discounting behavior (Tuller, Case, Ding, & Kelso, 1994; see also O'Hora, Dale, Piiroinen, & Connolly, 2013; van Rooij, Favela, Malone, & Richardson, 2013) that reproduces existing data (Scherbaum et al., 2013) and predicts new behavioral patterns such as a dependence of current choices on choice history or the temporal decay of these choice persistence effects. The model's predictions are validated in three experiments, indicating the complementary value of harvesting decision dynamics at different time scales on the modeling and the experimental side of the investigation of delay discounting decisions.

Acknowledgments

This research was partly supported by the German Research Council (DFG grant SFB 940/1 2014 and DFG grant SCH1827/11 to Stefan Scherbaum).

References

Doyle, J. (2013). Survey of time preference, delay discounting models. *Judgment and Decision Making*, *8*(2), 116–135.

Dshemuchadse, M., Scherbaum, S., & Goschke, T. (2012). How Decisions Emerge: Action Dynamics in Intertemporal Decision Making. *Journal of Experimental Psychology. General*, *142*, 151–185.

Duran, N. D., & Dale, R. (2014). Perspective-taking in dialogue as self-organization under social constraints. *New Ideas in Psychology*, *32*, 131–146.

Frederick, S., Loewenstein, G., & O'Donoghue, T. (2002). Time Discounting and Time Preference: A Critical Review. *Journal of Economic Literature*, *40*(2), 351–401.

O'Hora, D., Dale, R., Piiroinen, P. T., & Connolly, F. (2013). Local dynamics in decision making: The evolution of preference within and across decisions. *Scientific Reports*, *3*.

Rodriguez, C. A., Turner, B. M., & McClure, S. M. (2014). Intertemporal Choice as Discounted Value Accumulation. *PLoS ONE*, *9*(2), e90138.

Scherbaum, S., Dshemuchadse, M., & Kalis, A. (2008). Making decisions with a continuous mind. *Cognitive, Affective, & Behavioral Neuroscience*, *8*(4), 454–474.

Scherbaum, S., Dshemuchadse, M., Leiberg, S., & Goschke, T. (2013). Harder than expected: increased conflict in clearly disadvantageous intertemporal choices in a computer game. *PLoS ONE*, *8*(11), e79310.

Spivey, M. J., Grosjean, M., & Knoblich, G. (2005). Continuous attraction toward phonological competitors. *Proceedings of the National Academy of Sciences of the United States of America*, *102*(29), 10393–10398.

Tuller, B., Case, P., Ding, M., & Kelso, J. A. S. (1994). The nonlinear dynamics of speech categorization. *Journal of Experimental Psychology-Human Perception and Performance*, *20*(1), 3–16.

Van Rooij, M. M., Favela, L. H., Malone, M., & Richardson, M. J. (2013). Modeling the Dynamics of Risky Choice. *Ecological Psychology*, *25*(3), 293–303.

Combining Dynamic Modeling and Continuous Behavior to Explore Diverging Accounts of Selective Attention

Simon Frisch (frisch@psychologie.tu-dresden.de)
Department of Psychology, Zellescher Weg 17
Dresden, Germany

Maja Dshemuchadse (maja@psychologie.tu-dresden.de)
Department of Psychology, Zellescher Weg 17
Dresden, Germany

Thomas Goschke (goschke@psychologie.tu-dresden.de)
Department of Psychology, Zellescher Weg 17
Dresden, Germany

Stefan Scherbaum (scherbaum@psychologie.tu-dresden.de)
Department of Psychology, Zellescher Weg 17
Dresden, Germany

Keywords: dynamic neural fields; mouse-tracking; selective attention; cognitive inhibition

Abstract

Selective attention is paramount for adaptive behavior as it biases information processing towards stimuli that are relevant for achieving our goals. The mechanisms underlying this bias are under debate, however: Whereas one class of models postulates that selective attention solely relies on the amplification of goal-relevant information (e.g. Cohen, Dunbar, & McClelland, 1990), a second class of models deems additional inhibitory processes necessary to suppress distracting stimuli (e.g. Houghton & Tipper, 1994).

Here, we explore the explanatory value of both accounts from a dynamic perspective that focuses on the continuous unfolding of goal-directed behavior over time (see Scherbaum, Dshemuchadse, Fischer, & Goschke, 2010; Spivey & Dale, 2006). We present two variants of a Dynamic Neural Field model (see e.g., Johnson, Spencer, & Schöner, 2008; Sandamirskaya, Zibner, Schneegans, & Schöner, 2013) that incorporate the diverging assumptions regarding the nature of selective attention. Running simulations of an attentional set-switching paradigm with both models, we show that – even though they make similar predictions with regard to discrete markers of performance like response times – the continuous development of response tendencies over the course of single trials differs markedly whether or not inhibitory processes take part in attentional selection.

To test these dynamic predictions empirically, human participants completed the same set-switching paradigm using mouse-tracking as a continuous measure of performance (see e.g., Scherbaum et al., 2010). Comparing modeled and observed behavior revealed clear evidence for the persisting amplification of previous target information but no signs of sustained distracter suppression.

These findings illustrate that dovetailing dynamic computational modeling with continuous measures of behavior can open promising avenues for understanding the mechanisms underlying fundamental cognitive abilities.

Acknowledgments

This research was supported by the German Research Council (DFG-project 1827/11 granted to Stefan Scherbaum and SFB 940/1 2014).

References

Cohen, J. D., Dunbar, K., & McClelland, J. L. (1990). On the control of automatic processes: A parallel distributed processing account of the Stroop effect. *Psychological Review*, *97*(3), 332–361.

Houghton, G., & Tipper, S. P. (1994). A model of inhibitory mechanisms in selective attention. In D. Dagenbach & T. H. Carr (Eds.), *Inhibitory processes in attention, memory, and language*. San Diego, CA, US: Academic Press.

Johnson, J. S., Spencer, J. P., & Schöner, G. (2008). Moving to higher ground: The dynamic field theory and the dynamics of visual cognition. *New Ideas in Psychology*, *26*(2), 227–251.

Sandamirskaya, Y., Zibner, S. K. U., Schneegans, S., & Schöner, G. (2013). Using Dynamic Field Theory to extend the embodiment stance toward higher cognition. *New Ideas in Psychology*, *31*(3), 322–339.

Scherbaum, S., Dshemuchadse, M., Fischer, R., & Goschke, T. (2010). How decisions evolve: The temporal dynamics of action selection. *Cognition*, *115*(3), 407–416.

Spivey, M. J., & Dale, R. (2006). Continuous Dynamics in Real-Time Cognition. *Current Directions in Psychological Science*, *15*(5), 207–211.

Neural Correlates of Cognitive Models

Marcel van Gerven (m.vangerven@donders.ru.nl)
Donders Institute for Brain, Cognition and Behaviour, Radboud University Nijmegen
Montessorilaan 3, 6525 HR, Nijmegen, The Netherlands

Sennay Ghebreab (s.ghebreab@uva.nl)
Cognitive Neuroscience Group, University of Amsterdam
Nieuwe Achtergracht 129, 1018 WS Amsterdam, The Netherlands

Guy Hawkins (guy.e.hawkins@gmail.com)
Amsterdam Brain and Cognition Center, University of Amsterdam
Nieuwe Achtergracht 129, 1018 WS Amsterdam, The Netherlands

Jelmer Borst (j.p.borst@rug.nl)
Dept. of Artificial Intelligence, University of Groningen
Nijenborgh 9, 9747 AG Groningen, The Netherlands

Keywords: Cognitive modeling, fMRI analysis, EEG analysis

Introduction

In recent years, model-based approaches in neuroscience have become more prevalent, providing an answer to the claim that neuroscience is data rich yet theory poor. Indeed, different cognitive models provide predictions that can be empirically validated using neural data, providing new insights into the neural basis of human cognition.

In this symposium four speakers will present new results in the field of model-based cognitive neuroscience. The first two speakers will outline how models that have their roots in machine learning and computer vision can be used to obtain new insights into the neural basis of visual perception. The third speaker addresses how computational models can provide new insights into the neural basis of decision-making. Finally, the fourth speaker complements this work by discussing how neuroimaging data can be used to constrain cognitive models.

This symposium will provide attendants with new insights into this new field of research which promises to bridge the gap between formal models of cognition and their neuronal correlates, as measured by non-invasive imaging techniques.

Brief Description of the Speakers

In the following a brief description of the symposium speakers is given.

Dr. van Gerven is an assistant professor at Radboud University Nijmegen. He is principal investigator of the Computational Cognitive Neuroscience lab and works at the interface between machine learning and cognitive neuroscience. In the past, van Gerven developed new paradigms for brain-computer interfacing based on covert spatial attention and developed new Bayesian techniques for structural and functional connectivity analysis. His current research focuses on the development of statistical approaches for understanding distributed representations in the human brain.

Dr. Ghebreab is an assistant professor at the University of Amsterdam, and head of Studies Social Sciences at the Amsterdam University College. His research combines computational and neural analysis of everyday visual scenes, and aims at understanding what visual information evokes what perceptual and emotional experiences. He teaches several courses on these topics and leads the Brain & Technology Amsterdam lab (BeTA lab).

Dr. Hawkins is a postdoctoral fellow at the Amsterdam Brain and Cognition Center, University of Amsterdam. His research focuses on developing and testing computational and mathematical models of cognitive processes, in particular decision-making. He completed his PhD in experimental and mathematical psychology at the University of Newcastle, Australia. He then commenced a postdoctoral position at the University of New South Wales, Australia, where he used computational approaches to study judgment and decision-making phenomena.

Dr. Borst is a postdoctoral researcher at the Department of Artificial Intelligence, University of Groningen. His research focuses on new neuroimaging analysis techniques, with the goal of providing additional constrains for cognitive computational models. He obtained his PhD in cognitive modeling at the University of Groningen. During his PhD research, he developed computational models of behavioral and neural data of multitasking. Afterwards, he performed post-doctoral research in the lab of Prof. John R. Anderson at Carnegie Mellon University, where he focused on new analysis methods for EEG data.

Symposium Abstracts

Probing Cortical Representations of Naturalistic Stimuli with Deep Learning (Marcel van Gerven)

Recent advances in machine learning have shown that deep learning achieves state-of-the-art performance in visual object recognition. In this talk I outline how we used deep learning to disentangle the functional organisation of the cortical visual stream. Our results show that downstream areas code for features that are also represented in deeper layers of artificial neural networks. Furthermore, the outlined framework can be used as a high-throughput method for analysing how individual stimulus features are represented across the cortical sheet as well as for estimating voxel-level receptive fields. I argue that the marriage of statistical machine learning with cognitive neuroscience yield new insights into human cognition that cannot be easily achieved via more conventional approaches.

FMRI Evidence for Face Recognition by Visual Words (Sennay Ghebreab)

Neuroimaging evidence has shown that a network of face sensitive brain regions underlies the ability of humans to grasp a person's gender, age, race and mood in a single glance. The mechanism to represent faces remains unclear, however. Bag-of-words are effective in the field of computer vision for learning to recognize an object by its type. In this model, a word represents small, mutually similar patches. Images are represented as bags of words by counting the occurrence of each of the thousands different words. We examined the relation between two representations of face images based on a Bag-of-words model and fMRI-responses to these images. The first representation is the similarity distance to the cluster of faces, which we refer too as ordinate bag-of-word representation. The second is the similarity distances with respect to face sub-clusters (males, females, Asians, Africans, Caucasians, children, adults and seniors). Results reveal neural sensitivity in the core-face network (FFA) to ordinate and in the extend face network (aITG) to subordinate bag-of-word face representations. We provide evidence for bag-of-words as a simple yet effective model for face representation in the brain, possible applicable to generic visual object recognition as well.

Behavioural and Neural Evidence for Urgency in Decision-Making (Guy Hawkins)

Most modern accounts of perceptual decision-making assume that evidence is accumulated for one choice option over another until the balance of evidence reaches one of two decision boundaries, triggering a choice. Decision-making models have typically assumed fixed decision boundaries where the amount of evidence required to trigger a decision does not change with time. A more complicated assumption has recently gained popularity in some neurophysiological accounts of decision-making: collapsing boundaries, where decisions are triggered by less and less evidence as time passes. Such dynamic decision boundaries, often interpreted as implementing rising "urgency signals", have attractive normative properties but have not been stringently tested against data. To this end, I will provide an overview of a large-scale analysis of behavioural data from previously published human and non-human primate studies. We found that the use of dynamic decision boundaries depends on task-specific paradigms or procedures, such as extensive task practice or delayed feedback protocols. I will also discuss the neural correlates of urgency in decision-making using an expanded judgment task that induced dynamic decision boundaries in some participants but not others. The amount of urgency evident in individual subject decision strategies was linked with BOLD activity, to investigate the neural bases of urgency. We conclude that various paradigms and procedures can lead decision makers to adopt qualitatively different decision strategies, and individual differences drive the extent to which one uses urgency-related decision strategies.

Using fMRI data to Constrain a Cognitive Model of Working Memory in Multitasking (Jelmer Borst)

Cognitive models are notoriously hard to evaluate. One important requirement for models is that they should be able to predict data of new experiments. However, even if models are capable of predicting reaction times and accuracy data, their complexity often exceeds constraints provided by behavioral data alone. To provide additional constraints, researchers have turned to neuroscience. A prime example of this is the ACT-R cognitive architecture. After a development based on behavioral and eye-tracking data that extends back to the 1970s, in 2003 a mapping was developed from components of the architecture to brain regions. Since then, models developed in ACT-R predict the fMRI BOLD response in several regions of the brain, and can thus be tested and constrained by fMRI data. In this talk, I will show how this approach led us to re-evaluate a model of working memory use in multitasking. Although our initial model matched behavioral data perfectly, a subsequent fMRI experiment indicated that the assumptions underlying the model were incorrect. We adapted the model to both capture the behavioral data, as well as the neuroimaging data. On the basis of this example, I argue that neuroimaging data can provide valuable constraints for cognitive models.

The Role of Simple and Complex Working Memory Strategies in the Development of First-order False Belief Reasoning: A Computational Model of Transfer of Skills

Burcu Arslan (barslan.cogs@gmail.com), Stefan Wierda (wierda.stefan@gmail.com), Niels Taatgen (N.A.Taatgen@rug.nl), Rineke Verbrugge (L.C.Verbrugge@rug.nl)

Institute of Artificial Intelligence, University of Groningen, P.O. Box 407, 9700 AK Groningen, The Netherlands

Abstract

In their fourth year, most children start to understand that someone else might have a false belief, which is different from the reality that the children know. The most studied experimental task to test this development is called the first-order false belief task. What kind of prior cognitive skills help children to pass the false belief task? There are hundreds of correlational studies that have shown that language and executive functions (such as inhibition and working memory) play a role. Moreover, several training studies have shown the importance of language and inhibition in the development of false belief reasoning. However, to the best of our knowledge there has been no training study (with normally developing children) to investigate the role of working memory strategies in the development of false belief reasoning.

We present here a computational cognitive model to investigate transfer from working memory strategies to false belief reasoning. For this reason, in addition to the false belief task, we constructed two tasks that children encounter in their daily life: a pencil task (simple working memory) and a marble task (complex working memory). Our simulation results confirm our hypothesis that there is more transfer from the marble task to the first-order false belief task than from the pencil task to the first-order false belief task, because of the more complex working memory strategies that appear to be necessary in the false belief task. The results of our simulations suggest conceptual predictions to be tested experimentally.

Keywords: Theory of Mind; False Belief Reasoning; Working Memory; Transfer; Cognitive Modeling; Actransfer.

Introduction

Children's development of reasoning about other people's representational mental states such as beliefs, desires and knowledge has been one of the most studied areas in developmental psychology. In order to conclude that an agent has such a theory of mind (ToM, Premack & Woodruff, 1978), Dennett (1978) argued that it is necessary to test whether the agent can correctly attribute a false belief to another agent. Since then, the explicit false belief task (Wimmer & Perner, 1983) has become one of the most commonly used tasks that verbally tests children's ToM. In the explicit first-order false belief task, children are required to make and report a decision about another person's mental state while they know the real situation, which happens to be different from the other person's false belief. Various studies have shown that children cannot pass the explicit first-order false belief tasks until the age of four (Wimmer & Perner, 1983; Wellman, Cross & Watson, 2001).

One of the most commonly studied explicit first-order false belief tasks is called the unexpected location change task. In this task the story goes more or less as follows: 'Sally and Anne are in the room. Sally puts her chocolate into the basket. After that, she leaves the room. Anne takes the chocolate from the basket and puts it into the box and she also leaves the room. Later, Sally comes back to the room.' The first-order false belief question is "Where will Sally look for the chocolate?" If a child correctly reasons about Sally's mental state, s/he reasons that because Sally did not see Anne taking the chocolate from the basket and putting into the box, Sally will look for the chocolate in the place where she last saw it—thus, the child would answer that Sally will look in the basket.

Interestingly, until the age of 4, children make systematic errors by reporting the real location of the chocolate, which is the box in the above story. This phenomenon is called 'reality bias' (Mitchell et al., 1996). Previous studies of the explicit false belief task showed that 3-year-old children's accuracy is around 30%, 4-year-olds' accuracy is around 50%, 6-year-olds' accuracy is around 80%, and finally around the age of 8, children's performance is at ceiling, similar to adults' performance (Wellman, Cross & Watson, 2001). According to the 'reality bias' view, in order to give correct answers, children should inhibit their own response and take into account others' perspectives.

What kind of cognitive skills are required for children to overcome their 'reality bias' and pass the explicit first-order false belief task? It is a matter of debate whether the development of first-order ToM is purely a matter of conceptual change. In fact, it has been shown that other cognitive factors contribute to the development of first-order false belief reasoning. Several studies have examined the so-called 'far transfer' of skills by training children with different cognitive tasks and investigating whether children's performance on the first-order false belief task has improved or not after the training. Those studies revealed that there is indeed a far transfer of skills from language (Hale & Tager-Flusberg, 2003) and inhibition (Kloo & Perner, 2003) to first-order false belief reasoning. We believe that the working memory strategies that children use also contribute to the development of false belief reasoning. The important role of working memory for first-order false belief reasoning has already been shown by

correlational studies (Gordon & Olson, 1998; Hughes, 1998; Keenan, Olson, & Marini, 1998). Moreover, we have evidence for a significant effect of the complex working memory task but not the simple working memory task in second-order false belief reasoning (Arslan, Hohenberger, &Verbrugge, *submitted*). However, there has so far been no experimental training study focused on the role of working memory strategies in the development of first-order false belief reasoning.

Training studies need more time and effort than correlational studies. For this reason, constructing computational cognitive models to predict what kind of skills might be transferred to another domain (far transfer) is an effective way of designing an appropriate training study. There have been a few computational models of the development of explicit false belief reasoning (Wahl & Spada, 2000; Triona, Masnick & Morris, 2002; Bello & Cassimatis, 2006; Hiatt & Trafton, 2010; Arslan, Taatgen & Verbrugge, 2013). However, none of those models are aimed to predict and explain far transfer from daily life tasks to explicit false belief reasoning.

In the current study, we aim to investigate the possible transfer of cognitive skills from working memory strategies that children use in their daily-life tasks to first-order false belief reasoning by constructing a computational cognitive model that helps us to make more precise predictions. To investigate the role of working memory strategies, we modeled one simple working memory task (the pencil task) and one complex working memory task (the marble task) together with the first-order false belief task. The pencil and marble tasks were inspired by Brain Quest game cards for children of ages 5 to 6 (http://www.brainquest.com/) and they differ from each other in terms of the complexity of the working memory strategies required to solve them (see the sections "A cognitive model of the pencil task" and "A cognitive model of the marble task" for details). We hypothesized that there would be more transfer from the marble task to the first-order false belief task than from the pencil task, because of the more complex working memory strategies required by the marble task, which are also necessary in the false belief task.

In order to model transfer from the pencil and the marble tasks to first-order false belief reasoning, we modeled the tasks using Actransfer (Taatgen, 2013). Actransfer implements the primitive elements theory (Taatgen, 2013) of the nature and transfer of cognitive skills. Actransfer builds on the symbolic computational cognitive architecture Adaptive Control of Thought–Rational (ACT-R; Anderson, 2007). The Actransfer architecture uses ACT-R modules, buffers and mechanisms such as production compilation (Taatgen, 2002).

The primitive elements theory (Taatgen, 2013) breaks down the complex production rules typically used in ACT-R models into the smallest possible elements (PRIMs) that move, compare or copy information between modules. There is a fixed number of PRIMs in the Actransfer architecture. When PRIMs are used often over time, production compilation combines them to form more complex production rules. While those PRIMs may have some task-specific elements, PRIMs also have task-general elements that can be used by other tasks. Transfer occurs if two tasks have common task-general elements: One task can benefit from another trained task because of the already compiled production rules that are learned through production compilation. Taatgen (2013) showed the predictive power of Actransfer by modeling a variety of transfer experiments such as text editing (Singley & Anderson, 1985), arithmetic (Elio, 1986), and cognitive control (Chein and Morrison, 2010).

In the following sections, we will explain our Actransfer models in detail, present the results of the simulations and discuss our findings.

A Cognitive Model of the First-order False Belief Task

Our Actransfer model for the first-order false belief task was inspired by Arslan, Taatgen and Verbrugge's (2013) ACT-R model and Wierda and Arslan's (2014) Actransfer model of first- and second-order false belief reasoning. A simulated storyteller presents the first-order false belief story to our model. The way we implemented this is by updating the perceptual buffer every 4 seconds with new story facts. For each picture in the story, the storyteller tells what happens in that particular picture. The model "listens" to the story and stores what happened in each picture in its declarative memory. The pictures that have actions related to changing the location of the object of interest are chained together in chronological order. Adding a pointer that refers to the previous picture fact realizes the chaining of the picture facts. Also, all related action facts are linked in a similar manner with the corresponding picture fact.

At the end of the story, the storyteller presents the model with a first-order false belief question ('Where will Sally look for the chocolate?'). First, the model creates a first-order chunk in declarative memory that represents the first-order false belief question ("Where will Sally look for the chocolate?"). Next, the model creates a zero-order chunk that represents the corresponding zero-order question ("Where is the chocolate?") by breaking up the first-order false belief question. The model keeps a reference to the zero-order chunk in working memory, which in turn has a pointer towards the first-order chunk. After the question is presented, the model uses two strategies to reason about the question. The first strategy is a memory strategy in which the model always tries to retrieve a picture fact that has an action related to the object's location change. It then looks at that picture when remembering facts about it, such as "Anne put the chocolate into the box". The second strategy is a perception strategy, which is used whenever the model has forgotten the story facts. The model looks at each picture in detail and extracts the story facts from the picture. Below, we present the details of these two strategies (memory and perception) in detail.

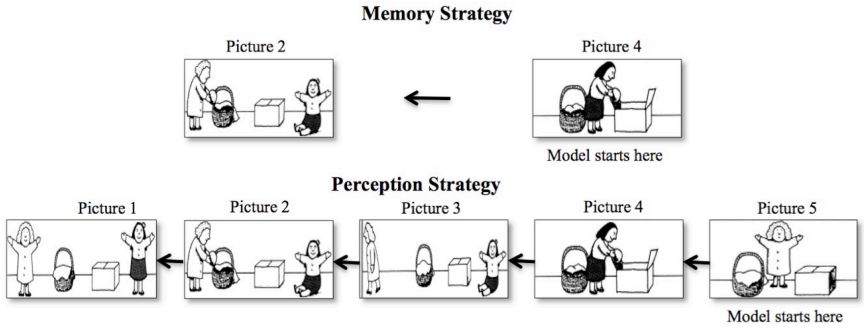


Figure 1. The order of the pictures that the false belief model attends in memory and perception strategies

The memory strategy

The *memory strategy* is the first strategy that the model uses. The model tries to retrieve what was the last picture in which an action happened that was related to the location of the object. If it retrieves that picture fact, it then tries to remember what exactly happened in that picture (for example, a location change of the chocolate). If the model successfully remembers that Anne put the chocolate into the box, it puts the location of the chocolate ("the box") in its working memory and then tries to recall the question. First, the zero-order question is retrieved by the reference that is kept in working memory. If the zero-order chunk does not point to a first-order chunk, the model gives an answer by reporting the location from its working memory ("the box"). However, in this particular task, the actual question put to the model is the first-order false belief question.

Thus, the model then tries to recall the first-order question ("Where will Sally look for the chocolate?"). If it retrieves the first-order question, it checks whether the person in the question performed the action in that picture. Because it was not Sally but Anne who put the chocolate into the box and Sally is absent in the picture, the model tries to retrieve another picture fact at which another action towards the object happened and again it tries to recall what exactly happened in that picture (Figure 1). This process continues until the person who moved[1] the chocolate is the same person who is mentioned in the question.

If the model's run-time passes a preset threshold, the model stops reasoning and answers whatever it currently has in working memory. In this way, we simulate that the model gives up for whatever reason (for example, it takes too long or it gets distracted). As a result, the model will at first give either no answer at all or a zero-order answer. Note that this is because the model first stores the most recent location of the chocolate in its working memory, which corresponds to the zero-order answer ("the box"). When the model reaches the part of the story where the first-order answer ("the basket") can be found, this location will be stored in working memory and the model starts giving the correct first-order answer.

[1] We used the action of moving the chocolate, but the model could also easily be adapted for seeing.

The perception strategy

In our behavioral study (Arslan, Verbrugge, Taatgen, & Hollebrandse, 2014), we have successfully trained 5-6 year old children to pass the second-order false belief tasks. We experienced that on most occasions, children look back in the pictures. Similarly, our model uses the *perception strategy* by looking at the pictures in more detail if it fails to apply the memory strategy because it has forgotten some of the facts of the story as told by the storyteller. In the perception strategy, the model first focuses its attention at the most recently seen picture and inspects whether there is an action related to the salient object in the picture. If there is a person present in the picture, it checks whether this person performed an action or not. Subsequently, it creates a new action fact about the picture in memory and starts to reason with those newly created chunks in the same way as in the memory strategy.

Note that both the perception strategy and the memory strategy use almost the same mechanism for reasoning about the question. The difference is that the irrelevant pictures for finding the answer are skipped in the memory strategy, whereas every picture has to be inspected in the perception strategy (see Figure 1). This is because the memory strategy broke down, and the model cannot immediately recall Picture 4 and subsequently Picture 2 of the false belief story (see Figure 1) at which there are actions related to the location of the object.

A Cognitive Model of the Pencil Task

As we mentioned above, we modeled one simple working memory task, the pencil task, and one complex working memory task, the marble task. In the former task, the goal is to count the total number of yellow and green pencils in a group of blue, red, yellow and green pencils (Figure 2). We modeled this task as follows. The model first looks at a pencil that is in its perceptual buffer. If the color of the pencil is blue or red, it focuses its attention to another pencil. This procedure is repeated until the model finds a yellow or green pencil. It then initializes counting by retrieving a counting fact from its declarative memory and copying the retrieved number to the working memory. It keeps on searching pencils until it finds another yellow or green pencil. When it finds one of those, the counter in

working memory is updated by retrieving the next counting fact. After attending all pencils, the model reports the total number of yellow and green pencils. As becomes clear from this explanation, this task does not need any complex working memory strategies. It simply uses one slot in the working memory buffer and it updates that slot whenever it is necessary.

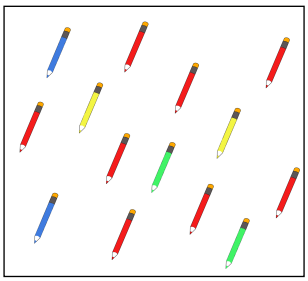


Figure 2. The pencil task (simple working memory)

A Cognitive Model of the Marble Task

The goal of this task is to find, out of a small number of bags of marbles, the two bags that contain the same number of marbles of the same color (Figure 3). Our model uses a strategy that focuses on one color in a bag and counts that color of marbles in each bag until finding a bag that shares the same number of that color. We assume that this is one of the strategies[2] that children use in general. Because we are interested in comparing a complex working memory strategy with a simple one, the strategy that we used for modeling will suffice for our purposes.

The model starts by looking at the first bag and retrieving a color fact from its declarative memory to count the marbles of that color. For example, if the model retrieves the color red, it copies red to one of four working memory slots. At the same time it copies the identity of the bag (Bag-1) in another working-memory slot to report it back when necessary. Then, it looks at the first marble that is in its perceptual buffer, which is blue in our example. Since blue is not the same as the color that is in working memory (red), the model focuses its attention to another marble and repeats that procedure until it finds a marble that matches the color in working memory. After it finds a red marble, it initializes counting by requesting the retrieval of a counting fact from its declarative memory and copying the retrieved number to a third working memory slot. The model then updates that counting slot if it attends another red marble.

Once all marbles of the current color in the current bag (Bag-1) are counted, the model tries to remember if it has already seen another bag that has the same number of marbles of the same color. In the example, because it is the first bag, the model cannot remember a bag that has the same number of red marbles and focuses its attention on another bag to continue to count the red marbles. It carries out the same procedures for the second and the third bags.

After counting all the red marbles in all bags and not remembering any bags that have the same number of red marbles, the model creates a new working memory chunk by emptying all its slots except the slot that has the current color (red). This process also consolidates all information present in working memory and thus creates a new chunk in declarative memory that can be retrieved later on—effectively it remembers which bags it has seen with how many marbles of a given specific color.

Later, it repeats the procedures above by retrieving another color from its declarative memory. Let's say the color blue is retrieved this time. The model counts the blue marbles in the first and second bags, and checks if they have the same number of blue marbles. Because this is not the case, it moves its attention to the third bag and counts the blue marbles. At this point the model can successfully retrieve the first bag with the same number of blue marbles, which is 1, from its declarative memory. Finally, it gives an answer by reporting the first and third bag.

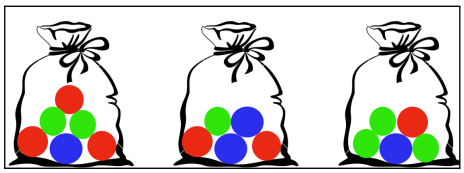


Figure 3. The marble task (complex working memory)

Results

In order to investigate transfer from the simple working memory task (pencil) and the complex working memory task (marble) to the first-order false belief task, we ran simulations in three conditions. In the first condition (FB-only), we ran 100 simulations of a child doing the first-order false belief task 100 times (thus, a total of 100 × 100 = 10,000 trials were simulated).

In the second condition (Marble-FB), we first ran the marble task for 10,080 minutes (24 hours in 7 days) in ACT-R's time. The model would perform as many trials as it could possibly do within that time. Subsequently, the model performed 100 trials of the first-order false belief task. This condition was also simulated 100 times, simulating 100 children.

In the third condition (Pencil-FB), we followed the same protocol as in the second condition but first we ran the pencil task instead of the marble task. Table 1 shows the mean and the standard deviations of the number of simulations for each task. As can be understood from Table 1, the model could squeeze more trials of the pencil task than trials of the marble tasks into the 10,080 minutes. After all, each trial of the marble task, in which several numbers of objects need to be compared, takes much more time than the pencil task, which just involves counting an easily recognizable subset of objects. Therefore, the model has much more previous experience as expressed in number of

[2] Another possible strategy would be focusing on a bag and counting the number of marbles of all its colors, and repeating this procedure until another bag has the same number of a color with the bag in focus. Because both strategies use similarly complex working memory strategies, this would probably not change the simulation results of transfer.

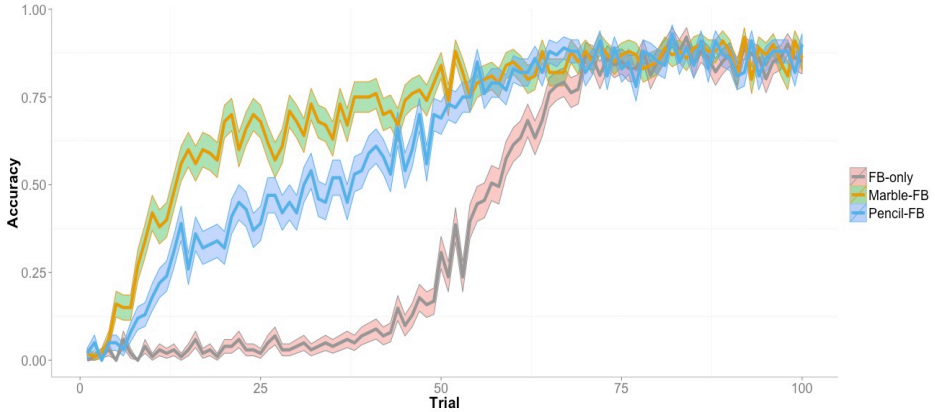


Figure 4. The results of the simulations of 100 trials averaged over 100 runs representing 100 children. The FB-only condition represents 100 trials of false belief task simulation only. The Marble-FB condition represents 100 trials of false belief task simulation after 10,080 minutes (ACT-R time) of training on the marble task. The Pencil-FB condition represents 100 trials of false belief task simulation after 10,080 minutes (ACT-R time) of training on the pencil task.

trials in the pencil task before we run the false belief task model compared to if it is first trained with the marble task. However, as mentioned above, the amount of exposure as expressed in seconds is equal for both tasks.

Table 1. The mean and the standard deviations of the number of simulations for each task

Task	Mean	Standard Deviation
FB-Only	100	0
Marble-FB	217	8.6
Pencil-FB	850	21.0

Figure 4 shows the results of the simulations. In the FB-only condition, in which the model starts without any prior knowledge other than the PRIMs as described in the Introduction, the first-order false belief task model gives the zero-order answer ("reality bias") by reporting the real location of the chocolate (i.e., "the box") until around the 60th trial. After that, it gives the correct answer for the first-order false belief question (i.e., "the basket").

In the Marble-FB condition, in which the first-order false belief task model experienced the prior practice of the marble task, the model starts to give the correct answer much earlier, around the 15th trial. Finally, in the Pencil-FB condition, the model starts to give the correct answer for the first-order false belief question around the 35th trial, which is earlier than in the FB-only condition, but later than in the Marble-FB condition.

Discussion

Our goal was to investigate the role of working memory (WM) strategies in the development of first-order false belief reasoning. In order to achieve this goal, we modeled two real life examples, the pencil task and the marble task, corresponding to a simple and a complex working memory strategy, respectively, by using the cognitive architecture Actransfer.

In agreement with the previous behavioral studies that have shown the correlation between working memory and the development of first-order false belief reasoning (Gordon & Olson, 1998; Hughes, 1998; Keenan, Olson, & Marini, 1998; see Arslan, Hohenberger & Verbrugge, forthcoming for second-order false belief reasoning), our results show that having an experience with tasks that need working memory strategies contribute to this development. Because more complex working memory strategies are needed in our first-order false belief task model than a simple strategy that needs to just update the WM, we predicted that there would be more transfer from the marble task (complex working memory) to the first-order false belief task than from the pencil task (simple WM) to the first-order false belief task. The results confirm our hypothesis.

The first-order false belief task model learns to pass the task faster when it has a prior experience of a task that needs simple or complex WM strategies. This result is straightforward, as we compare the simulations with prior knowledge to a model that has no prior experience at all. More interestingly, the model that was first trained in the marble task, which required complex working memory strategies, mastered the first-order false-belief task much faster—even though the model was able to do fewer trials of the marble task in a given time period ($M_{\text{no of simulations}}$ =217, SD= 8.6) than the model that was first trained in the pencil task, which required simple WM strategies ($M_{\text{no of simulations}}$ =850, SD= 21.0). Note that the amount of exposure to both models was similar in terms of time, as stated above. Together with the experimental training studies that we mentioned in the Introduction, our work implies that passing false belief tasks is not a skill acquired through maturation, but by experience.

Future directions

Although the amount of exposure-time in the Marble-FB and the Pencil-FB conditions was the same, one could argue that it is the general complexity of the marble task (complex

working memory), which causes the transfer to the false belief task. In addition to comparing the marble task to the pencil task (simple working memory), including a third task that has the same complexity as the marble task but that does not require complex working memory strategies might be a better control condition. Also, finding a task to model that has the same complexity as the first-order false-belief task but without the need of working memory might be worthwhile.

The results of our simulations suggest conceptual predictions that should be tested experimentally in experiments with 3-4 year old children.

Acknowledgments: We are grateful to the Netherlands Organization for Scientific Research for Vici grant NWO-277-80-01, awarded to Rineke Verbrugge.

References

Anderson, J. R. (2007). *How Can the Human Mind Occur in the Physical Universe?* New York: Oxford University Press.

Arslan, B., Verbrugge, R., Taatgen, N. & Hollebrandse, B. (2014). Teaching children to attribute second-order false belief: A training study. In Szymanik, J. & Verbrugge, R. (eds.), *Proceedings of the Second Workshop Reasoning About Other Minds: Logical and Cognitive Perspectives,* CEUR Workshops Proceedings, 1208, 1-5.

Arslan, B., Taatgen, N., Verbrugge, R. (2013). Modeling developmental transitions in reasoning about false beliefs of others. In R. West & T. Stewart (eds.), *Proceedings of the 12th International Conference on Cognitive Modeling*, Ottawa: Carleton University.

Arslan, B., Hohenberger, A. & Verbrugge, R. (submitted). The role of language and memory in the development of second-order theory of mind.

Bello, P. & Cassimatis, N. (2006). Developmental accounts of theory-of-mind acquisition: Achieving clarity via computational cognitive modeling. *Proceeding of the 28th Annual Meeting of the Cognitive Science Society*, 1014-1019. Vancouver: Cognitive Science Society.

Chein, J. M., & Morrison, A. B. (2010). Expanding the mind's workspace: Training and transfer effects with a complex working memory span task. *Psychonomic Bulletin & Review*, 17, 193–199.

Dennett, D. C. (1978). Beliefs about beliefs. *Behavioral and Brain Sciences*, 1, 568-570.

Elio, R. (1986). Representation of similar well-learned cognitive procedures. *Cognitive Science*, 10, 41–73.

Gordon, A. C. L., & Olson, D. R. (1998). The relation between acquisition of a theory of mind and the capacity to hold in mind. *Journal of Experimental Child Psychology*, 68, 70–83.

Hale, C., M., & Tager-Flusberg, H. (2003). The influence of language in theory of mind: A training study. *Developmental Science*, 6 (3), 346-359.

Hiatt, L. M. & Trafton, J. G. (2010). A cognitive model of theory of mind. In D. D. Salvucci & G. Gunzelmann (Eds.), *Proceedings of the 10th International Conference on Cognitive Modeling*, 91-96. Philadelphia, PA: Drexel University.

Hughes C. (1998). Executive function in preschoolers: Links with theory of mind and verbal ability. *British Journal of Developmental Psychology*, 16, 233–253.

Keenan, T., Olson, D. R., & Marini, Z. (1998). Working memory and children's developing understanding of the mind. *Australian Journal of Psychology,* 50, 76-82.

Kloo, D. & Perner, J. (2003). Training transfer between card sorting and false belief understanding: Helping children apply conflicting descriptions. *Child Development,* 74, 1823-1839.

Mitchell, P., Robinson, E. J., Isaac, J. E. & Nye, R. M. (1996). Contamination in reasoning about false belief: An instance of realist bias in adults but not children. *Cognition*, 59, 1-21.

Pratt, C., & Bryant, P. (1990). Young children understand that looking leads to knowing (so long as they are looking into a single barrel). *Child Development,* 61, 973-983.

Premack, D., & Woodruff, G. (1978). Does the chimpanzee have a theory of mind? *Behavioral and Brain Sciences*, 4, 515-526.

Singley, M. K., & Anderson, J. R. (1985). The transfer of text-editing skill. *International Journal of Man-Machine Studies*, 22, 403–423.

Tager-Flusberg, H., & Sullivan, K. (1994). A second look at second-order belief attribution in autism. *Journal of Autism and Developmental Disorders,* 24, 577-586.

Taatgen, N.A. (2013). The nature and transfer of cognitive skills. *Psychological Review,* 120(3), 439-471.

Taatgen, N.A. & Anderson, J.R. (2002). Why do children learn to say "broke"? A model of learning the past tense without feedback. *Cognition,* 86(2), 123-155.

Triona, L. M., Masnick, A. M., & Morris, B. J. (2002). What does it take to pass the false belief task? An ACT-R model. In *Proceedings of the 24th Annual Conference of the Cognitive Science Society* (p. 1045). Mahwah, NJ: Lawrence Erlbaum Associates, Inc.

Wahl, S. & Spada, H. (2000). Children's reasoning about intentions, beliefs and behaviour. *Cognitive Science Quarterly*, 1, 5-34.

Wellman, H., Cross, D. & Watson, J. (2001). Meta-analysis of theory of mind development: The truth about false-belief. *Child Development,* 72 (3), 655-684.

Wierda, S. & Arslan, B. (2014). Modeling theory of mind in Actransfer. In Szymanik, J. & Verbrugge, R. (eds.), *Proceedings of the Second Workshop Reasoning About Other Minds: Logical and Cognitive Perspectives* , CEUR Workshops Proceedings, 1208, 40-44.

Wimmer, H. M. & Perner, J. (1983). Beliefs about beliefs: Representation and constraining function of wrong beliefs in young children's understanding of deception. *Cognition*, 13, 103–128.

A Two-level Computational Architecture for Modeling Human Joint Action

Jens Pfau (jens.pfau@cgi.com)
CGI Deutschland Ltd. & Co. KG, Space Business Unit, 64295 Darmstadt, Germany

Liz Sonenberg (l.sonenberg@unimelb.edu.au)
Department of Computing and Information Systems
University of Melbourne, Parkville, VIC 3010 Australia

Yoshihisa Kashima (ykashima@unimelb.edu.au)
School of Psychological Sciences
University of Melbourne, Parkville, VIC 3010 Australia

Abstract

We propose a computational architecture of human joint action that accounts for interactions between higher- and lower-level coordination processes. A proof-of-concept implementation of the architecture is used to model the social Simon task, a well known experimental task that reveals an interplay between higher- and lower-level processes. We show that our model is able to generate results aligned with human performance data for four task configurations. This work contributes to an understanding of mechanisms involved in joint actions.

Keywords: Joint Action; Computational Cognitive Model.

Introduction

Coordination during joint actions typically requires collaborative planning as well as fine-grained alignment of movements. Two musicians performing together have to agree on the pieces they are going to perform and on the parts each of them is going to play. During their performance, however, they synchronize their movements to a level of detail that goes well beyond the specification of this overall plan.

Philosophers have long analyzed higher-level collaborative planning processes based on propositional attitudes such as (joint) intentions, plans, goals, and beliefs (e.g. Bratman, 1992; Tuomela, 2000).

Notions of shared intention have also underpinned investigations by psychologists (e.g. Pacherie, 2011), though empirical psychological research has put much emphasis on the role of lower-level mechanisms of coordination in joint action such as direct perception-action links (e.g. Wolpert et al., 2003; Haazebroek et al., 2011) and it has been noted that integrative perspectives in psychology that represent the interplay between higher- and lower-level coordination processes are less well studied (Knoblich et al., 2011). Psychological studies suggest that coactors develop shared task representations, i.e. they tend to represent their partners' part of the joint action even if this is not required for successful performance (Sebanz et al., 2005), but there remains debate about the nature and detail of what is shared (Knoblich et al., 2011).

Complementing these philosophical and psychological perspectives, a large body of computational work formalizes models of joint action and collaborative planning, (e.g. Grosz & Kraus, 1996; Rao & Georgeff, 1995; Tambe, 1997) as well as models of lower-level coordination processes, (e.g. Hurley, 2008; Wolpert et al., 2003). Although these latter models often seek to explain the emergence of higher-level coordination from lower-level processes, they are not as powerful in representing collaborative plans as are the former models based on higher-level propositional attitudes.

Computational approaches to cognitive modelling can play many roles as discussed by Sun (2009). In particular, computational cognitive models of joint action can inform applied research on human-robot interaction (e.g. Haazebroek et al., 2011; Vesper, 2013, p. 146), a motivation we share.

Multilayer computational cognitive models have been studied for some time (see Thagard, 2012), and examples are mentioned in related work below. We build on such work to pose a specific set of building blocks that can account for observed phenomena in joint action. We describe a computational architecture that includes interactions between collaborative planning based on propositional attitudes and lower-level coordination processes. In gathering evidence in support of the proposed architecture, we draw on theoretical and empirical work on human joint action. We provide a proof-of-concept implementation and simulate a particular experimental task—the *Simon task* (Simon & Rudell, 1967)—that, together with its social variant (Sebanz et al., 2005), has revealed interactions between higher-level planning and lower-level coordination processes. We demonstrate that our model can account for empirical results obtained from different conditions of the Simon task.

Next we describe our architecture. We then describe its (partial) implementation as a model for the social Simon task, present our analysis and a short comparison with related work, and briefly conclude the paper.

Architecture

Figure 1 shows an overview of our architecture, which is composed of two levels. At the lower *perception-action level* perceptual input is received from the environment and mental action representations are translated into muscular movements. Shared representations for perception and action based on common coding theory (Prinz, 1997) support the engagement in joint actions. At the upper *intentional level* practical reasoning operates on higher-level mental attitudes. The distinction between these levels is not crisp and only adopted to guide the discussion and not to make strict distinctions.

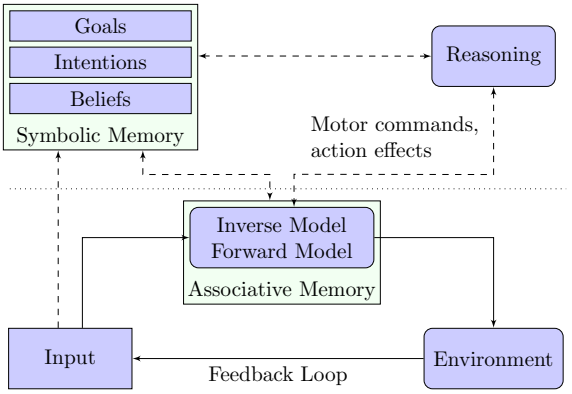


Figure 1: The levels of the architecture. Rectangles denote representations, rectangles with rounded corners denote processes. Solid lines show the feedback loop at the perception-action level. Dashed lines represent information flow.

Two Memory Systems

We employ a dual-process model of a fast, automatic, and subconscious mode of processing and a slow, controlled, and conscious mode of processing. Following common practice (Smith & DeCoster, 2000), we assume that the first mode of processing operates on a *sub-symbolic memory* and the second mode of processing operates on a *symbolic memory*. Symbolic memory and its processing mode enable the processes of the intentional level. Sub-symbolic memory and its processing mode enable the processes of the perception-action level. Via interactions between these two memories, intentional and perception-action level and thereby higher-level and lower-level coordination processes interact.

Building on connectionist models (e.g. Rumelhart, McClelland, & PDP Research Group, 1986), sub-symbolic memory is composed of so called *features*, each of which has a certain *activation level* at any point in time. Activation spreads between features via inhibitory and excitatory connections. Hence, this memory encodes associations between features, and processing exercises these associations by propagating activation. The current activation pattern constitutes a kind of working memory. Learning adjusts connections between features based on how often they are activated simultaneously.

Symbolic memory consists of representations in a language that allows for symbolic reasoning. Processing on this memory amounts to logical inference.

Symbols in symbolic memory are represented by sets of features in sub-symbolic memory. Activating a set of features representing a symbol affects the truth value of that symbol in symbolic memory. However, only those symbols whose features are sufficiently activated are available for reasoning. This constitutes another kind of working memory and allows perceptual context, which is encoded by activated features, to influence which information is accessible for higher-level reasoning. Inferring a formula by symbolic reasoning causes corresponding features in sub-symbolic memory to be activated, which can lead to further activation of features. Mappings between symbols and features are subject to learning.

Actions

An *action* is represented by a *motor command* that produces a movement and by its expected perceptual *action effects*. Both motor commands and action effects are represented by feature sets in sub-symbolic memory. In symbolic memory, these feature sets are represented by symbols. This enables action representations to be shared between the intentional and perception-action levels. In sub-symbolic memory, different action representations can have overlapping features. Likewise, representations of action effects and perceptual input can have overlapping features, as postulated by Prinz (1997). Because the features of the motor command and of its effects are activated at the same time frequently, there is a bi-directional association between motor command and effects in sub-symbolic memory. Consequently, an activation of features associated with the effects of an action also activates the associated motor command and vice versa. This allows for a translation between action effects and motor commands and the planning of actions in terms of their effects. Furthermore, the same action representation can be activated multiple times, yielding an increased activation level; and multiple action representations can be activated at the same time.

Perception-Action Level

We adopt a control system perspective to perception and action for the perception-action level (Hurley, 2008; Wolpert et al., 2003). Two types of internal models are distinguished: *Inverse models* determine the motor command required to cause particular effects. *Forward models* predict the effects of motor commands. Inverse and forward models are implemented by associations between features in sub-symbolic memory.

An appropriate composition of inverse and forward models enables the actor to deal with basic motor control without the involvement of any higher-level cognition. We consider the set-up depicted in Figure 2. We assume that input received from the environment causes an activation of features in sub-symbolic memory.

The inverse model translates effects into a motor command given the current input, which specifies the preconditions that the motor command has to satisfy. In basic execution mode, the effects correspond to a *goal state* that the control system is to achieve (point 2 in Figure 2). The execution of the activated motor command acts on the environment, in turn affecting the input to the inverse model. An internal forward model estimates the effects of the current motor command, supporting the inverse model in its control task. When the inverse model fails to control for the error between the input and goal state, control returns to the intentional level to correct for that error.

The inverse model can be used to generate motor commands (3) for different goal states (2). The forward model can then be used to make predictions for the effects (1) of these motor commands (4). This prediction can also be applied to another actor's actions. In line with common coding theory (Prinz, 1997), we assume that the observed effects of others' movements activate corresponding action effect fea-

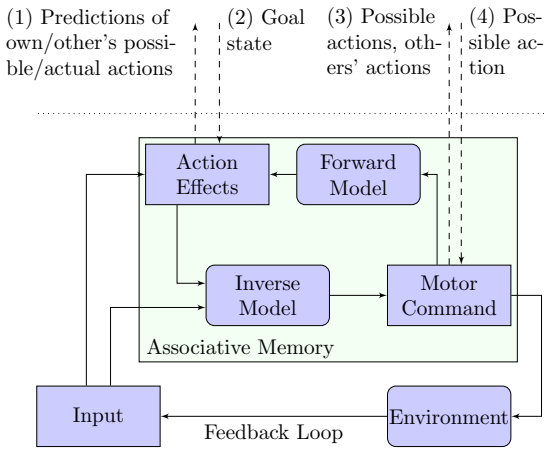


Figure 2: Control and information flow at the perception-action level. Rectangles depict representations, rectangles with rounded corners depict processes. Solid lines represent control flow and dashed lines show the points at which information exchange with the intentional level happens.

tures, causing the respective motor command to be activated via the inverse model. When the extracted motor command (3) is used as an input to the forward model (4), *action simulation* is obtained, which allows to predict the other actor's movement (1) and infer their intentions at a basic level. Given sufficient activation, copying of the other actor's movements results, which facilitates synchronization and imitation. A side-effect is that any input can lead to the activation of action effect features. This represents the interference of observed and planned actions postulated by common coding theory.

Intentional Level

The intentional level implements practical reasoning based on higher-level mental attitudes such as beliefs, goals, and intentions and builds on the perception-action level.

Practical reasoning employs *means-end reasoning* and *intention inference* (using symbolic reasoning and action simulation). Practical reasoning determines the construction of joint intentions and enables, for example, social factors to modulate whether shared task representation are constructed. A *joint intention* is a mental attitude that links the coactors' intentions and practical reasoning to each other's actions and to the overall joint action. Joint intentions drive collaborative planning towards the goal of the joint action (Bratman, 1992). Action representations consist of effects (end) and motor commands (means) as described previously. Goals and intentions are arranged in a hierarchy of alternating levels. A goal corresponding to action effects and an associated intention referring to an action that corresponds to a motor command form the lowest level of this structure. This represents the integration of the perception-action and intentional levels. At higher levels of the hierarchy intentions refer to action plans instead of primitive motor commands.

Goals, intentions, and beliefs are attributed to particular actors, which contrasts with feature activations. If we assume that joint intentions follow the same structure as individual intentions (hierarchies of goals and plans), they integrate well with the practical reasoning of individual actors.

(a) Stimulus-response compatibility (b) (c) Stimulus-response incompatibility

Figure 3: Stimulus-response compatibility and incompatibility in the Simon task. In (a), the stimulus appears on the side of the button that is to be pushed. In (b), the stimulus appears on the other side. Note that buttons are not colored in the actual task.

The Simon Task

In the *individual Simon task* a subject pushes one of two buttons (left or right) depending on a non-spatial attribute of a *stimulus* object appearing on a screen. The non-spatial, *task-relevant* stimulus attribute is typically the color of this object. For example, a subject could be instructed to push the left button when the object is green and the right button when the object is red. It turns out that reaction time increases if spatial, *task-irrelevant* attributes of the object are incompatible with spatial features of the expected *response* (push left or right button). The spatial, task-irrelevant attribute is typically the location of the object on the screen. For example, the object can appear on the left or right side of the screen and be either on the same side as the button that is to be pushed or on the other. In the first case, we talk about *stimulus-response compatibility* and about *stimulus-response incompatibility* in the second case (Figure 3). The increase in reaction time in the stimulus-response incompatibility condition is called the *Simon effect* (Simon & Rudell, 1967). The Simon effect is absent when the subject performs only one part of the task.

In the *social Simon task*, two subjects carry out the Simon task together, i.e. each subject is responsible for one of the two stimulus-response mappings (task rules). A task-irrelevant spatial attribute referring to the other subject's action leads to an increase in reaction time similar to the one in the individual Simon task (Sebanz et al., 2005). Such an increase in reaction time does not occur if the subjects carry out their parts of the task individually. This suggests that subjects corepresent their coactor's action in the joint task (*action corepresentation*). The representation of the coactor's action can be activated by a compatible stimulus feature, causing an action conflict (i.e. a situation where the inappropriate action receives activation which needs to be suppressed to allow the correct action). Consequently a Simon effect is observed.

An increase in reaction time is also observed when a stimulus calls for both subjects to carry out an action at the same time (*task conflict*). The interpretation is that a subject also corepresents the task rule of the partner (*task corepresentation*). If the subject corepresents their coactor's task rule, the associated action is activated when the stimulus triggers that rule's precondition. Like with the action conflict, a task con-

	Conditions			
Setup	No Conflict	Action Conflict	Task Conflict	Action and Task Conflict
Left subject's task	green → left	green → **left**	**right** → left	**left** → **left**
Right subject's task	**red** → **right**	**red** → right	**red** → **right**	**red** → right
Stimulus color/location	red/right	red/left	red/right	red/left

Table 1: The experimental conditions of the social Simon task. By green → left we mean that the subject reacts to a green stimulus by pushing the left button. In the action-conflict condition, there is an action conflict for the right subject. The parts of each task rule activated by the stimulus either because of the relevance to the rule's precondition or an overlap with the features of its action are printed in **bold**.

flict is not observed when the subject performs its part of the task individually. Action and task corepresentation can affect task performance in isolation as well as simultaneously. Results show that the reaction time increase due to task conflicts is larger than the one due to action conflicts. If action and task conflict occur simultaneously, reaction time is more than the sum of the reaction times when action and task conflict occur in isolation. Table 1 lists the different experimental conditions and the conflicts they evoke for the right-hand side subject. The no-conflict and action-conflict conditions mirror the same conditions in the individual Simon task.

For a recent review of research involving the Simon task, and its social variant, refer to Dolk et al. (2014).

Model and Analysis

We describe an implementation of those parts sufficient to model the (right-hand side) subject in the individual and social Simon tasks based on our architecture. We compare the Simon effect in our simulations with empirical data.

Sub-symbolic and symbolic memory provide representations of stimulus features (green color, red color, left position, right position) and of the effects and motor commands of the two available actions (push left and push right). Beliefs and goals of agents and the effects of actions are represented with propositional logic. Figure 4 displays the mapping between the elements of both memories and the associations between features in sub-symbolic memory. We assume mapping and associations have been established by some means a priori. To prevent clutter, we do not show here symbols and features referring to stimulus color.

Sub-symbolic memory contains feature sets representing the goal (action effects) of perceiving the left button being pushed (l, u_1, u_2) and the right button being pushed (r, v_1, v_2). Via threshold units, both feature sets forward activation to respective features representing the motor commands achieving these goals (x_1, x_2, x_3 and y_1, y_2, y_3). This represents the inverse model. Motor command features are part of a fully recurrent auto-associative connectionist network in which activation settles into that previously learnt pattern which is closest to the current input (Rumelhart et al., 1986). By prior learning, strong associations were created between the features of the same motor command. Hence, input to both motor commands leads to the respective feature sets competing for activation. The activation pattern requires some time to settle to a stable state. The feature s represents a stimulus,

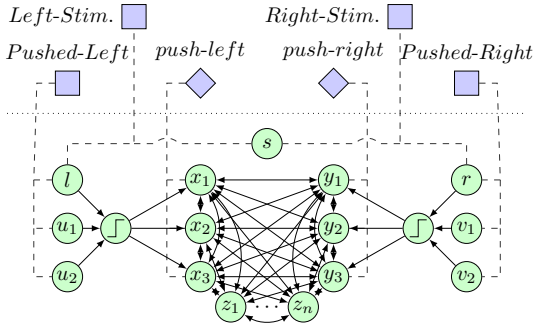


Figure 4: Elements of symbolic and sub-symbolic memory and their relationships. At the intentional level, rectangles denote literals in symbolic memory, diamonds denote actions. At the perception-action level, circles denote features in sub-symbolic memory. Dashed lines show associations between features and symbols, and solid lines with arrows show associations between different features. The threshold symbol indicates that activation through this unit is only propagated if it exceeds a certain threshold.

whose activation together with the feature for right (r) or left (l) or features representing color (not shown in the figure) represents the perception of a stimulus attribute. Note that the features for right and left (r and l) are both part of a set of features representing action effects and a set of features representing a stimulus attribute. This amounts to the common coding of actions and observations. Note that high-level action representations *push-left* and *push-right* are not available a priori but obtained by employing the inverse model.

Features in sub-symbolic memory can receive activation in four ways: (1) when a belief, goal, or intention is created (*attitude representation*); (2) when a stimulus is perceived (*stimulus perception*); (3) when the intentional level employs the inverse model at the perception-action level by providing a goal state to retrieve an appropriate motor command (*action planning*); (4) when the intentional level invokes the execution of a primitive action (*action execution*). Feature activations in sub-symbolic memory due to stimulus perception are translated into corresponding symbols in symbolic memory.

Only together, attitude representation and stimulus perception generate enough activation on the features representing the action effects *Pushed-Left* (*Pushed-Right*) so that this activation is propagated to the features representing the motor command *push-left* (*push-right*). In Figure 4, this is represented by threshold units. For example, representing a goal which refers to *Pushed-Left* is not sufficient for activation to be propagated to the features of *push-left*. Likewise, acti-

vating the $left$ feature is not sufficient to activate the features of *push-left*. Thereby the agent is able to represent and plan with goals without executing an action. We assume that activation due to action planning and execution is sufficient to overcome this threshold, which enables action planning based on inverse models and the execution of motor commands. A motor command is executed once sufficient activation is provided and its features settle into a stable pattern.

The intentional level is driven by a modified BDI interpreter that implements practical reasoning based on goals, intentions, and beliefs (Rao & Georgeff, 1995). Means-end reasoning can be based on symbolic reasoning (as is standard) or the inverse model at the perception-action level can be employed to retrieve a motor command for a given goal state. An agent annotates each goal and intention with the actor(s) that is (are) supposed to hold that attitude. A joint intention consists of a goal that has multiple actors and of its subordinate intentions and goals. An agent can plan for coactors but does not act on intentions that it is not the sole actor of.

There is an individual and a social task. In the individual task, there is a no-conflict (*NoC*) and an action-conflict (*AC*) condition. The social task adds a task-conflict (*TC*) and a both-conflicts (*TCAC*) condition. All conditions can be setup so that only one part of the task is represented (no corepresentation) or both parts (corepresentation). Corepresentation is the default but was manipulated by Sebanz et al. (2005) in the individual and by Hommel et al. (2009) in the social task.

A top goal $Goal_A(SimonTask)$ is provided to the agent where A is the set of agents performing the task, e.g. $\{you, me\}$. The agent is equipped with a complex action *SimonTask*, which can be performed to achieve the top goal and itself evokes subgoals according to the experimental condition, e.g. $Goal_{\{you\}}(Green\text{-}Stimulus \Rightarrow Pushed\text{-}Left)$ and $Goal_{\{me\}}(Red\text{-}Stimulus \Rightarrow Pushed\text{-}Right)$. The first subgoal means that if the stimulus is green, the left button is to be pushed by the other agent.

In the action-conflict condition, the agent adopts the top goal and means-end reasoning creates an intention to perform the *SimonTask* action with the other subject. This leads to the adoption of the above mentioned subgoals. With corepresentation the subgoal for the other subject is represented. By representing subgoals, activation is added to the corresponding features of symbols (i.e. *Green-Stimulus*, *Red-Stimulus*, *Pushed-Left*, *Pushed-Right*). This is what corresponds to action and task corepresentation according to Sebanz et al. (2005). Both the other subject's task (encoded in the subgoal) and action (via action effects) is corepresented. However, no activation is propagated to motor command features yet.

A red stimulus is represented on the left. This leads to an activation of the features *s* and *l* and the feature representing red. Now the feature set representing *Pushed-Left* has sufficient input to have activation leak over into the features representing the *push-left* action. Also, corresponding propositions (*Red-Stimulus* and *Left-Stimulus*) are then made true in symbolic memory. Now, the subgoal $Goal_{\{me\}}(Red\text{-}Stimulus \Rightarrow Pushed\text{-}Right)$ needs to be achieved because *Red-Stimulus* is true and *Pushed-Right* false. Means-end reasoning employs the inverse model to obtain an action that can achieve *Pushed-Right*, which we call *push-right* but which was not explicitly available to the agent before. By using the inverse model, activation is added to the feature sets representing *Pushed-Right* and *push-right*.

An intention is created to execute *push-right*, which further adds to the activation of the features representing that action. Then activation is provided to the *push-right* features to execute that action but only after the recurrent motor command network settles into a stable activation pattern. Perceptual input of pushing the button then increases the activation of the *Pushed-Right* features, so that the proposition is made true and the subgoal of this agent deemed achieved.

The time until motor command features settle to a stable activation pattern is an estimate of the response time and hence of the Simon effect, c.f. Haazebroek et al. (2011). Any activation on the incorrect response (*push-left* for the right-hand side subject) increases this time. In the action-conflict condition, the left-side stimulus adds activation to the feature *l*, which is shared with the features representing *Pushed-Left* and hence provides further activation to the features of *push-left*. In the task-conflict condition, means-end reasoning for the other subject's task via the inverse model adds onto the activation of the features representing the *push-left* motor command.

We performed a parameter estimation for this model against the empirically observed reaction times in the different conditions of the Simon task. The goal was to maximize the correlation between the *relative* reaction times observed empirically and the ones observed in our model. The term "relative" here refers to the differences of reaction times between the conditions, reflecting that the Simon effect is an increase of reaction time compared to a baseline condition (the no-conflict condition). Note that the reaction time observed in the no-conflict condition is comparable to the reaction time in all conditions without corepresentation (in the individual and social task); and the observed reaction times in the social no-conflict and action-conflict conditions are comparable to those in the individual task. Therefore the four experimental conditions presented here implicitly represent 12 conditions.

Figure 5 shows the relative reaction times obtained empirically and from the parameter set (9 parameters) that maximizes the correlation between empirical and simulated data. By all reasonable means, this match is very close. In fact, we found a large set of parameters that achieve a correlation of 0.95 or more, which we cannot show here due to space constraints. Suitable parameters cover a significant portion of the parameter space, which suggests that the model is not overly sensitive to any of its parameters.

Related Work

MOSAIC (Wolpert et al., 2003) is a computational model of motor control that relies on forward and inverse models sim-

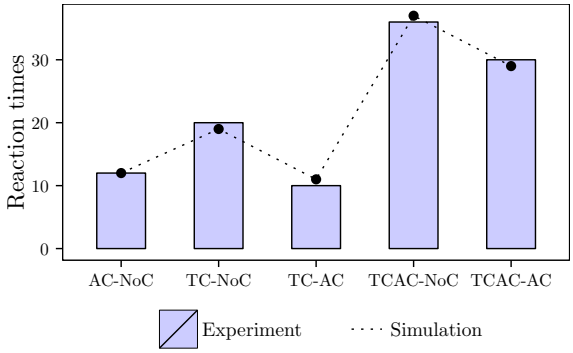


Figure 5: Relative reaction times using the parameter set that maximizes the correlation between empirical and simulated data. Simulated data points are regression-adjusted to the empirical data. Empirical data drawn from Sebanz et al. (2005).

ilar to our perception-action level. In contrast to our intentional level, MOSAIC explains higher-level (collaborative) planning with a hierarchy of control modules.

SCM (Hurley, 2008) is a description of motor control at an intermediary level between neural-level mechanisms and higher-level reasoning. SCM predicts how neural-level mechanisms enable higher-level ones, in particular those for joint action such as imitation and mind-reading. Our motor control models at the perception-action level borrow from SCM.

HiTEC (Haazebroek et al., 2011) is a cognitive architecture of the interplay between perception and action based on common coding theory. Representations of stimuli and action effects share the same set of features. The architecture has been used to represent the Simon task. Without intentional mechanisms, however, higher-level modulations such as social factors cannot easily be represented.

Conclusion

The cognitive mechanisms underlying joint action are not yet well understood. We describe a computational architecture of human joint action that incorporates an interplay between higher- and lower-level coordination processes and have reproduced results of four conditions of the social Simon task. Our model is consistent with the referential coding account of Dolk et al. (2014), that provides a novel approach to analyzing the Simon effect. While explorations with computational models cannot directly shed light on human cognition, c.f. (Sun, 2009), our demonstration contributes to analyses of potential building blocks for mechanisms involved in coordination in joint action – whether it be in purely human, or human-robot interaction contexts.

References

Bratman, M. E. (1992). Shared cooperative activity. *The Philosophical Review*, *101*, 327–341.

Dolk, T., Hommel, B., Colzato, L. S., Schütz-Bosbach, S., Prinz, W., & Liepelt, R. (2014). The joint Simon effect: a review and theoretical integration. *Frontiers in Psychology*, *5*.

Grosz, B. J., & Kraus, S. (1996). Collaborative plans for complex group action. *Artificial Intelligence*, *86*(2), 269–357.

Haazebroek, P., Dantzig, S. van, & Hommel, B. (2011). A computational model of perception and action for cognitive robotics. *Cognitive Processing*, *12*(4), 355–365.

Hommel, B., Colzato, L. S., & Van Den Wildenberg, W. P. M. (2009). How social are task representations? *Psychological Science*, *20*, 794–798.

Hurley, S. (2008). The shared circuits model (SCM): How control, mirroring, and simulation can enable imitation, deliberation, and mindreading. *Behavioral and Brain Sciences*, *31*(1), 1–22.

Knoblich, G., Butterfill, S., & Sebanz, N. (2011). Psychological research on joint action: Theory and data. In B. Ross (Ed.), *The psychology of learning and motivation* (Vol. 54, pp. 59–101). Burlington, MA: Academic Press.

Pacherie, E. (2011). Framing joint action. *Review of Philosophy and Psychology*, *2*(2), 173-192.

Prinz, W. (1997). Perception and action planning. *European Journal of Cognitive Psychology*, *9*(2), 129–154.

Rao, A. S., & Georgeff, M. P. (1995). BDI agents: From theory to practice. In *Proceedings of the first international conference on multi-agent systems* (pp. 312–319).

Rumelhart, D. E., McClelland, J. L., & PDP Research Group. (1986). *Parallel distributed processing*. MIT Press.

Sebanz, N., Knoblich, G., & Prinz, W. (2005). How two share a task: Corepresenting stimulus–response mappings. *Journal of Experimental Psychology*, *31*(6), 1234–1246.

Simon, J. R., & Rudell, A. P. (1967). Auditory S-R compatibility: The effect of an irrelevant cue on information processing. *Journal of Applied Psychology*, *51*(3), 300–304.

Smith, E. R., & DeCoster, J. (2000). Dual-process models in social and cognitive psychology: Conceptual integration and links to underlying memory systems. *Personality and Social Psychology Review*, *4*(2), 108–131.

Sun, R. (2009, June). Theoretical status of computational cognitive modeling. *Cognitive Systems Research*, *10*(2), 124–140.

Tambe, M. (1997). Towards flexible teamwork. *Journal of Artificial Intelligence Research*, *7*, 83–124.

Thagard, P. (2012). Cognitive architectures. In K. Frankish & W. Ramsay (Eds.), *The Cambridge Handbook of Cognitive Science* (p. 50-70). University Press.

Tuomela, R. (2000). Collective and joint intention. *Mind & Society*, *1*(2), 39–69.

Vesper, C. (2013). *Acting together: Mechanisms of intentional coordination*. Unpublished doctoral dissertation, Radboud Universiteit Nijmegen. Available from `http://hdl.handle.net/2066/112295`

Wolpert, D. M., Doya, K., & Kawato, M. (2003). A unifying computational framework for motor control and social interaction. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, *358*(1431), 593–602.

Metacognition in the Prisoner's Dilemma

Christopher A. Stevens (c.a.stevens@rug.nl), Niels A. Taatgen (n.a.taatgen@rug.nl), Fokie Cnossen (f.cnossen@rug.nl)
Department of Artificial Intelligence, Nijenborgh 9
9747 AG Groningen, The Netherlands

Abstract

In this paper, we show ACT-R agents capable of metacognitive reasoning about opponents in the repeated prisoner's dilemma. Two types of metacognitive agent were developed and compared to a non-metacognitive agent and two fixed-strategy agents. The first type of metacognitive agent (opponent-perspective) takes the perspective of the opponent to anticipate the opponent's future actions and respond accordingly. The other metacognitive agent (modeler) predicts the opponent's next move based on the previous moves of the agent and the opponent. The modeler agent achieves better individual outcomes than a non-metacognitive agent and is more successful at encouraging cooperation. The opponent perspective agent, by contrast, fails to achieve these outcomes because it lacks important information about the opponent. These simple agents provide insights regarding modeling of metacognition in more complex tasks.

Keywords: Theory-of-mind; Metacognition; Prisoner's Dilemma; ACT-R

Metacognition in Two-Person Games

Humans can reason about the minds of others and predict their behaviors, a metacognitive ability known as theory of mind (Premack & Woodruff, 1978). A question of great current interest is why humans evolved this ability. One possible reason is that theory of mind allows people to understand and predict the actions of others (McCabe, et al., 2000). People can use this ability to determine whether another person is likely to be cooperative or competitive. Theory of mind might also be used to learn the strategy of an opponent and devise an appropriate counter-strategy (Hingston et al., 2007).

The prisoner's dilemma is a task that embodies the basic conflict between cooperation and competition often found in real-world interactions. It is often used to study how various strategies may help or harm an individual's or group's chances of survival (Axelrod, 1980; Wedekind & Melinski, 1996; Nowak & Sigmund, 1993). However, very little is known about how metacognition impacts performance in this task. In the present work, we develop two cognitive agents that embody different metacognitive strategies. We then compare the performance of these agents against an existing, non-metacognitive agent (Lebiere, Wallach, & West, 2000) and two normative strategies (tit-for-tat: Axelrod, 1980; win-stay-lose-shift: Wedekind & Melinski, 1996).

The Prisoner's Dilemma

The prisoner's dilemma is a 2 x 2 game in which players must choose to cooperate with their opponent (move B) or to defect (move A). This results in one of four possible outcomes. The following is a typical payoff matrix for the prisoner's dilemma game.

		Player 2	
		Cooperate (B)	Defect (A)
Player 1	Cooperate (B)	1,1	-10, 10
	Defect (A)	10, -10	-1, -1

If both players consistently choose to cooperate, then they both will enjoy a positive payoff. However, cooperation is risky, because both players have a temptation to defect. If the opponent defects when a player cooperates, the cooperating player will lose a large number of points. Defection also carries risks. When there is more than one round, opponents may retaliate by defecting in later rounds. The optimal strategy is not obvious and depends on the opponent. Therefore, reasoning about an opponent's goals and predicting their future behavior should provide an advantage (Hingston et al., 2007). To determine if this is true, we developed two cognitive agents that represent different strategies for metacognitive reasoning. We then tested these agents against fixed strategies and a non-metacognitive model.

The ability to reason about others may have implications not only for individual outcomes, but also for collective outcomes. De Weerd, Verbrugge, and Verheij (2013) developed agents with different levels of metacognitive ability and pitted them against one another in a negotiation game. They found that agents that could reason about their opponents' beliefs obtained both greater rewards for themselves and found opportunities for greater collective rewards. In a similar way, metacognitive abilities may improve cooperation in the prisoner's dilemma. Metacognitive agents may have an easier time predicting their opponents, allowing them to know when cooperation is possible.

To evaluate the metacognitive agents presented here, we used a previous agent developed by Lebiere, Wallach, & West (2000) as a baseline. This agent is built within the ACT-R cognitive architecture (Anderson et al., 2004). The agent provides a good fit to human data, but it is not metacognitive because it bases its decisions only on the immediate payoffs of its previous moves. It does not

attempt to learn its opponent's strategy or explicitly reason about its opponent. For this reason, we hereafter refer to it as the self-payoff agent.

We present two types of metacognitive agent: opponent-perspective and modeler. The opponent-perspective agent is inspired by the simulation theory of mind (Gallese & Goldman, 1998; Meltzoff, 2007), which states that people reason about the mental states of another by adopting the other's perspective. On every trial, the agent computes the move that it would make if it was the opponent and then selects its own move accordingly.

The modeler agent is inspired by opponent-modeling agents in the multi-agent systems literature (e.g. Hingston et al., 2007) as well as models of sequence learning in ACT-R (Lebiere & West, 1999). An opponent-modeler agent develops a mental model of the opponent's strategy over time and attempts to predict the opponent's next move probabilistically. Hingston et al. (2007) present an opponent-modeler that can successfully play the prisoner's dilemma against a variety of other agents. However, this agent is not a cognitive agent, and does not attempt to capture the flexibility or variability of human behavior. Our modeler agent extends this approach by using the declarative memory system of ACT-R to create a cognitively plausible, dynamic model of its opponent that can be rapidly updated to handle new information. This approach should be helpful for adapting to new opponents or strategy shifts in current opponents.

Simulations

In the following simulations, we use an instance-based learning approach to allow the agents to adapt to their opponents (Logan, 1988). In this approach, the outcomes of previous trials are encoded as chunks in declarative memory. The agents then attempt to predict outcomes or opponent behaviors by retrieving a chunk from memory that matches the current situation. By updating the contents of declarative memory, the agents can adapt their strategies to suit their opponent.

The likelihood of retrieval of a chunk in ACT-R is determined by its activation level. The more frequently and recently a chunk has been used, the more active it will be. For all simulations, we used the full (non-optimized) learning equation of ACT-R:

$$B_i = \ln\left(\sum_{j=1}^{n} t_j^{-d}\right) + Logistic(0, s)$$

In this equation, n is the number of presentations of chunk i. t_j is the time since the jth presentation. A presentation is either the creation of a chunk or a retrieval of that chunk. d is the rate of activation decay. By default this is set to 0.5. The rightmost term of the equation represents noise added to the activation level. s is an ACT-R parameter that determines the standard deviation of the noise. For all simulations reported here, we use an s value of 0.25, consistent with the value used in Lebiere, et al.'s (2000) model.

In the following simulations, we compare the performance of the two metacognitive agents to the self-payoff agent. The aim was to determine whether metacognitive reasoning can give an agent more robust performance across a variety of agents. To do this, we played all three of these agents against the self-payoff agent and two fixed-strategy agents. We hypothesized that the metacognitive agents would have better overall performance across all opponents because of their greater adaptability.

The strategies of the fixed-strategy agents were based on two normative strategies found in the prisoner's dilemma literature: tit-for-tat (TFT; Axelrod, 1980) and win-stay-lose-shift (WSLS) (Nowak & Sigmund, 1993). Both of these strategies have been shown to provide robust performance against a variety of opponents. There are several variations of the tit-for-tat strategy, but all of the variations tend to copy the previous move of their opponent. We used a strict TFT strategy that always copied the previous move of the opponent. The WSLS strategy, by contrast, continues to make the same move until it loses points, then it changes moves.

The Self-Payoff Agent

This agent is a replication of the model reported in Lebiere, et al. (2000). It was originally designed to account for behavior in the prisoner's dilemma task without resorting to notions of altruism or to long-term payoff calculations. It does not attempt to explicitly reason about its opponent or predict its opponent's behaviors. Instead, it predicts the most likely payoff of each of its possible moves. Then it selects the move associated with the highest payoff.

The self-payoff agent remembers the previous rounds of the game using four declarative memory chunks. Each chunk represents one of the four possible outcomes of the game.

```
A1-A2 isa outcome move1 A move2 A payoff1 -1 payoff2 -1
A1-B2 isa outcome move1 A move2 B payoff1 10 payoff2 -10
B1-A2 isa outcome move1 B move2 A payoff1 -10 payoff2 10
B1-B2 isa outcome move1 B move2 B payoff1 1 payoff2 1
```

The four outcomes are A1A2, A1B2, B1A2, and B1B2. The first letter of the pair represents player 1's move and the second letter represents player 2's move. The move1 and move2 slots represent the moves chosen by players 1 and 2 respectively. The payoff slots contain the number of points both players receive. In every round, the model creates a new outcome chunk in the goal buffer. When the model selects its move, it records it in the move1 slot. At the end of the trial, the opponent's move and the resulting payoffs are also recorded in this chunk.

The self-payoff model uses the relative activation levels of these four chunks to determine the most likely outcome of a given move. It does this by using a set of four

production rules. The first two productions retrieve two outcome chunks, one in which move1 is A and one in which move1 is B. The remaining productions then select move A if payoff A is higher or move B is payoff B is higher.

The self-payoff agent provides a good fit to human data both in the prisoner's dilemma and in other 2 x 2 games (Lebiere et al., 2000). It can account for cooperative behavior because the A1-A2 and B1-B2 chunks should become more active over time (as long as the opponent is not a consistent defector). When these chunks are more active than the other two chunks, the agent determines that cooperation is more profitable than defection.

The Opponent-Perspective Agent

The opponent-perspective agent is an adaptation of the Lebiere et al. (2000) model that attempts to predict the opponent's move by deciding which move it would take in the opponent's place. The opponent-perspective agent stores the same information in memory as the self-payoff agents. But instead of determining its own most likely outcomes, it predicts the most likely outcomes for its opponent. It then predicts that its opponent will select the move with the highest payoff. Based on this prediction, it selects an appropriate response.

Due to the nature of the prisoner's dilemma, there is not an optimal countermove for each possible opponent move. Regardless of an opponent's move, defecting will always lead to a higher immediate payoff than cooperating. Therefore, we designed these agents to use an imitative strategy. That is, the agent will do what it thinks the opponent is going to do this round. If the opponent's behavior can be successfully predicted, then the agent will be able to find opportunities for cooperation without being exploited.

The perspective model uses the same declarative chunks and production rules described above. However, its production rules instead compare the opponent's payoffs rather than its own payoffs. Based on this comparison, the model will predict the opponent's next move and select the same one.

In principle, this agent should be able to perform well against the self-payoff agent and the normative strategies. The metacognitive agent uses the same payoff calculation as the self-payoff agent, and this should make it easier to predict the self-payoff agent's moves. The TFT and WSLS agents calculate their moves differently, but they are all based on the same principle of cautious cooperation. When the opponent cooperates and punishes defection, these agents will tend to cooperate more.

The Modeler Agent

The modeler agent represents a different form of metacognition than the opponent-perspective agents. The modeler attempts to build a mental model in declarative memory to predict the opponent's most likely next move based upon their previous moves. Unlike Hingston et al.'s (2007) opponent-modeler, the modeler makes a specific prediction about the move the opponent is going to make in the current round. Also, because it makes use of ACT-R's declarative memory system, the modeler weighs information from more recent rounds more heavily than less recent rounds. This should afford the agent greater flexibility in its behavior.

Unlike the opponent-perspective agent, the modeler does not make any assumptions about the specific strategy used by the opponent. Instead, it tracks how the opponent responds to each of the four possible outcomes in the game (double-defect, defect-cooperate, cooperate-defect, and double-cooperate). It then predicts that the opponent will make the same move after the outcome appears again.

The memory structure of the modeler agent is different from that of the self-payoff and opponent-perspective agents. The model does not start with any predefined chunks, but after every trial, it will create a new chunk like the following example:

SEQUENCE0 isa sequence move1 A move2 B next-move A

Move1 represents the player's move and move2 represents the opponent's move. Next-move represents the opponent's move on the following round. This chunk represents an instance in which the agent defected and the opponent cooperated; in the next round, the opponent responded with a defection.

Before deciding on a move, the modeler agent will retrieve a previous instance that matches the current situation. For example, after a double-cooperation round, the modeler will attempt to retrieve a chunk in which both move1 and move2 are B. Based on this retrieved chunk, it will predict the opponent's next move. Like the opponent-perspective agent, the modeler will select the same move that it thinks its opponent is going to select. If this prediction turns out to be incorrect, the modeler will create a new chunk to reflect the correct prediction and store it in memory. If the modeler fails to retrieve a similar instance, it will select a move randomly.

Simulation Results

Fifteen simulations were run. The self-payoff, opponent-perspective, and modeler agents were all played against all other agents. Each simulation consisted of 1000 runs of 100 trials. Results were averaged over the runs. A summary of the performance can be found in Tables 1 and 2.

Self-payoff

The self-payoff agent behaved consistently with the version previously reported (Lebiere et al., 2000). When the self-payoff agent plays against itself, some runs are strongly cooperative (A1A2 = 4%; B1B2 = 92%) and in others there is no cooperation at all (A1A2 = 96%; B1B2 = 0%). However, when all of the runs were averaged together, the

Table 1. Individual Scores of Agent 1 (95% confidence intervals in parentheses)

Agent 1	Agent 2				
	Self-payoff	Opponent – perspective	Modeler	TFT	WSLS
Self-payoff	-56 (±7)	27 (±13)	-19 (±6)	-58 (±3)	250 (±7)
Opponent-perspective	-147 (±11)	-37 (±22)	-67 (±4)	-56 (±3)	335 (±11)
Modeler	-46 (±4)	-61(±4)	-1 (±5)	9 (±6)	103 (±1)

self-payoff agent demonstrated an overall tendency to play aggressively. Overall, the self-payoff agent defected on 73% of the trials. Given the design of the agent, it may seem peculiar that it has such a strong tendency to defect. Reinforcing the A1A2 chunk should make the cooperate move more appealing (because cooperating may yield +1 rather than -1). The answer to this puzzle may lie in the agent's prediction process. Each time the agent retrieves an outcome, that outcome is reinforced in declarative memory, even if it never occurs. In other words, the expectations of the agent are self-reinforcing. If the agent frequently retrieves the B1A2 chunk by chance, the B1B2 chunk may never have a chance to become sufficiently active. Moreover, cooperation is fragile because two conditions must be met for the model to select cooperate. The model must believe that defection will be punished (A1A2 > A1B2) and that cooperation will not be exploited (B1B2 > B1A2). If the opponent cooperates too frequently, then the agent will attempt to exploit it. If the opponent defects when the agent tries to cooperate, it will quickly retaliate.

Against TFT, the self-payoff agent earned a low negative individual score and combined score. The TFT agent swiftly and consistently punishes defection, but it will only cooperate again after its opponent has cooperated. This results in a loss for the self-payoff agent, making it less likely to cooperate in the future.

The self-payoff agent earned a very high score against WSLS, but it did so by exploiting WSLS's cooperation. The average score of the WSLS agent was very low. This is probably due to the self-payoff agent's strong tendency to defect. When the opponent defects, the WSLS agent loses points and therefore switches strategies. As a result, the WSLS agent essentially became a random decision agent because it lost points regardless of its move. This constant switching made the WSLS agent extremely vulnerable to exploitation.

Opponent-Perspective

The opponent-perspective agent performed the worst of the three agents. In terms of individual outcomes, it earned the lowest score against the self-payoff and modeler agents. It performed the best against the WSLS agent, but only because it tended to exploit the WSLS agent's cooperation.

The self-payoff agent heavily exploited the opponent-perspective agent, often defecting when the opponent-perspective agent cooperated (B1A2 = 12%). This happened mostly on runs in which the self-payoff agent very rarely cooperated. On these runs, the opponent-perspective agent's A1A2 chunk and A1B2 chunk were both highly active (because both outcomes are very frequent). The B1A2 and B1B2 chunks, on the other hand, only receive activation from the internal predictions of the model. This sometimes causes the B1B2 chunk to become highly active, leading the agent to predict cooperation. To make matters worse, when the opponent-perspective model chose to cooperate, it reinforced the A1B2 chunk of the self-payoff agent, reinforcing its tendency to defect. As a result, the rate of mutual cooperation was quite low.

Against the TFT agent, the performance of the opponent-perspective agent was very similar to the self-payoff agent. It showed a slight tendency to exploit the TFT agent (A1B2 = 5%). And, like the self-payoff agent, the rate of mutual cooperation was low. The reason why the opponent-perspective agent is not exploited by the TFT agent is because the TFT agent will always answer a cooperation with a cooperation. So when the TFT agent does unilaterally defect, the opponent-perspective agent has the opportunity to do the same next round.

Like the self-payoff agent, the opponent-perspective agent had a strong tendency to exploit the WSLS agent. The two most common outcomes were mutual defection (A1A2 = 38%) and unilateral defection by the opponent-perspective agent (A1B2 = 38%). The high activation of the A1B2 chunk caused the opponent-perspective agent to predict that the WSLS agent would never cooperate because it lost points so frequently as a result of doing so.

Modeler

The modeler agent, by contrast, did succeed in both achieving more favorable outcomes for itself and learning to cooperate with other agents when possible. The modeler obtained higher scores than both other agents against the self-payoff and the TFT agents. It also demonstrated a high positive score against the WSLS agent without exploiting it.

Table 2: Percentage of Joint Cooperation Trials (B1B2) (95% confidence intervals in parentheses)

Agent 1	Agent 2				
	Self-payoff	Opponent-perspective	Modeler	TFT	WSLS
Self-payoff	13 (±1)	10 (±1)	29 (±2)	11 (±2)	53 (±2)
Opponent-perspective	-	7 (±1)	13 (±2)	13 (±2)	19 (±2)
Modeler	-	-	45 (±2)	51 (±3)	97 (±.1)

The modeler agent does not achieve perfect prediction against the self-payoff agent. In fact, when the modeler plays against the self-payoff agent, the self-payoff agent achieves a higher score. However, the modeler is more successful at encouraging the self-payoff agent to cooperate. This improves the collective score and explains why the modeler scores more points when it plays against the self-payoff agent than the self-payoff agent does when it plays against itself. In addition, the modeler agent is able to predict the self-payoff agent well enough that it can avoid the heavy exploitation suffered by the opponent-perspective agent.

The modeler agent achieves the highest rates of cooperation of all three agents, demonstrating that the agent can quickly learn when cooperation with an opponent is possible. By a large margin, the modeler agent obtains the most mutually cooperative trials. There is room for improvement, however. The joint score of the modeler against the TFT agent is far from the ideal 200 points. This is because early defection from the TFT agent can lead the modeler agent to expect defection and respond in kind. This is not so with the WSLS agent, which changes to cooperation after a mutual defection.

Discussion

The fundamental problem for the player in the prisoner's dilemma is knowing when the opponent can be trusted. Our simulations suggest that metacognitive reasoning, if done appropriately, can help to solve this problem. Only one of the metacognitive agents we tested demonstrated an advantage over a non-metacognitive agent. The modeler agent was successful both in increasing its own individual gains and in discovering opportunities for cooperation with opponents. The opponent-perspective agent, however, was not able to achieve better outcomes for itself or find opportunities for cooperation.

Overall, the modeler agent is the best of the metacognitive agents because of its ability to flexibly adapt to different opponents. Against cooperative agents, it will quickly learn to cooperate and achieve positive scores. Against aggressive agents, it will learn to play defensively and defect most of the time. In addition, he modeler agent demonstrates one example of how metacognition may work to improve collective outcomes as well as individual outcomes. In interactions like those in the prisoner's dilemma, uncertainty can be a major obstacle to cooperation. Metacognitive reasoning may help to make other agents more predictable. When agents can be confident that their partners will cooperate, they may be more willing to cooperate themselves.

However, the modeler agent has several important limitations. One current limitation of the modeler agent is that it represents a very simple theory of mind because it does not represent the opponent's declarative knowledge or beliefs. These were not necessary for the present purposes because of the simplicity of the task, but modeler agents in more complex tasks will likely require such representations. An additional limitation is that it does not consider the relative payoffs of its choices. Rather, it imitates the opponent. This makes sense in the prisoner's dilemma, where unilateral choices (AB and BA) are generally avoided by agents because of severe costs. But it may not extend well to other tasks. This limitation could be addressed by making the model make three predictions: (1) the opponent's move on the current trial and (2) the opponent's response to each of the agent's possible moves. The model could then select the move that will lead to the highest immediate and future payoffs. The same declarative chunks used by the model to predict the opponent's current move could also be used to predict how the opponent will respond to the agent's current move.

The present simulations, together with those of Kennedy and Krueger (2013), highlight important challenges in modeling theory of mind. A "like me" agent (Meltzoff, 2007) may fail if the agent has access only to one strategy or a small number of strategies. This prevents the agent from considering that the opponent may be approaching the task in a different way. Kennedy and Krueger used a "like me" approach to develop a theory-of-mind agent that could play a voluntary trust game. This agent computed that it could achieve the highest average score by defecting. Believing that the opponent would reach the same conclusion, the agent always defected. Our opponent-perspective model shares a similar weakness. Because the short-term payoffs are skewed in favor of defection, both agents predict that their opponents will have a strong tendency to defect. In our simulations, this prevented the opponent-perspective agent from predicting the behavior of the more cooperative agents (TFT and WSLS). These results do not mean that taking the opponent's perspective is not helpful. But it may be necessary for agents to have access to a larger set of strategies so that they can find one that resembles the opponent's behavior. For example, if a model had two strategies (e.g. one cooperative and one aggressive), it could make predictions for the opponent's behavior based on both strategies. It could then select its own strategy based on which one was a better fit to the opponent's behavior.

A further challenge in constructing "like me" agents for 2-person games is tracking stochasticity in an opponent's behavior. Human performance in many of these games contains varying degrees of noise (Lebiere & West, 1999). Even if an agent has access to the same memory structure and decision rules as an opponent, that agent may still have difficulty tracking moment-to-moment variations. The opponent-perspective agent had a difficult time predicting the behavior of the self-payoff agent because it did not have access to its trial-by-trial predictions. On some runs, the self-payoff model's B1A2 chunk became active very early on by a series of chance retrievals, making it very unlikely to cooperate. However, the opponent perspective model did not know this, and still predicted that the self-payoff model would cooperate.

Opponent modeling, as opposed to the "like me," approach, is a more flexible strategy for a theory of mind

agent. Such agents are not limited by their own repertoire of strategies, and can successfully predict a wider range of opponents. In some cases, this approach may also be more cognitively efficient because it does not require mentally simulating the opponent's decision process. These agents may be especially powerful in situations in which opponents can change strategy without warning. If the agent has an adaptive declarative memory system, it could quickly update its mental model of the opponent and counteract the new strategy.

However, opponent modeling is not without drawbacks. It may be harder to implement such a strategy in more complex tasks. In the prisoner's dilemma, there are only two possible behaviors and an opponent's behavior is completely visible. When a greater number of behaviors is possible, it may be more difficult for an agent to determine which opponent behaviors are relevant.

These simulations do not address how well the model behavior replicates human behavior, nor does it show how the models would perform against humans. Playing against humans would provide a much stronger test for the modeler agent, as humans are likely to employ a variety of strategies and change strategies as the game progresses. We are currently planning an experiment in which we will collect this data. Of particular interest here is whether the modeler will be as successful in encouraging humans to cooperate as it is with the TFT and WSLS agents.

The work shown here demonstrates that metacognitive reasoning about an opponent can improve outcomes both for oneself and for one's opponent in the prisoner's dilemma. By learning an opponent's strategy, an agent can determine if it is safe to cooperate, or if it is better to defect. This increases the probability that the agent and its partner will discover a stable, mutually beneficial outcome. Metacognition also helps an agent detect and defend itself against cheaters. However, players should beware to avoid assuming that all opponents will play the game the same way they do. We expect that these benefits extend not only to other simple games but also to more complex scenarios. It remains for further work to discover how metacognitive reasoning can be best employed to achieve success in these other tasks.

Metacognitive reasoning about an opponent's behavior can provide an advantage in the repeated prisoner's dilemma. The ability to predict an opponent's next move helps to determine when it is safe to cooperate and when one should defect. This suggests that performance even in simple games may benefit from developing a theory of mind about one's opponent.

Acknowledgments

This work was funded by European Union Grant #611073: Multiperspective Multimodal Dialogue (METALOGUE).

References

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological Review*, *111*(4), 1036.

Axelrod, R. (1980). More Effective Choice in the Prisoner's Dilemma. *Journal of Conflict Resolution*, *24*(3), 379–403.

de Weerd, H., Verbrugge, R., & Verheij, B. (2013). Higher-order theory of mind in negotiations under incomplete information. In *PRIMA 2013: Principles and Practice of Multi-Agent Systems* (pp 101-16) Springer: Berlin Heidelberg.

Gallese, V., & Goldman, A. (1998). Mirror neurons and the simulation theory of mind-reading. *Trends in Cognitive Sciences*, *2*(12), 493–501.

Kennedy, W. G., & Krueger, F. (2013). Building a Cognitive Model of Social Trust Within ACT-R. In *AAAI Spring Symposium: Trust and Autonomous Systems* (pp. 29–34).

Hingston, P., Dyer, D., Barone, L., French, T., & Kendall, G. (2007). Opponent Modelling, Evolution, and the Iterated Prisoner's Dilemma. In *The Iterated Prisoner's Dilemma: Celebrating the 20th Anniversary* (pp. 139–170). World Scientific.

Lebiere, C., Wallach, D., & West, R. (2000). A memory-based account of the prisoner's dilemma and other 2x2 games. *Proceedings of International Conference on Cognitive Modeling*. 185-93.

Lebiere, C., & West, R. L. (1999). A dynamic ACT-R model of simple games. In *Proceedings of the Twenty-first Conference of the Cognitive Science Society*, pp. 296-301. Mahwah, NJ: Erlbaum.

Logan, G.D. (1988). Toward an instance theory of automatization. *Psychological Review, 95*, 492-528. Logan,

McCabe, K. a, Smith, V. L., & LePore, M. (2000). Intentionality detection and "mindreading": why does game form matter? *Proceedings of the National Academy of Sciences of the United States of America*, *97*(8), 4404–9.

Meltzoff A. N. 2007. Imitation and Other Minds: The "Like Me" Hypothesis. In: S. Hurley and N. Chater (eds) Perspectives on Imitation: From Neuroscience to Social Science, pp 55-77. Cambridge: MIT Press.

Nowak, M., & Sigmund, K. (1993). A strategy of win-stay, lose-shift that outperforms tit-for-tat in the prisoner's dilemma game. *Nature*, *364*, 56–58.

Premack, D., & Woodruff, G. (1978). Does the chimpanzee have a theory of mind?. *Behavioral and brain sciences*, *1*(04), 515-526.

Wedekind, C., & Milinski, M. (1996). Human cooperation in the simultaneous and the alternating Prisoner's Dilemma: Pavlov versus Generous Tit-for-Tat. *Proceedings of the National Academy of Sciences of the United States of America*, *93*(7), 2686–9.

Exploration-Exploitation in a Contextual Multi-Armed Bandit Task

Eric Schulz[1] **(eric.schulz.13@ucl.ac.uk)**, **Emmanouil Konstantinidis**[2] **(em.konstantinidis@gmail.com)**, & **Maarten Speekenbrink**[1] **(m.speekenbrink@ucl.ac.uk)**
[1]Department of Experimental Psychology, University College London, London, WC1H 0AP
[2]Department of Social and Decision Sciences, Carnegie Mellon University, Pittsburgh, PA 15213

Abstract

We introduce the Contextual Multi-Armed Bandit task as a method to assess decision making in uncertain environments and test how participants behave in this task. Within an experimental paradigm named Mining in Space, participants see 4 different planets that are described by 3 different binary elements (the context) and then have to decide on which planet they want to mine (which arm to play). We find that participants adapt their decisions to the context well and can best be described by a Contextual Gaussian Process algorithm that probability matches according to expected outcomes. We conclude that humans are well-adapted to contextualized bandit problems even in potentially non-stationary environments through probability matching, a heuristic that used to be described as biased behavior. We argue that Contextual Bandit problems can provide further insight into how people make decisions in real world scenarios.
Keywords: Decision Making, Active Learning, Exploration-Exploitation, Contextual Multi-Armed Bandits

Introduction

A Contextual Multi-Armed Bandit (CMAB) task is a task in which an agent is confronted with mutliple options ("arms" of a bandit) out of which one can be chosen. The context describes the currently available information that can be utilized to choose the best arm to play (Li et al., 2010). This scenario is a good model for many real world problems; from choosing what to eat, to buying clothes in a shop, all the way to finding the right person to befriend; many situations require us to make the right choice in a given context without the chance to actually observe the outcome of unchosen options, constantly trading-off between *exploration* (trying out new things) and *exploitation* (maximizing expected reward). Therefore, contextual bandit tasks might help to shed light on how we make contextual decisions in general and on how we integrate information into our decisions in particular.

Despite a vast amount of research on multi-armed bandit tasks (Steyvers et al., 2009), little is known about participants' behavior in experiments involving contextual bandits. This is remarkable given that contextual bandits provide us with a scenario in which, instead of treating learning and decision making distinctively, participants have to learn a function that maps a context to outcomes and then act according to their predictions of these. In that sense, contextual bandit tasks could be seen as a quintessential scenario of everyday decision making.

In what follows, we will introduce the contextual multi-armed bandit task (`CMAB`) and probe how participants perform in one simple version thereof. The experimental task can be approached as both a contextual bandit as well as a so-called restless bandit (in which the average rewards associated with the arms vary over time) by ignoring information, but is designed such that only taking the context into account will lead to above chance performance. We will show that humans are able to learn well within the `CMAB` and are best described by sensitive exploration-exploitation behavior based on probability matching decisions to the estimated outcomes of non-parametric Bayesian models (Srinivas et al., 2009). These models do not try and learn one particular parametric structure, but rather a distribution over different generating mechanisms in a given environment. Moreover, probability matching (also called *Thompson sampling*) offers a simple yet powerful way to balance exploration and exploitation in decisions, especially in non-stationary environments. The main contributions of this paper are threefold:

1. We introduce the `CMAB` as an experimental paradigm and emphasize its importance for psychological research.
2. We model human context learning as non-parametric: instead of relying on an arbitrary set of parametric candidate models, participants seem to learn in a way that represents distributions over generating mechanisms.
3. We show that participants apply a behavior best-described by Thompson sampling/Probability Matching. This behavior has often been referred to as biased and erroneous fallacy. However, it turns out to be a satisfyingly sensible strategy in dynamic environments (see Agrawal & Goyal 2012, for further details).

Definitions and Models

Contextual Bandit Problems

Consider a game in which, in each round $t = 1, \ldots, T$, an agent observes a context $s_t \in S$ from a set of S contexts and has to choose an action $a_t \in A$ from a set of possible actions A. The agent then receives a payoff $y_t = f(s_t, a_t) + \varepsilon_t$. It is the agent's task to take those actions that produce the highest payoff. As the expected payoff depends on the context, the agent has to learn the underlying function f; sometimes, this may require the agent to choose an action which is not expected to give the highest payoff, but which might provide more information about f, thus choosing to explore rather than exploit. As the different actions are normally described as playing a bandit's arm and the context provides information that might help to find the right arm to play, these games are called *Contextual Multi-Armed Bandit* tasks.

Different models can be used to learn in a contextual bandit setting. The models applied here broadly fall within two categories: *context-blind* and *contextual* models. Context blind

models ignore the provided context completely and only learn based on direct feedback of the chosen arms. Contextual models do take the context into account and therefore are generally expected to perform better than context-blind models.

We will first describe a general choice rule, then the context blind models, and afterwards the parameterization of the two used contextualized models (linear and Gaussian Process regression) before then describing two different decision rules that can be used for the contextual models.

Choice rule

In the psychological task considered later, the context s_t at time t will be the same for all arms, while the function that maps the context to the (expected) payoff of the arm will vary over arms. The task is therefore to learn functions f_k for each arm that map the context to the payoff and then choose the arm with the highest expected payoff while constantly trading-off between exploration and exploitation. To do so, the models proposed here produce n different values $\theta_{1,t}, \theta_{2,t}, \ldots, \theta_{n,t}$ to compare between the n different arms at a time point t given the current context s_t by some learned function f_k that matches the context s_t to the considered arm k:

$$\theta_{k,t} = f_k(s_t) \tag{1}$$

This could be the mean predicted outcome for every arm or any other value as described below. In order to transform these values to a probability of picking a given arm arm_j, the values are transformed by a softmax rule with inverse temperature parameter γ as in Equation 2.

$$p(\text{arm}_t = k) = \frac{\exp\{\gamma\, \theta_{k,t}\}}{\sum_{i=1}^{n} \exp\{\gamma\, \theta_{i,t}\}} \tag{2}$$

Context-blind Models

Context-blind models ignore the context completely and only respond to the observed outcomes of arms over time.

Random choice The most simplistic context-blind model is a random choice. This model picks every arm with equal probability $p(\text{arm}_t = k) = 1/\#\text{arms}$. As this model does not learn over time, it will provide a baseline against which all the other models can be compared.

μ-tracking The other context-blind model is based on simple mean tracking.

$$\theta_{k,t} = \hat{\mu}_{k,t} = \frac{1}{n} \sum_{\tau=1}^{t} \delta_{\text{arm}_\tau = k} y_\tau \tag{3}$$

where $\delta_{\text{arm}_t = k} = 1$ if arm k is chosen at time t and 0 otherwise.

Contextual Models

The contextual models learn the functions f_k that map the context to the (expected) payoff for each arm. Here, we will consider two contextual models: linear and Gaussian Process regression.

Linear Regression Linear regression is a simple approach to learn each function f_k that relates the contexts s_t to an output $f_k(s_t)$. Each context s_t has values on a total of m attributes, i.e., $s_t = (s_{1,t}, \ldots, s_{m,t})$. The regression model learns a linear function of the context attributes:

$$\hat{f}_k(s_t) = \beta_0 + \sum_{i=1}^{m} \beta_i s_{i,t} + \varepsilon_t \tag{4}$$

Let $s_{1:t} = (s_1, \ldots, s_t)$ denote all the contexts encountered at time t. The regression model is estimated from $s_{1:t}$ and then used to predict new outcomes for each arm given a new contexts at $t+1$. Once the new output has been chosen, the regression model is updated and then used for the next trial with new contexts. As this is a parametric model, it assumes that participants approach the problem in a way that only allows for linear effects of the context. In order for the regression approach to not suffer from matrix deficiencies, 10 pseudo-observations were created from a Normal distribution with $\mathcal{N}(50, 10)$.

Gaussian Process Regression Another class of models is non-parametric. Instead of postulating one concrete parametric form (e.g., a linear one) out of an infinite set of possible forms (a choice that, without any further knowledge, is arbitrary), non-parametric models implicitly assume that the function can be represented by an infinite number of parameters and let the data speak directly by the means of Bayesian inference. One example of a non-parametric model in the functional domain is a Gaussian Process.

A Gaussian Process (henceforth $\mathcal{GP}$) is a collection of random variables from which every finite marginal distribution is multivariate Gaussian. We define a mean function $m(x)$ and the covariance function $k(x,x')$ of a process $f(x)$ as

$$m(x) = \mathbb{E}[f(x)] \tag{5}$$

$$k(x,x') = \mathbb{E}[(f(x) - m(x))(f(x') - m(x'))] \tag{6}$$

A Gaussian process then can be expressed as

$$f(x) \sim \mathcal{GP}\big(m(x), k(x,x')\big). \tag{7}$$

Even though many different covariance functions exist, within all the examples and calculations presented here the *squared exponential* covariance function with a length scale λ will be used.

$$\text{cov}\big(f(x_p), f(x_q)\big) = k(x_p, x_q) = \exp\left(-\frac{|x_p - x_q|^2}{2\lambda}\right) \tag{8}$$

The lengthscale λ was estimated by using gradient descent.

In the noisy situation that will be analyzed in all of the upcoming situations, the covariance can be written as follows

$$\text{cov} = (y_p, y_q) = k(x_p, x_q) + \sigma_n^2 \delta_{pq}, \tag{9}$$

where δ is Kronecker's δ, which is 1 if $p = q$ and 0 otherwise.

Suppose we have collected observations $\mathbf{y}_t = [y_1, y_2, \ldots, y_t]^\top$ at inputs $\mathbf{x}_t = \{x_1, \ldots, x_t\}$, $y_t = f(x_t) + \varepsilon_t$,

$\varepsilon_t \sim \mathcal{N}(0,\sigma^2)$, then the posterior over f is a $\mathcal{GP}$ with mean $m_T(x)$, covariance $k_T(x,x')$, and variance $\sigma_T^2(x)$:

$$m_T(x) = \mathbf{k}_T(x)^\top (K_T + \sigma^2 \mathbf{I}) y_T \tag{10}$$

$$k_T(x,x') = k(x,x') - \mathbf{k}_T(x)^\top (\mathbf{K}_T + \sigma^2 \mathbf{I})^{-1} \mathbf{k}_T(\mathbf{x}') \tag{11}$$

$$\sigma_T^2(x) = k_T(x,x) \tag{12}$$

where $\mathbf{k}_T(x) = [k(x_1,x),\ldots,k(x_T,x)]^\top$ and $\mathbf{K}_T$ is the positive definite kernel matrix $[k(x,x')]_{x,x' \in A_T}$. The $\mathcal{GP}$ is used in the same fashion as the linear model by always using all the contexts and observed outcomes up to time point t in order to make predictions for time point $t+1$. Therefore, a $\mathcal{GP}$ for every arm over time will be estimated given the observed context as shown in Equation 13.

$$\hat{f}_j(s) \sim \mathcal{GP}\big(m(s), k(s,s')\big). \tag{13}$$

As a Gaussian Process is a non-parametric model for function learning, its application represents the assumption that participants do not a priori expect one parametric form of a function, but rather learn the form by the observed data over time. The Gaussian Process was initialized by the use of 10 pseudo-observations as in the regression approach described before.

Sampling strategies

Let us now look at the different algorithms that can be used to apply the two contextual models in a `CMAB`. Sampling strategies here mean different ways by which one could come up with a choice of an arm, given the estimated expected outcomes at a given time.

Upper Confidence Bounds The upper confidence bound algorithm estimates a trade-off between the current expected value and the variance per arm and optimistically picks the arm with the highest upper confidence bound. This algorithm has been shown to perform well in many real world contextual bandit tasks (Krause & Ong, 2011). The way a UCB-sampling agent would select an arm is described in Algorithm 1.

Algorithm 1 Upper Confidence Bands Sampling

Require: Context $\mathbf{s}$; Models $\mathcal{M}_{j,t-1}$
  for $t = 1,2,\ldots,T$ **do**
    Choose $\text{arm}^*_{jt} = \text{argmax}\ \mu(\mathcal{M}_{j,t-1}(\mathbf{s})) + 1.96\sigma(\mathcal{M}_{j,t-1}(\mathbf{s}))$
    Sample $y_t = f(\text{arm}^*_{jt}) + \varepsilon_t$
    Update $\mathcal{M}_{j,t-1} \rightarrow \mathcal{M}_{j,t}$
  end for

The trade-off is based on a confidence interval approximation based on a normal distribution and therefore the trade-off parameter is set to 1.96, marking the 95% confidence interval. The UCB-algorithm can be seen as a selection strategy with an exploration bonus, where the bonus depends on the confidence interval of the estimated mean return. As we will need probability estimates to model participants choices later on, the estimates for arm^*_{jt} were fed into the softmax equation described above.

Thompson Sampling Thompson sampling chooses each arm according to the (subjective) probability that it provides the highest payoff out of all the available arms, given the context (May et al., 2012). This is a form of probability matching. The algorithm can be implemented by sampling for each arm a payoff according to the learned models of the arms, and then choose the arm with the highest sampled payoff. Even though this model seems very simplistic, it can perform reasonably well in contextual bandit tasks and can describe human choices in (non-contextual) restless bandit tasks well (Speekenbrink & Konstantinidis, 2014). Whereas psychology has looked at probability matching as an inferior strategy of decision making for a long time, it has been shown to perform well in many restless bandit tasks and can easily adapt to changing environments as it still keeps on exploring other options over time.
An agent following the Thomson sampling algorithm would pick the next arm as described in Algorithm 2.

Algorithm 2 Thompson Sampling

Require: Contexts $s_{1:T}$; Models $\mathcal{M}_j$
  for $t = 1,2,\ldots,T$ **do**
    for $\text{arm}_{k,t}, k = 1,\ldots,n$ **do**
      Sample $y^*_{k,t-1} \sim \mathcal{M}_{k,t-1}(s_t)$
    end for
    Choose $\text{arm}_t = \arg\max_k y^*_{k,t}$
    Sample $y_t = f(\text{arm}_t) + \varepsilon_t$
    Update $\mathcal{M}_{j,t-1} \rightarrow \mathcal{M}_{j,t}$
  end for

Main advantages of Thompson sampling are (1) that it does not rely on additional parameter tuning, and (2) that it can adapt to many diverse environments. The probability of an arm to be chosen was calculated as shown in Equation 14.

$$p(\text{arm}_t = k) = p(\forall j \neq k : y^*_{k,t} \geq y^*_{j,t}) \tag{14}$$

This means that each arm is predicted to be chosen by its probability to produce the highest outcome at a given time.

Summary of all models

Taking all of the models (context-blind and contextual) and choice rules together results in the models shown in Table 1.

Class	Algorithm	Description
Context-blind	Random	Picks at random
	μ-tracking	Picks tracked mean
Linear	UCB	Picks upper confidence band
	Thompson	Probability matching
Gaussian Process	UCB	Picks upper confidence band
	Thompson	Probability matching

Table 1: Summary of all used models

Experiment : Contextual Bandit Task

The experiment was designed to test if participants are able to learn in a contextual bandit task. It used a relatively simple description of the context s and the different arms. Within this first CMAB experiment we focused on a task with three binary context variables that could either be on (+) or off (-) and 4 different arms.

Contextual Bandit setting

The outcomes of the different arms in dependency of the context are shown in Equations 14-17.

$$y_{1,t} = 50 + 15 \times s_{1,t} - 15 \times s_{2,t} + \varepsilon_{1,t} \quad (15)$$
$$y_{2,t} = 50 + 15 \times s_{2,t} - 15 \times s_{3,t} + \varepsilon_{2,t} \quad (16)$$
$$y_{3,t} = 50 + 15 \times s_{3,t} - 15 \times s_{1,t} + \varepsilon_{3,t} \quad (17)$$
$$y_{4,t} = 50 + \varepsilon_{4,t}, \quad (18)$$

with $\varepsilon_{k,t} \sim \mathcal{N}(0,5)$. This means that each arm reacted differently to the context $s_t = (s_{1,t}, s_{2,t}, s_{3,t})$ through linear functions, producing an outcome $f_k(s_t) + \varepsilon_{k,t}$ as described before.

For all different contexts, the probability of being + was set to $p(s_{j,t} = +) = 0.5$. The different arms were deliberately set up such that all the expected values are the same, $\mathbb{E}[y_{k,t}] = 50$ over time in order to avoid first order stochastic dominance of context-blind choices[1]. This means that the only way to gain higher values than the individual bandits' averages is by learning how the different factors influence the arms within every trial. The context-blind strategies therefore would not perform better than chance. Moreover, introducing an arm that only returns the overall mean with some added noise (Arm 4) helps us to distinguish even further between contextual and context-blind models. As context blind models only take the outcome into account, they should prefer Arm 4 as it produces the same mean over time, but exhibits less variance and therefore second order dominates all the other arms. Contextual models on the other hand should (at the end) almost never select Arm 4 as taking the context into account will generally lead to better outcomes than the simple mean alone.

Methods

Participants 47 participants (26 males, age: $M = 31.9$, $SD = 8.2$) were recruited via Amazon Mechanical Turk and received \$0.3 plus a performance-dependent bonus of up to \$0.5 as a reward. None of the participants were excluded from the remaining analysis.

Design Participants were told that they had to mine for "Emeralds" on different planets. Moreover, it was explained that at each time of mining the galaxy was described by 3 different environmental factors, "Mercury", "Krypton", and "Nobelium", that could either be on (+) or off (-) and had different effects on different planets. Participants were told that they had to maximize the overall production of Emeralds over time by learning how the different elements influence the planets and then picking the planet they thought would produce the highest outcome, given the currently available elements. It was explicitly noted that different planets can react differently to different elements. The total number of trials was fixed to be 150 and the experiment was well-received on Mechanical Turk.[2]

Notice that this task exactly corresponds to the contextualized multi-armed bandit problem described above, where different planets represent different arms and different elements represent the context. This means that a good strategy would involve a trade-off between learning the 4 different functions describing how the elements influence each planet and then maximizing the expected outcome by choosing the right planet (arm) at a given time and context. A screenshot can be seen in Figure 1.

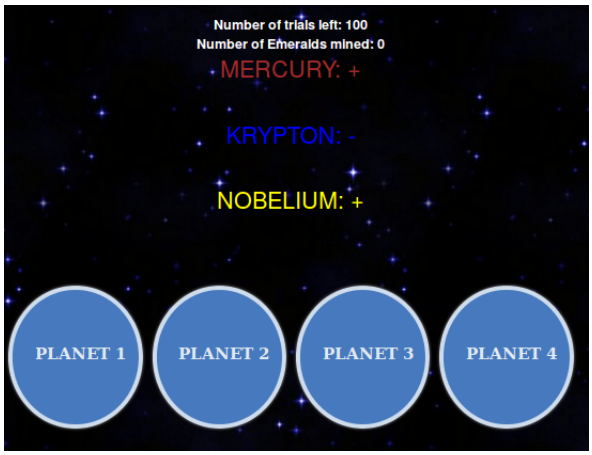


Figure 1: Screenshot of the Experiment

Which planet corresponded to which of the pay-off functions described above was assigned randomly before the start of the experiment.

Analysis All models were fitted by maximum likelihood. We assessed the ability for each of the 6 models to predict participants' choices over all trials and calculated Akaike's "An Information Criterion" (AIC) by finding the best inverse temperature parameter γ through a combination of golden section search and successive parabolic interpolation provided by the R-function `optimize` for all continuous outcomes (the UCB and the μ-tracker) or by using the estimated probabilities directly (for Thompson sampling). The AIC here is based on the log-likelihood of the predicted probabilities for eah chosen arm over all trials.

Hypotheses

Based on our conjectures above, we hypothesized the following 3 findings *a priori*:

[1] Situations only containing - or + were not used

[2] Search for Eric Schulz on Turkopticon

1. Participants will be able to learn how the context depends on the outcomes and therefore will be generally better described by contextual than by context-blind models.
2. Instead of one particular parametric strategy, participants will approach the problem in a non-parametric way allowing them to potentially learn different types of functions, if need arose. Therefore, participants will be better described by the Gaussian Process than by the linear model.
3. Instead of maximizing output by a deliberate mean-variance trade-off, participants approach dynamic decision making problems by utilizing a probability matching heuristic. Thus, they will be better described by the Thompson sampling choice rule than by the Upper Confidence Band approach.

Results

Figure 2 shows the raw data for each participant over all 150 trials.

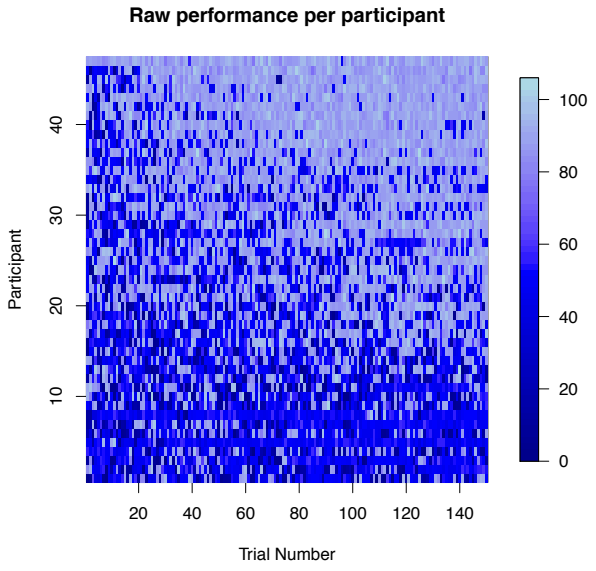


Figure 2: Obtained payoff for each participant at each trial.

In Figure 2, participants are ordered in ascending order according to their mean overall performance. It can be seen that almost all participants received higher payoffs towards the end. Moreover, some participants (the top half) seem to learn the functions very well and then consistently produced high scores over time. In the lower half, however, there are a few participants who do not seem to learn the functions too well.

Most participants also performed better than chance (an average score of higher than 50) as is displayed in the histogram of average rewards per participant shown in Figure 3.
Indeed, performing a simple t-test against $\mu = 50$ confirmed that most participants performed above chance with $t(46) = 7.17$, $p < 0.01$.

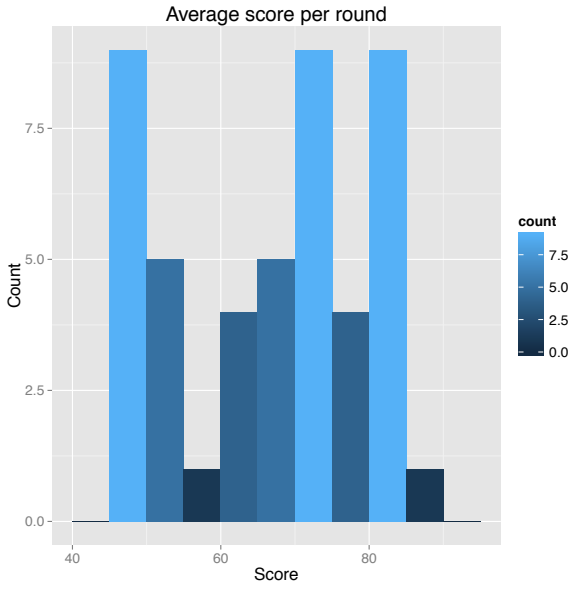


Figure 3: Average payoff per round. More participants per count are marked by a lighter blue.

Even though some participants performed below chance, we did not exclude any of them from the analysis described next as we did not want to bias our results in favor of the contextual models. The overall performance of all models is shown in Table 2.

Table 2: Average AIC, standard deviations, and the number of participants best fit by the different models.

Model	AIC_{mean}	AIC_{SD}	#best
Random	415.9	0	5
μ-tracking	412.9	5	6
Linear-UCB	387.8	34	4
Linear-Thompson	383.0	46	15
GP-UCB	389.4	34	3
GP-Thompson	381.6*	42	18*

The 5 participants that were best described by the Random model were also among the participants who performed at chance level as shown in Figure 3.
It can clearly be seen that the contextual models described participants behavior better than the two context-blind models. Taken together, only 7 participants were best described by the context-blind models, whereas 40 participants were best described by the contextual models.

The Gaussian Process models described more participants best than the linear regression models (21 vs. 19). Even though this is only a small difference, it is evermore surprising as the linear model here would be the best description of the underlying system a priori – the task is a linear system

after all. What this tells us is that instead of approaching the problem with a fixed parametric representation in mind, participants might indeed apply a learning strategy that is more easily adaptable to other scenarios than a linear one. Lastly, more people were described best by the probability matching algorithm of the Thompson sampler than by the expectation-variance-trade-off calculation of the UCB (33 vs. 7). This indicates that participants seem to apply this heuristic. Probability matching has been described as rather dumb in the past. However, in situations where the goal is to trade-off between exploration and exploitation, this heuristic is actually a smart strategy as it keeps exploring while at the same time generating high outcomes (Agrawal & Goyal, 2012).

That participants actually do learn over time while also sticking to some exploratory behavior can be see in Figure 4.

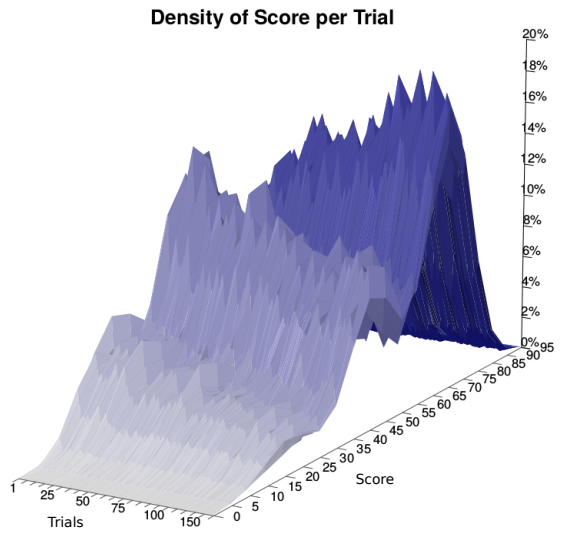


Figure 4: Density of outcome over participants per round.

As participants learn over time, the density for higher scores goes up and the density for lower scores goes down.

Discussion and Conclusion

We have introduced the Contextual Multi-Armed Bandit task as a new paradigm to assess participants' decision making in uncertain environments. Within this task, participants were able to learn the underlying structure well and took the provided context into account. Overall, most participants performed above chance and were best described by a GP-based Thompson sampling algorithm. That participants were best described by a Gaussian Process seems to suggest that – instead of having one specific parametrized representation of the environment– people learn by the means of general effective strategies that can potentially adapt to new or changing environments if required. However, future studies will have to replicate this findings in other domains. The good performance of the Thompson sampler fits well into past findings as Speekenbrink & Konstantinidis (2014) found that Thompson sampling predicts participants' choices well in a restless bandit task. Moreover, this means that probability matching, a behavior that used to be frowned upon as irrational, provides a sensitive strategy that people might actually apply in exploration-exploitation scenarios. In conclusion, all of our three main hypotheses were confirmed. This research can only be seen as a first step into research on contextual bandit problems. Future studies could try to assess how people behave in scenarios where more context is given either by creating a multi-context environment (for example, one context per planet) or by providing continuous context variables (for example, values between 0 and 10). Another option could be to assess how participants learn in a multi-context-multi-function environment, that is an environment where the different contexts relate to arms in different ways. As we have found that Thompson sampling can provide a good description of participants' behavior and Thompson sampling is known to be well-adapted towards dynamically changing environments, a future experiments could try to model participants' behavior in dynamic tasks, where the reward structure changes over time or with the number of times a given option has been chosen.

Here, we have introduced a comparison between a linear model and Gaussian process in what can essentially be described as an active learning task. However, in future experiments we aim to try and compare even more elaborate models within this context. Using an active learning domain as a platform for model comparison might be another useful approach to decide among models from a list of seemingly endless contestants (Schulz et al., 2014).

Acknowledgements

ES is supported by the UK Centre for Doctoral Training in Financial Computing & Analytics. Data sets and code are available at https://github.com/ericschulz/contextualbandits.

References

Agrawal, S., & Goyal, N. (2012). Thompson sampling for contextual bandits with linear payoffs. *arXiv preprint arXiv:1209.3352*.

Krause, A., & Ong, C. S. (2011). Contextual gaussian process bandit optimization. In *Advances in Neural Information Processing Systems*, (pp. 2447–2455).

Li, L., Chu, W., Langford, J., & Schapire, R. E. (2010). A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, (pp. 661–670). ACM.

May, B. C., Korda, N., Lee, A., & Leslie, D. S. (2012). Optimistic bayesian sampling in contextual-bandit problems. *The Journal of Machine Learning Research*, *13*(1), 2069–2106.

Schulz, E., Speekenbrink, M., & Shanks, D. R. (2014). Predict choice: A comparison of 21 mathematical models. Cognitive Science Society.

Speekenbrink, M., & Konstantinidis, E. (2014). Uncertainty and exploration in a restless bandit task.

Srinivas, N., Krause, A., Kakade, S. M., & Seeger, M. (2009). Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*.

Steyvers, M., Lee, M. D., & Wagenmakers, E.-J. (2009). A bayesian analysis of human decision-making on bandit problems. *Journal of Mathematical Psychology*, *53*(3), 168–179.

Predicting Surprise Judgments from Explanation Graphs

Meadhbh I. Foster (meadhbh.foster@ucdconnect.ie)
Mark T. Keane (mark.keane@ucd.ie)

Department of Computer Science & Informatics, University College Dublin
Belfield, Dublin 4, Ireland

Abstract

Surprise is a ubiquitous phenomenon that is implicated in many areas of cognition, from learning, to decision making, to creativity. For example, it has recently been proposed as a trigger for learning in robotic agent architectures. This paper describes a novel cognitive model of surprise based on the idea that surprise is fundamentally about explaining why the surprising event occurred; events that can be explained easily are less surprising than those that are more difficult to explain. Using explanations that people have produced, this surprise model builds a directed graph of explanations that link the setting and outcome of a given scenario, and uses this graph to predict surprise ratings. Simulations are reported which show that the model's performance corresponds closely to the psychological evidence, as measured by people's ratings of different surprising scenarios.

Keywords: surprise; explanation; cognitive; judgments

1. Introduction

The phenomenon of surprise has been intensively researched since Darwin's time, perhaps because it involves an interesting mixture of emotion and cognition. Though surprise clearly involves an emotional reaction (often accompanied by a startle response), it also seems to serve a strategic, cognitive goal, as it directs attention to explain why the surprising event occurred and to learn for the future (e.g., Ranganath & Rainer, 2003). Originally conceived of as a "basic emotion" (e.g., Darwin, 1872; Ekman & Friesen, 1971; Izard, 1977; Tomkins, 1962), more recently surprise has been re-appraised as a cognitive state because, unlike most emotions, it can be either positively or negatively valenced (Ortony & Turner, 1990).

In Artificial Intelligence, Macedo, Cardoso, Reisenzein, Lorini, and Castelfranchi (2009) have argued that any agent operating in a changing and imperfectly known environment needs a surprise mechanism to survive. Specifically, surprise is considered an essential requirement in robotic, agent architectures to identify learning events (e.g., Macedo, Reisenzein, & Cardoso, 2004).

In Cognitive Psychology, theories of surprise fall into two identifiable camps, the "expectation" and "sense-making" approaches. Expectation theories focus on the properties of surprising outcomes, characterizing them as low-probability events, disconfirmed expectations, schema-discrepant events or events of contrasting probabilities (e.g., Meyer, Reisenzein, & Schützwohl, 1997; Reisenzein & Studtmann, 2007; Teigen & Keren, 2002, 2003). Sense-making theories stress the importance of understanding and integrating the surprising event, a task often carried out retrospectively rather than predictively (e.g., Kahneman & Miller, 1986; Maguire, Maguire, & Keane, 2011). While this theoretical opposition is real, they may actually be complementary, addressing different classes of events (see section 1.1).

The current paper focuses on sense-making aspects of surprise, where the sense-making process is cast as explanation formation; people's perception of surprise is a metacognitive estimate of the cognitive work involved in explaining a surprising event (see Foster & Keane, 2013). Stated simply, some surprises are more surprising because they are harder to explain. Though both are surprising events, it is more surprising to hear that an X-Factor teen contestant has died, than it is, unfortunately, to hear that Amy Winehouse has died (given her pre-history of substance abuse), because the former is harder to explain than the latter (although, as surprise is a subjective experience, this can depend on the individuals level of knowledge surrounding each event). Traditionally, explanation is seen as playing a role in building causal models or predictive schemas to deal with future events (Heider, 1958; Lombrozo & Carey, 2006). However, apart from having a predictive role when a new situation is initially encountered, explanation may also serve to help people decide how information should be weighted or how attention should be allocated, as events occur (Keil, 2006). In the remainder of this section, before presenting our model of surprise, we briefly consider broad categories of surprising events, and previous models proposed from probabilistic and sense-making perspectives.

1.1 Categories of Surprising Events

The theoretical division between expectation and sense-making accounts of surprise may reflect a different emphasis on two broad classes of scenario. Some scenarios invite the formation of definite expectations, whereas others do not. For example, if a coin is tossed in the air, it is reasonable to develop the expectation that it will come down as either heads or tails. If someone is running a race, it is reasonable to develop expectations that they will win (or lose). However, if you are sitting at home watching TV, you are unlikely to develop the expectation that a rock will come through the window. If you are watching a teenage X-Factor contestant singing on TV, it is not reasonable to develop the expectation that they will die an hour from now. Indeed, many everyday scenarios are probably ones in which people *do not* generate expectations for every possible outcome of the scenario before-the-fact. As Ortony and Partridge have

noted: "It is not realistic to suppose that a system with goals and tasks to perform is at the same time randomly spawning inferences about unrelated and improbable possibilities" (p.107, 1987).

As we shall see in the following sections, this distinction between two broad classes of surprising event – those that are prior-expectation-inviting and those that are not prior-expectation-inviting – is important, as the latter present specific issues for probabilistic accounts that do not arise for sense-making interpretations of surprise.

1.2 A Probabilistic Model of Surprise

Most of the existing computational models of surprise are framed from the expectation-disconfirmation perspective rather than the sense-making one (see, e.g., Bae & Young, 2008; Lorini & Castelfranchi, 2006, 2007; Macedo & Cardoso, 2001; Macedo et al., 2004; Baldi & Itti, 2010). For example, Baldi and Itti's (2010) Bayesian theory of surprise, mathematically defines surprise as the effect that an event has on an observer; specifically, surprise is defined as the distance between prior and posterior belief distributions (see also Itti & Baldi, 2006, 2009). They have shown this theory of surprise to work well in predicting human gaze by computing surprise over images and video stimuli in a computer vision system using a neural network architecture. However, Itti and Baldi (2009) note that a consistent definition of surprise (using a Bayesian framework), must involve prior and posterior distributions to capture subjective expectations. So, for this theory prior beliefs (i.e., expectations) necessarily need to be computed so that the change between prior and posterior belief distributions can be calculated. As such, the theory cannot account for instances of surprise in which expectations are not computed in advance.

Many other probabilistic models recognize that this "missing-expectations problem" needs to be addressed. For instance, Bae and Young (2008) employ the notion of *postdictability* in their model of surprise in narratives. That is, often in a narrative there is a "hidden truth" revealed at the end of a story that resolves some surprising event, where this resolution essentially involves an after-the-fact explanation step. Similarly, Macedo and Cardoso (2001; Macedo et al., 2004) draw on both the cognitive-psychoevolutionary model of surprise and Ortony and Partridge's (1987) distinction between active and passive expectations to model surprise. Passive expectations are those that cause the agent to be surprised after-the-fact; cognitive attempts to retrospectively construct what they could have expected to happen. However, they still rely on expectations, and, as such, are distinct from more sense-making, explanation-based proposals for surprise.

At present, it is not wholly clear how the missing-expectation problem might be best handled within the probabilistic framework. What can be said is that, at the very least, it requires some "retrospective machinery" to recover what expectations should have been adopted, after-the-fact. As we shall see, these issues do not arise for sense-making accounts, as they fundamentally operate in a retrospective way; they typically see surprise as a process of resolving some inconsistency after the event occurs.

1.3 A Sense-Making Model of Surprise

We know of only one fully-implemented computational model adopting the sense-making perspective; that developed by Maguire, Costello and Keane (2006), based on Grimes-Maguire and Keane's (2005) theory of Representation-Fit (see also Maguire et al., 2011). It conceptualizes surprise as a representation-fitting process of integrating surprising events with existing schemas, as opposed to a process of expectation disconfirmation. Their model consists of two parts: an integration stage and an analysis stage, utilizing many ideas from Connell and Keane's (2006) Plausibility Analysis Model (PAM). The integration stage links each event in a scenario with those that have happened already so that a current, coherent representation is formed, and a total incoherency score for the scenario is created, based on the ratio of linked concepts to unlinked concepts. Then, the analysis stage involves a systematic assessment of this representation; calculating the surprise for a given event. For this assessment, the model detects factors that are both directly supportive of the surprising event, and those that are vaguely supportive of it. Using WordNet (cf. Miller, 1995) as a foundation for their knowledge base, they showed this model of surprise to be consistent in predicting people's surprise ratings for a series of short stories with predictable, neutral, and unpredictable outcomes. However, this approach could, possibly, be linked with an expectation account, as degree-of-expectation could be seen as a function of this incoherency score.

Having reviewed the main issues in previous theories and models of surprise, in the remainder of the paper we advance a new sense-making model, the Explanatory Analysis Model of Surprise (EAMoS) based on Foster and Keane's (2015) Metacognitive Explanation-Based (MEB) theory of surprise. Like Maguire et al.'s (2006) model, the present model focuses on several studies of surprise in discourse (see Maguire et al., 2011; Foster & Keane, 2013, 2015). Hence, before presenting the model, we briefly review this empirical evidence.

2. Recent Evidence on Surprise

There is now a considerable body of empirical work on aspects of surprise in discourse (e.g., Grimes-Maguire & Keane, 2005; Gendolla & Koller, 2001; Maguire et al. 2011; Foster & Keane, 2013). This work makes use of simple stories describing surprising events, presented to people before asking them to judge the surprisingness of the outcome. So, for example, for the story in Table 1, people would typically read the key sentences one at a time, before being shown the outcome and asked to rate its surprisingness. This research has revealed several interesting aspects of surprise behavior. For instance, Foster and Keane (2013) have recently shown that some surprising events may be "known" or "less-known"; that is, some

surprising events may have "ready-made" explanations for them, whereas others do not (see also Schank, 1986). Imagine, one day you are walking home and discover that your wallet is missing from the pocket of your jeans. You would be surprised, but also have some ready explanations for what might have occurred (e.g., "Could it have been robbed?", "Might I have dropped it?"). Now, imagine you are walking home and discover that your belt is missing from the waist of your jeans. Again, you would be surprised, but few explanations present themselves (the only one we could think of, eventually, was leaving one's belt in the security area at the airport). So, here, a small change in the object mentioned in the outcome (i.e., wallet or belt) subtly changes the activated knowledge, altering the ease with which the surprise is resolved.

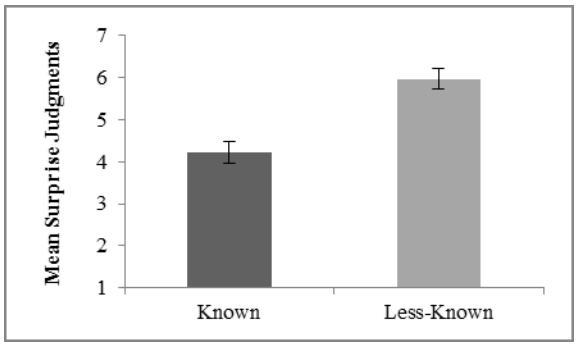


Figure 1: Mean surprise judgments across scenarios for both levels of Outcome-Type (known vs. less-known) (Foster & Keane, 2013, Experiments 1 & 2) with standard errors.

Foster and Keane (2013) operationalized this known/less-known dimension for these story materials using (i) a pre-test sorting task by an independent group of participants and (ii) Latent Semantic Analysis (LSA) coherence scores (cf. Landauer & Dumais, 1997). They showed that these different classes of outcomes either eased or inhibited the process of explanation and, respectively, reduced or increased the experienced surprise associated with the scenario (as measured by surprise ratings). Importantly, Foster and Keane also asked participants to explain the surprising outcomes; thus providing data about the range of possible explanations used (data that forms the basis for the current simulations). The procedures in the Foster and Keane (2013) experiments were similar. In both, participants were asked to read nine stories and to judge the surprisingness of their outcomes. The participants in one condition (explanation) were asked to produce the first explanation they could think of for why the outcome may have occurred, before rating it for surprise; in a second condition (comprehension), participants were asked to answer two simple comprehension questions about the scenario before rating it for surprise. In both experiments participants rated their surprise on a scale from 1-7, with 1: not surprising and 7: very surprising. As predicted, Foster and Keane (2013) found main effects for the known/less-known dimension (see Figure 1). These are the results modeled in the following simulation.

3. Presenting EAMoS: An Explanatory Analysis Model of Surprise

The current novel sense-making model, EAMoS, is based on the MEB theory of surprise (Foster & Keane, 2015). EAMoS takes a different, simpler approach to previous models of surprise, focusing on the structure of the set of explanations for a given surprising event. We posit that every surprising scenario has an explanatory structure, consisting of a space of putative explanations that link the outcome to its preceding setting (see also Leake, 1991; Schank, Kass, & Riesbeck, 1994), and that surprise is resolved by building an explanation to relate the setting to the outcome. The shape of the explanation space for a given scenario determines whether it will furnish explanations that emerge easily (i.e., almost as in "normal" comprehension) or whether this requires more concerted cognitive effort (at the extreme, even involving conscious problem solving).

Following these proposals, the present model conceptualizes the explanation space as a graph of the set of explanations for the surprising event, and analyses the structure of this graph to predict the surprisingness of the outcome. Note, the model has no mechanics for generating explanations *per se*; but rather builds its graph from provided text descriptions of people's explanations gathered in previous experiments.

Table 1: Sample scenario used by Foster & Keane (2013, Experiments 1 & 2).

Setting	Rebecca is on the beach. She goes for a swim in the water	
	Known	**Less-Known**
Outcome	After she dries herself off she notices that her skin has turned red.	After she dries herself off she notices that her skin has turned turquoise.

2.1 How EAMoS Works

Operationally, EAMoS takes scenarios consisting of pairs of setting and outcome inputs (e.g., see Table 1), and explanations that were produced for these scenarios by groups of participants in previous empirical work (Foster & Keane, 2013). No changes were made to these explanations before they were read in to the model, aside from the correction of some spelling mistakes. EAMoS uses these explanations to build an *explanation graph*; that is, a directed graph, *G*, from setting to outcome (e.g., see Fig. 2). It then outputs a surprise rating for the outcome described in the scenario.

2.1.1 Phase One: Populating the Explanation Space The model itself first reads in the setting of the scenario for which it is judging surprise, followed by the explanations and the outcome in question. These settings and outcomes are pre-processed (i.e., punctuation, capitalization, and stop

words[1] are removed, and stems/root-forms are determined where appropriate), and represented as single nodes (see Fig. 2). The explanation text-strings are processed in more detail: The explanations used to populate *G* are pre-processed, but otherwise are used as given by people. After this pre-processing step, *G* is built as the set of nodes; every entity in each explanation string that remains after pre-processing becomes a node in *G*. For each node, a directed edge is added to the graph for (i) every pair of consecutive entities in an explanation, (ii) from the setting to the first entity, and (iii) from the last entity to the outcome.

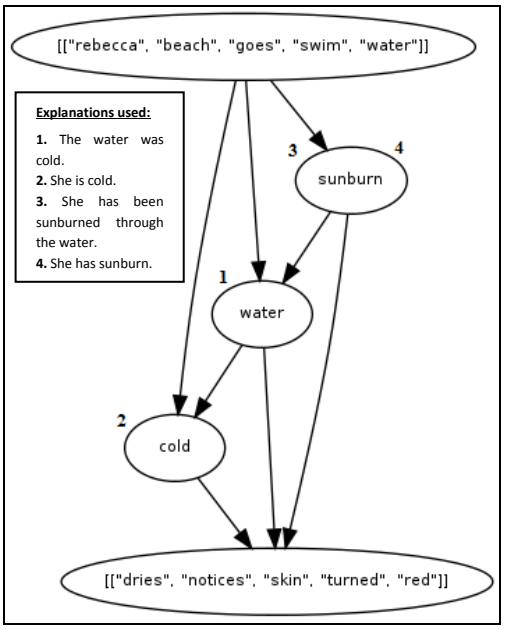


Figure 2: Simplified explanation structure produced by EAMoS for the known variant of the Rebecca scenario (see Table 1) using only four explanations.

If two entities in different explanation strings are the same (such as "water" in the simple explanation structure demonstrated in Fig. 2), then they are represented as a single node in *G*, though the edges between this entity and those consecutive to it may differ for the two different explanations. So, the explanations "she has been sunburned through the water" and "the water was cold" both include "water" in their representation in *G*, but each explanation path can be traced distinctly through *G* from setting to outcome. Thus, the relatively simple computational process described above produces *G*, which encodes the approximate relationship and overlap among all the entities in the provided explanations for each outcome.

2.1.2 Phase Two: Calculating Surprise EAMoS's analysis takes as its core input two variables from *G* to represent the explanatory structure of that scenario: (i) the number of edges, and (ii) the number of nodes. It also counts the total number of given entities (Given Information; Total GI) provided in the setting and outcome, and the number of given entities that are used as nodes in *G* (GI Used). EAMoS uses these variables, scaled by the number of explanations included in the building of *G*, to calculate surprise by applying a function that ascertains the difficulty of explaining the outcome, shown in Figure 3. Theoretically, this function is based on MEB's idea that surprise is based on explanation; the structure of the explanation space is an approximation for how easy or difficult this process will be. Surprise increases as the ratio between edges and nodes increases. Surprise decreases when more of the given entities are used in explanations. This is scaled by the number of explanations that were used to populate *G*.

$$Surprise = 1 - \frac{Edges - \frac{GI\,Used}{Total\,GI}}{Nodes} \times \frac{1}{Explanations}$$

Figure 3: The surprise function used by EAMoS.

2.2 Model Simulation and Evaluation

To evaluate the model, we compared the surprise ratings that EAMoS produces to the ratings produced by participants in two experiments reported by Foster and Keane (2013, Experiments 1 & 2). In this simulation, the model was run on the exact same scenarios presented to the human participants.

2.2.1 Simulation Setup For the purposes of the simulation, the mean surprise rating of each scenario was used. The model took as input each scenario, built the explanations graph for each setting and outcome pair, and produced surprise ratings using the formula in Figure 3. These ratings were then normalized and translated into a number between 1 and 7, to allow for direct comparison with human ratings from Foster and Keane (2013). The two experiments used 18 different story scenarios, all of which were used in the simulation; the mean surprise ratings for each scenario were recovered from the raw data of participant responses.

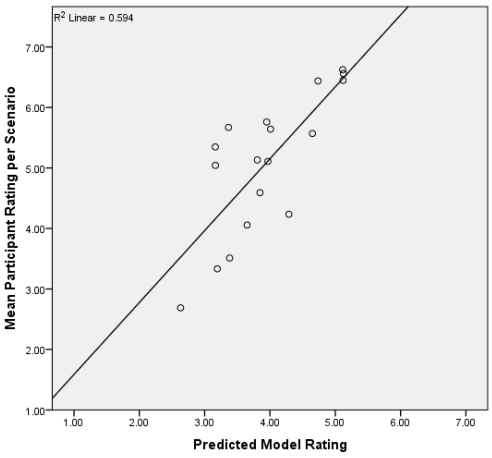


Figure 4: EAMoS's output against human surprise ratings.

[1] Default English stop words list used: www.ranks.nl/stopwords

2.2.1 Results and Discussion EAMoS returned surprise scores that were highly correlated with the human data (N = 95) from Foster and Keane (2013), $r(16) = 0.771$, $p = 0.0002$. At a finer level of detail, EAMoS's surprise ratings correlated with the data for both Experiment 1, $r(16) = 0.737$, $p = 0.0005$, and Experiment 2, $r(16) = 0.779$, $p = 0.0001$. A regression analysis confirmed that EAMoS's output could be used to predict human performance in surprise ratings, again when the total data set was used, $R^2 = 0.594$, $p = 0.0002$ (see Figure 4 for a scatterplot of the relationship between model output and participant means), and individually, for Experiment 1, $R^2 = 0.544$, $p = 0.0005$ and for Experiment 2, $R^2 = 0.607$, $p = 0.0001$. Next, we wanted to see how the model would perform in relation to the two levels of Outcome-type, known and less-known. Accordingly, we performed an independent measures t-test, which revealed a significant difference between the surprise ratings produced by the model for known and less-known Outcome-types, $t(16) = -2.66$, $p = 0.017$, 2-tailed. As expected, the model scored the less-known scenarios as more surprising ($M = 4.36$, $SD = .764$) than the known scenarios ($M = 3.55$, $SD = .504$). This compares favorably to the experimental results and suggests that the model was able to represent the different explanation structures afforded by the two levels of Outcome-type.

Table 2: Sensitivity analysis detailing correlations when weights are varied for nodes and edges in EAMoS's analyses (**Correlation is significant at <.001).

		Edges				
	Weights	**0%**	**25%**	**50%**	**75%**	**100%**
Nodes	**0%**	-.002	-.048	-.047	-.047	-.047
	25%	-.180	.775**	.772**	.771**	.771**
	50%	-.180	.775**	.772**	.771**	.771**
	75%	-.180	.775**	.772**	.771**	.771**
	100%	-.180	.775**	.772**	.771**	.771**

2.3 Sensitivity Analysis: Robustness of Model and Contribution of Different Variables

We then systematically varied the weights of the two core contributing variables (edges and nodes) to ascertain the robustness of the model. Table 2 displays the resulting correlations when varying the weights for the number of nodes (0-100%) and the number of edges (0-100%). As one cannot divide by zero, the top row of the table represents the model output if the node variable is removed entirely from the formula. As can be seen, when either nodes or edges are not taken into account, the correlations are not reliable, whereas, as the formula uses both of these variables in a ratio relationship, equally increasing the weight attached to either of these variables merely scales the variable differently and has no major effect on the model's highly significant correlations with human surprise judgments. Although the correlations are slightly higher when the edges variable is weighted at 25%, we wished to avoid over fitting the model to this data set, so have not altered the formula to reflect this. Overall, these findings suggest that our approach of not weighting the variables separately is a succinct and suitable approach.

3. General Discussion

Previous models have approached the modelling of surprise largely from an expectation-disconfirmation perspective. In this paper we have described a computational model of surprise that takes the novel approach of using explanation structure to predict surprise ratings for a variety of scenarios. Simulations have shown a strong correspondence between predictions made by EAMoS and participant generated surprise ratings, and the model has tested favorably for reliability. Although the scenarios used in the simulation detailed here have been short, simple textual descriptions of events, we believe that the model could be extended to predict surprise in more extended discourse, and in real life situations – indeed, for any situation in which explanations for why the event occurred can be computed .

One future direction that we are currently implementing is to alter the model to predict surprise for individual participants, rather than at the group level. Another fruitful direction could be to include a semantic knowledge base; although the simplicity of this model works well in predicting human surprise, allowing the model to match, say, "sea" with "water" in explanations for the Rebecca scenario described above, may provide even more accurate predictions.

In conclusion, this work has shown that surprise can be predicted by a simple analysis of explanations that link preceding settings with target surprising outcomes. These initial simulations are promising, and even in the simple form presented here the model correlates strongly with people's surprise ratings. In addition to providing further support for the MEB theory of surprise by illustrating that explanation plays a key role in surprise, these results point to promising future research directions for surprise that have not been previously explored.

Acknowledgments

This research was supported by a PhD Scholarship to the first author from the School of Computer Science & Informatics, University College Dublin, Ireland.

References

Bae, B.C., & Young, R. M. (2008). A use of flashback and foreshadowing for surprise arousal in narrative using a plan-based approach. In U. Spierling & N. Szilas (Eds.), *Proceedings of ICIDS, LNCS,* 5334. Heidelberg: Springer.

Baldi, P., & Itti, L. (2010). Of bits and wows: a Bayesian theory of surprise with applications to attention. *Neural Networks, 23*(5), 649-666.

Connell, L., & Keane, M. T. (2006). A model of plausibility. *Cognitive Science, 30*(1), 95-120.

Darwin, C. R. (1872). *The expression of the emotions in man and animals.* London: John Murray.

Ekman, P., & Friesen, W. (1971). Constants across cultures in the face and emotion. *Journal of Personality and Social Psychology, 17*(2), 124–9.

Foster, M. I., & Keane, M. T. (2013). Surprise! You've got some explaining to do. In M. Knauff, M. Pauen, N. Sebanz, & I. Wachsmuth (Eds.), *Proceedings of the 35th Annual Conference of the Cognitive Science Society*. Austin, TX: Cognitive Science Society.

Foster, M. I., & Keane, M. T. (2015). Why some surprises are more surprising than others: Surprise as a metacognitive sense of explanatory difficulty. Manuscript under review.

Gendolla, G. H., & Koller, M. (2001). Surprise and motivation of causal search: How are they affected by outcome valence and importance? *Motivation and Emotion, 25*(4), 327-349.

Grimes-Maguire, R. & Keane, M. T. (2005). Expecting a surprise? The effect of expectations in perceived surprise in stories. In *Proceedings of the 27th Annual Conference of the Cognitive Science Society.* Hillsdale, NJ: Erlbaum.

Heider, F.(1958). *The psychology of interpersonal relations.* New York: John Wiley & Sons.

Itti, L., & Baldi, P. (2006). Bayesian surprise attracts human attention. *Advances in neural information processing systems, 19,* 547-554.

Itti, L., & Baldi, P. (2009). Bayesian surprise attracts human attention. *Vision Research, 49*(10), 1295-306.

Izard, C. (1977). *Human emotions.* Plenum Press, New York.

Kahneman, D., & Miller, D. T. (1986). Norm theory: Comparing reality to its alternatives. *Psychological Review, 93*(2), 136-153.

Keil, F. C. (2006). Explanation and understanding. *Annual Review of Psychology, 57*, 227-254.

Landauer, T. K., & Dumais, S. T. (1997). A solution to Plato's problem: The latent semantic analysis theory of the acquisition, induction, and representation of knowledge. *Psychological Review, 104*(2), 211-240.

Leake, D. B. (1991). Goal-based explanation evaluation. *Cognitive Science, 15*(4), 509-545.

Lombrozo, T., & Carey, S. (2006). Functional explanation and the function of explanation. *Cognition, 99*(2), 167-204.

Lorini, E., & Castelfranchi, C. (2006). The unexpected aspects of surprise. *International Journal of Pattern Recognition and Artificial Intelligence, 20*(6), 817-835.

Lorini, E., & Castelfranchi, C. (2007). The cognitive structure of surprise: looking for basic principles. *Topoi: An International Review of Philosophy, 26*(1), 133-149.

Macedo, L., & Cardoso, A. (2001). In G. Wiggins (Ed.), *Proceedings of the AISB'01 Symposium on Artificial Intelligence and Creativity in Arts and Science*. United Kingdom: University of York.

Macedo, L., Cardoso, A., Reisenzein, R., Lorini, E., & Castelfranchi, C. (2009). Artificial surprise. In Vallverdú, J. & Casacuberta, D. (Eds.) *Handbook of research on synthetic emotions and sociable robotics: New applications in affective computing and artificial intelligence*. UK: Information Science Reference.

Macedo, L., Reisenzein, R., & Cardoso, A. (2004). Modeling forms of surprise in artificial agents: Empirical and theoretical study of surprise functions. In K. Forbus, D. Gentner & T. Regier (Eds.), *Proceedings of the 26th Annual Conference of the Cognitive Science Society*. Mahwah, NJ: Erlbaum.

Maguire, R., Costello, F., & Keane, M. T. (2006). A cognitive model of surprise judgements. In R. Son (Ed.), *Proceedings of the 28th Annual Conference of the Cognitive Science Society*. Hillsdale, NJ: Erlbaum.

Maguire, R., Maguire, P., & Keane, M. T. (2011). Making sense of surprise: An investigation of the factors influencing surprise judgments. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 37*(1), 176-186.

Meyer, W. U., Reisenzein, R., & Schützwohl, A. (1997). Toward a process analysis of emotions: The case of surprise. *Motivation and Emotion, 21*(3), 251-274.

Miller, G. A. (1995). WordNet: a lexical database for English. *Communications of the ACM, 38*(11), 39-41.

Ortony, A., & Partridge, D. (1987). Surprisingness and expectation failure: What's the difference? In *Proceedings of the 10th International Joint Conference on Artificial Intelligence*. San Francisco: Morgan Kaufmann Publishers.

Ortony, A., & Turner, T. J. (1990). What's basic about basic emotions? *Psychological Review, 97*(3), 315-331.

Ranganath, C., & Rainer, G. (2003). Neural mechanisms for detecting and remembering novel events. *Nature Reviews Neuroscience, 4*(3), 193-202.

Reisenzein, R., & Studtmann, M. (2007). On the expression and experience of surprise: no evidence for facial feedback, but evidence for a reverse self-inference effect. *Emotion, 7*(3), 601-611.

Schank, R. C. (1986). *Explanation patterns: Understanding mechanically and creatively.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Schank, R. C., Kass, A., & Riesbeck, C. K. (Eds.) (1994). *Inside case-based explanation.* UK: Lawrence Erlbaum Associates.

Teigen, K. H., & Keren, G. (2002). When are successes more surprising than failures? *Cognition and Emotion, 16*(2), 245-268.

Teigen, K. H., & Keren, G. (2003). Surprises: Low probabilities or high contrasts? *Cognition, 87*(2), 55-71.

Tomkins, S. S. (1962). *Affect, imagery, consciousness.* (Vol. 1). New York: Springer.

Reconciling two computational models of working memory in aging

Violette Hoareau Benoît Lemaire Sophie Portrat
(<first name>.<last name>@upmf-grenoble.fr)
Univ. Grenoble Alpes, LPNC, F-38040 Grenoble, France
CNRS, LPNC UMR5105, F-38040, Grenoble, France

Gaën Plancher (Gaen.Plancher@univ-lyon2.fr)
Univ. Lyon 2, EMC, 5 avenue Pierre Mendès-France, F-69676, Bron, France

Abstract

It is well known that working memory performance changes with age. Two recent computational models of working memory, TBRS* and SOB-CS, corresponding to two distinct causes of forgetting, namely time-based decay and interference, are applied on a set of complex span data produced by young and older adults. As expected, these models are unable to account for the older adult data. An investigation on the effect of the main parameters of these models showed that the poorer performance of older adult does not come from a weaker encoding of items, or even a longer time spent on distractors, but rather on difficulties during the free time that immediately follows each distractor, as well as a higher level of confusion between items. These results are discussed with respect to the current theories of working memory and aging.

Keywords: working memory; aging; TBRS; SOB-CS.

Introduction

Working memory is a cognitive construct that describes how information can be maintained for a limited period of time, while concurrent processing is also performed. Several computational models of working memory have been proposed in the last decades. Most of them concern young adults. However, it is known that working memory tends to decline with age (Logie & Morris, 2004) for reasons that are not completely understood. Indeed, several explanations have been proposed and the question is still under debate. This paper is an attempt to contribute to the debate by means of a computational modeling approach.

To this end, we will test two recent theoretical models that propose two distinct mechanisms to account for forgetting in working memory: time-based decay and interference. Despite a strong opposition between these two models in the recent literature, we will show that adapting each one to reproduce older adult data leads to similar conclusions about the reason of the older adult working memory loss of performance.

The first model, named TBRS for Time-Based Resource Sharing (Barrouillet, Portrat & Camos, 2011) claims that our difficulty to maintain several items in memory while performing distracting tasks in-between their presentation comes from the fact that item activation decays with time as soon as attention is directed towards another item or a distractor. Hence, according to TBRS, working memory performance depends on the cognitive load of the processing task, which is defined as the proportion of time during which this task captures attention. This model is supported by several experiments using a complex span design (e.g. Barrouillet, Bernardin, Portrat, Vergauwe, & Camos, 2007; Portrat, Barrouillet, Camos, 2008; Vergauwe, Barrouillet, & Camos, 2010).

The second model, called SOB-CS for Serial Order in a Box – Complex Span (Oberauer & Lewandowsky, 2012), has a completely different point of view. It is based on the idea that forgetting is not based on decay but rather on the effect of interference between items or between items and distractors. The interference from distractors depends on the strength of their encoding and this strength relies on the novelty of the to-be-processed items. This novelty varies with the number of to-be-processed items and their similarity: the more the number of items, the poorer the recall and the more similar the items, the better the recall performance.

There has been a strong debate in the literature in the past years between these two models (Plancher & Barrouillet, 2013; Lewandowsky, Geiger, Morrel, & Oberauer, 2010). It is therefore useful to challenge both models by testing how they would account for older people data, in particular because older people present reduce attentional capacities (Luo & Craik, 2008) but are also more sensitive to interference (Hasher, Zacks, & May, 1999). We therefore first present the data that we collected on a complex span task on young and older people.

Experiment

Procedure and Material

In a serial recall task, participants were presented with 5 images in-between which they had to read aloud 3 distractor words. They were then asked to recall the image names in order. Such a trial was repeated 16 times. In order to study both a possible interference effect and a time effect which are markers of SOB-CS and TBRS respectively, we defined two variables:

- the novelty of distractors which either contain repetitions (low interference, e.g., *duck, duck, duck* or *duck, duck, horse*) or all distinct (high interference, e.g., *duck, plane, horse*);

- the duration in-between distractors, which could be long (slow pace, one word to read every 2 seconds) or short (fast pace, one word to read every 1.2 seconds).

There were therefore 4 experimental conditions resulting from two types of novelty of the distractors (repeated vs. novel words) and two paces of the processing task (fast vs. slow), with then four trials in each condition. Repeated distractors are generally three identical words (AAA), but we also used patterns in which only two are identical and the third one different (called ABA, ABB or AAB), in order to prevent participants from anticipating the distractor.

Participants

20 young participants (12 females; mean age = 21.62; SD = 2.51) and 20 healthy older participants (13 females; mean age = 71.92; SD = 5.18) voluntarily took part in this experiment.

Results

As expected, the two populations behave differently. Older participants recalled fewer images (2.80) than younger participants (3.78), $F(1,38)=27.77$, $p<.001$. An interesting finding is that older adults did not spend more time to process distractors (489 ms in average) compared to young adults (527 ms). Their worse recall performance therefore does not come from a longer time spent on distractors.

We now present two sets of simulations performed on two computational models that are able to simulate a working memory trial, TBRS* and SOB-CS. Each model is exposed to 5 items during 1500 ms each. In-between each presentation of items, three distractors are presented during a specific duration that corresponds to the time actually spent by participants for reading a distractor word. Models then simulates the recall phase at the end of each trial. When asked to recall an item at a given position, models could, exactly like participants, recall the correct item, recall a wrong one or even do not recall anything if none of them is activated enough in memory.

TBRS*

Description

TBRS* (Oberauer & Lewandowsky, 2011) implements the verbal theory (Barrouillet et al., 2011) which assumes that the core component of working memory is attention. If attention is directed towards an item, its activation value is increased and the activation values of all other items is decreased. TBRS* is based on a two-layer connectionist network. One layer is composed of nodes representing the items to be memorized and the other layer encodes the sequential position of items. Each position is coded by a subset of position units, so that two adjacent positions share a proportion of P units. Memorizing is modeled as a process of connecting positions with items, by Hebbian learning (Anderson, 1995). The strength of the increase of any connection weight (w) depends on a strength value (η) and it is bound by an asymptote L, defined in such a way that the total activation strength of an item is always between 0 and 1: $\Delta w = (L-w)\eta$. The strength depends on the time t devoted to encoding as well as a stochastic parameter r modeling human variability: $\eta = 1 - e^{-r.t}$ with $r = \mathcal{N}(R, s_2)$.

For instance, if the sequence of letters to be memorized is KZFP, K is first encoded which results in strengthening the links between item K and the nodes coding for position 1. When attention is captured by another task, like reading a word in our case, those values w decrease according to an exponential function: $w(t) = w_0.e^{-D.t}$.

When attention is redirected towards the memory task, a refreshing process takes place and leads to an increase of the w values. All positions are successively considered, starting with the first one, and the most activated item at each position is retrieved and refreshed. In order to simulate retrieval errors, a Gaussian random noise, defined by its standard deviation , is added to each item node before the best one is selected.

This refreshing process cycles until a new activity requires attention.

To pursue our example, when Z is encoded, activation values between the node representing Z and the node representing position 2 are strengthened (while in the meantime, the activation values of K are decreased). If there is time for refreshing, it is alternately done between the items retrieved at position 1 (K if there is no retrieval error) and the one at position 2 (Z in most cases).

Comparison to experimental data

TBRS* was run[1] 5 000 times on each experimental condition (slow or fast pace repeated or unrepeated distractors) for young and older adults, using the default parameters suggested by Oberauer & Lewandowsky (2011). Since TBRS* does not model interference between distractors, the experiment was simulated by computing the durations of processing distractors according to the different patterns of repetition dicussed previously: AAA, ABA, ABB, AAB and ABC. The only difference between young and older adult simulations comes from the variability of the time used by participants to process a distractor, as mentioned previously. Results are presented in Table 1.

As expected, it turned out that the model reproduces quite well the young adult performance but it cannot account for the older adult data.

Several TBRS* parameters could be tuned to better reproduce the lowest performance of older adults, representing thus possible causes of forgetting in WM. We investigated the effects of the level of noise (σ) which controls the amount of retrieval errors, the decay rate (D), the encoding strength (R) and the duration for refreshing an item in the refreshing cycle (T_r). We performed a grid search in this 4-dimensional space and computed for each point the root mean square error (RMSE) between each model score under the 4 conditions and the averaged experimental data. We then studied the effect of each parameter by performing a projection on the parameter dimension, and analyzing the evolution of the

[1] data and model codes that have been used in this work are available on the first author webpage

Table 1: Mean number of items recalled in each condition, for young and older adults (observed/Data and simulated/TBRS* and SOB-CS). Both models used default parameters.

FAST PACE

	Low Interference		High Interference	
	Young	Older	Young	Older
Data	3.51	2.55	3.69	2.88
TBRS*	3.51	3.52	4.33	4.32
SOB-CS	3.64	3.67	3.57	3.59

SLOW PACE

	Low Interference		High Interference	
	Young	Older	Young	Older
Data	3.90	3.04	4.03	2.73
TBRS*	4.19	4.18	4.43	4.40
SOB-CS	3.98	3.99	3.97	3.99

average RMSE.

Figure 1 and 2 show the average RMSE as a function of various values for the noise σ and he duration of atomic refreshing T_r, for both young and older adults.

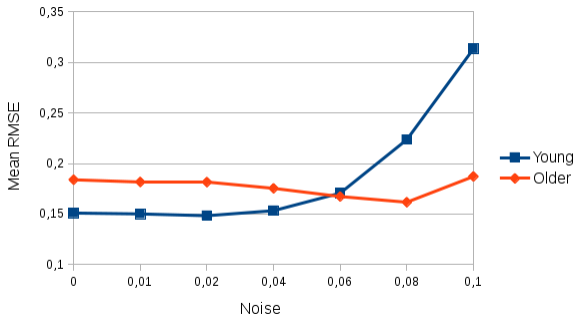


Figure 1: RMSE between TBRS* simulation and data as a function of the noise parameter σ.

It turned out that the models better fit the older adult data for a higher level of noise (0.08) compared to the young adult data for which the best RMSE is for a low level of noise, coherent with the default value of 0.02 proposed by Oberauer & Lewandowsky (2011) for young adults. The higher that noise, the more likely retrieval errors. This could be the sign of a weaker inhibition ability for the older population or a higher sensitivity to interference.

The duration for refreshing a single item during free time, in the refreshing loop, also needs to be adjusted to reproduce the older adult data. That parameter was set to an average value of 80 ms in the original model, which, for instance, permits to make a full cycle of refreshing the five items in about 400 ms.

The best RMSE for young adults is now obtained for a value of 40ms, half the default value of the original model

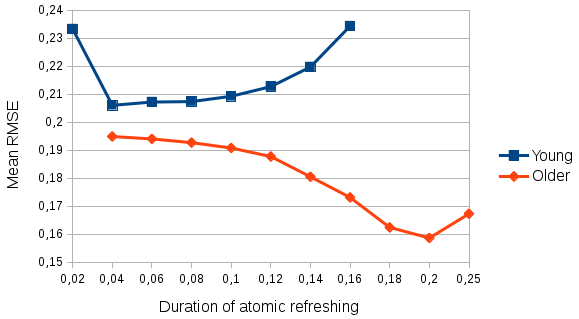


Figure 2: RMSE between TBRS* simulation and data as a function of the duration of atomic refreshing T_r.

but coherent with Portrat & Lemaire (in press) which showed that this value has to be decreased if the model has an attentional focus size of only one item at a time. As expected, the best RMSE based on the older adult data is obtained for a much higher value of about 200 ms for refreshing a single item, which is 5 times the duration of the young adult model.

However, we could not find any difference between young and older people concerning the rate of encoding strength, nor a significant difference between decay rates.

To summarize, two parameters need to be adjusted to fit older adult performance. First, the noise during retrieval for refreshing or recall has to be increased. Second, the duration for refreshing a single item during the free time available in-between processing steps has also to be substantially enlarged.

One interesting finding of that simulation is therefore that the older population would not suffer from a lack of encoding, but rather from difficulties in taking advantage of the free time that occurs after each distractor, either because of inhibition difficulties or defaults in managing interference (parameter σ) or because it takes time for them to refresh items (parameter T_r). We now present the second model that we used to simulate our data .

SOB-CS

Description

SOB-CS (Oberauer & Lewandowsky, 2012) assumes that working memory limitation is due to interference between to-be-maintained items or between items and distractors. This model is also based on a two-layer connectionist network that associates a distributed item representation with distributed position markers. Contrary to TBRS*, item representation is distributed in order to reproduce interference between items, distractors and both according to their similarity. For instance, if items are highly similar they share patterns across the same set of units, and inversely, different items are represented with very different patterns.

Memory is maintained by standard Hebbian learning (Anderson, 1995): $\Delta W = \eta_e(i).W$ where $W = v_i p_i^T$ represents the weight matrix connecting the ith position markers p_i with

ith item representation v_i and $\eta_e(i)$ represents the encoding strength which depends on the time spent to encode (t_e), the rate of encoding R and the item's novelty $A(i)$ by means of $\eta_e(i) = A(i)(1 - e^{-t_e \cdot r})$. Item's novelty reflects the degree of mismatch between the expectation (computed as $W.p_i$) and the actual item. The higher the novelty, the stronger the encoding.

SOB-CS assumes that, during the processing step, distractors are encoded in the same way as items such as $\Delta W = \eta_e(i,k).d_{i,k}.p_i^T$ where $d_{i,k}$ represents the distractor k following item i and $\eta_e(i,k)$ is the encoding strength of the distractor. That is the reason why a distractor following item i creates interference on this item. Because of the item's novelty notion, repeatedly processing the same distractors produces less interference than does processing different distractors.

After each processing step, more or less free time is available which allows restoration of an unimpaired memory state. The removal of the distractor has been modeled by Hebbian antilearning : $\Delta W = -\eta_r(i,k).d_{i,k}.p_i^T$ where $\eta_r(i,k)$ is the antilearning strength which depends on the free time t_f, the rate of removal r of representations from working memory and the asymptotic value $\Omega(i,k)$.

Finally, for the recall step, the position markers are used as cues to determine which items to recall. For instance, to recall the ith item, the vector position p_i is considered to compute $v_i' = W.p_i$, which is the distorded version of the original vector v_i. To retrieve this original item v_i within all the candidates item of the list, the model computes all the probabilities of recalling an item j depending on the similarities $s(v_i', v_j) = e^{-c.D(v_i', v_j)^2}$ with c, the discriminability parameter and D the euclidian distance between v_i' and v_j. Before each recall item, a Gaussian noise with a standard deviation N_o is added to represent output interference.

Comparison to experimental data

As previously described for the TBRS* simulation, we first simulated the young and older adult data, with the default parameters suggested by Oberauer & Lewandowsky (2012), using also 5000 runs for each condition. As in TBRS*, the difference between young and older adult simulation comes from the variability of the time used by participants to process a distractor. However, contrary to TBRS*, the fact that distractors are repeated or not in the experiment is taken into account by SOB-CS. Results are presented in Table 1. Like TBRS*, SOB-CS reproduces more accurately the young adult performance than the older adult one.

Hence, we studied the effect of four parameters proper to this model to better fit the performance of older adult: the encoding rate (R), the removal rate (r), the standard deviation (N_o) of the Gaussian noise added to each weight in W after recall of each item and the discriminability (c) between recall candidates which controls the level of confusion between retrieval candidates.

Figures 3, 4 and 5 show the average RMSE as a function of the various values for the removal rate (r), the discriminability (c) between recall candidates and the standard deviation N_o of Gaussian noise respectively for both young and older adults. The model better fits the older adult data for a lower discriminability (0.9) compared to young adult data for which the optimal RMSE value appears to be much higher, even higher than the default value (1.3) proposed by Oberauer & Lewandowsky (2012). With lower values of c, similarity falls off less steeply with distance, so that the most similar candidate is less clearly discriminated from the less similar ones. In accordance with a decline in inhibition with aging (Hasher & Zacks, 1988 ; Hasher, Zacks, & May, 1999), this lower discriminability for old people could explain more intrusion errors that have been found in the recall of older participants (e.g. Carretti, Cornoldi, De Beni, & Palladino, 2004 ; Hedden, & Park, 2001).

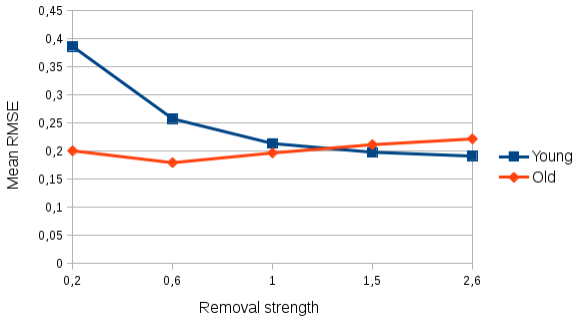


Figure 3: RMSE between SOB-CS simulation and data as a function of the removal rate strength r.

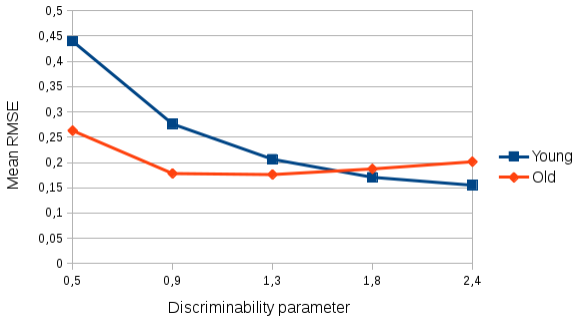


Figure 4: RMSE between SOB-CS simulation and data as a function of the discriminability parameter c.

We also observed that the model simulating older adults needs more time to remove the previous distractor during the free time period than the one based on young adults. The RMSE based on the older adult data is lower for a removal rate (r) of 0.6 whereas the RMSE based on the young adult data is lower for a higher removal rate (1.5 or more), coherent with the default value proposed by Oberauer & Lewandowsky (2012) for young adults. This result is also in line with the inhibition explanation of working memory aging (e.g., Hasher

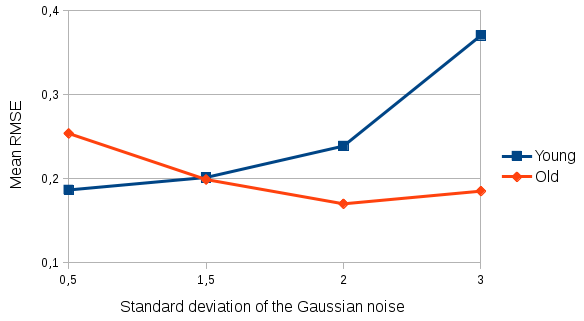


Figure 5: RMSE between SOB-CS simulation and data as a function of standard deviation N_o of the Gaussian noise.

& Zacks, 1988). Older participants would have difficulties to suppress irrelevant information in WM.

The output noise parameter N_o in SOB-CS, which occurs only in the recall step, has also to be modified for a good simulation of the older people performance. Increasing the noise is therefore a good way to simulate older people data.

Finally, as with TBRS*, we could not find any difference between young and older people concerning the rate of encoding strength. An important finding of that second simulation is that the conclusion is exactly the same as with TBRS* simulation: older population would not suffer from a lack of encoding, but rather from difficulties in taking advantage of the free time that occurs after each distractor, either because of inhibition difficulties or defaults in managing interference (parameter c and N_o) or because it takes time for them to remove distractors (parameter r).

Discussion

The aim of the present paper is to give more understanding of working memory aging through the comparison of behavioral data collected on young and old adult with simulations from two very recent and influential models of working memory. The first outcome is that they produced results that are coherent with each other, in spite of their very different theoretical foundations. First, it appears that the strength with which items are encoded does not have to be weaker for the models simulating older adult performance. We could not find any difference between young and older adults for the R parameter in TBRS*. That is exactly the same for the encoding parameter in SOB-CS. This finding tends to indicate that the lower recall performance of older adults would not be due to a lack of encoding the to-be-recalled items. However, the difference between young and older adult performance can be explained by two kinds of parameters, both controlling what is happening during the free time following the processing of each distractor or during recall. The first parameters control the likelihood of confusion between items when retrieving one at a given position whereas the second ones controls the post-distractor processes. In addition, we computed the AIC for both models using the probability mass functions of recalling x items, generated from a binomial model. We found that the two original models do not differ much: on young adult data, $AIC_{TBRS} = 199.03$ and $AIC_{SOB-CS} = 196.02$; on older adult data, $AIC_{TBRS} = 318.74$ and $AIC_{SOB-CS} = 310.57$.

Defaults in the retrieval processes

In the TBRS* simulations, a much higher noise during refreshing and recall has to be set to account for the older adult data. That Gaussian noise, which directly boosts confusion between items, is added to the activation value of each candidate item, each time the model has to retrieve an item before refreshing it or recalling it. The processes in charge of the retrieval of items given a position seems therefore to be affected for the older adults.

A similar result was obtained from the SOB-CS simulations. The parameter controlling discriminability between items has to be decreased to account for the older adult data. A low discriminability produced retrieval errors because the best candidate can be mixed up with other candidates. This parameter is highly similar to the noise parameter in TBRS* because a higher noise added to the activation values of candidate items also introduces some confusion between items.

Defaults in the post-distractor processes

The other kind of parameters that has to be changed in the simulations of older adult data concerns the processes that appear right after processing a distractor. In the TBRS* simulations, to simulate older adults data, the model has to spend 5 times more time to refresh a single item than the default value. It could be that older people needs more time to refresh items but it could also be that they spend time to switch from processing distractor to maintaining items.

This result is in line with what has to be modified in SOB-CS, that is the parameter that controls the removal of the previous distractor, in order to reinstate the correct state of memory. Once again, this process occurs right after a distractor has been presented. According to the results of both model simulations, it is therefore right after the processing phase, when participants have to switch from a distracting activity to a process trying to maintain items vivid in memory, that something goes amiss in older adults.

Theoretical explanations

There are several theories in the literature to explain the lower working memory performance of older people. In this section, we focus on three theories. One strong explanation comes from a deficit of inhibition control (Hasher & Zacks, 1988; Hasher, Zacks, & May, 1999). Older participants would have difficulties to suppress irrelevant information in WM and, in consequence, access to relevant information would be reduced. Our simulations are coherent with that explanation because the processes that have to be altered in the computational models to account for older adult data are precisely those at the frontier between processing a distractor and taking advantage of the free time. And this moment is when an inhibition process occurs.

Another explanation that is proposed in the literature (Salthouse, 1996) suggests that aging goes along with a slower processing speed. The encoding phases of the two models do not have to be modified to account for the data, but we cannot conclude that older people spend as much time as young people to encode items because of the experimental design: duration presentation is probably too long (1.5s) to observe differences of strength encoding at the end of the encoding phases.

Finally, another hypothesis is linked to the deficit of switching between storage and processing. According to Verhaeghen and colleagues (Vaughan, Basak, Hartman, & Verhaeghen, 2008), focus-switching might be a good candidate for the locus of age differences in WM. In accordance, difficulties to remove distractors (SOB-CS) or to refresh items (TBRS*) for older adults could be viewed as a symptom of a longer switching mechanism between processing and storage instead of a deficit of inhibition. Similarly, a higher rate of confusion between items could be the sign of lower accuracy of the process of switching items in and out of the focus of attention (Verhaeghen & Basak, 2005).

TBRS* and SOB-CS are concurrent models simulating working memory in different ways. The present study does not aim at choosing the best of them. On the contrary, we take advantage of both of them to investigate the possible causes for the decline of working memory performance in aging. We found that, in spite of theoretical divergences between these two models, simulations tend to the same conclusions: older people seems to have difficulties of taking advantage of the free time and more confusion between items. It would therefore be interesting in the future to model more precisely that specific moment where the default seems to occur, which is the switching between processing and storage.

Acknowledgments

We would like to thank Hélène Boyer who participated in the data gathering. We also acknowledge the French Regional Council named "Région Rhône-Alpes" which funded that work as well as provided a PhD grant to the first author.

References

Anderson, J. A. (1995). An introduction to neural networks. *MA: MIT Press*.

Barrouillet, P., Bernardin, S., Portrat, S., Vergauwe, E., & Camos, V. (2007). Time and cognitive load in working memory. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *33*, 570-585.

Barrouillet, P., Portrat, S., & Camos, V. (2011). On the law relating processing to storage in working memory. *Psychological Review*, *118*, 175-192.

Carretti, B., Cornoldi, C., Beni, R. D., & Palladino, P. (2004). What happens to information to be suppressed in working memory tasks? short and long term effects. *The Quaterly Journal of Experimental Psychology*, *57A*, 1059-1084.

Hasher, L., & Zacks, R. (1988). Working memory, comprehension, and aging: A review and a new view. In G. Bower (Ed.), *The psychology of learning and motivation.* New York: Academic Press.

Hasher, L., Zacks, R., & May, C. (1999). Inhibitory control, circadian arousal, and age. In D. Gopher & A. Koriat (Eds.), *Cognitive regulation of performance: Interaction of theory and application.* Cambridge, MA: MIT Press.

Hedden, T., & Park, D. (2001). Aging and interference in verbal working memory. *Psychology and Aging*, *16*, 666-681.

Lewandowsky, S., Geiger, S., Morrel, D., & Oberauer, K. (2010). Turning simple span into complex span: Time for decay or interference from distractors? *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *36(4)*, 958-78.

Logie, R., & Morris, R. (2014). Working memory and ageing. *Hove, UK: Psychologie Press*.

Luo, L., & Craik, F. (2008). Aging and memory: A cognitive approach. *La revue canadienne de psychatrie*, *53*, 346-353.

Oberauer, K., & Lewandowsky, S. (2011). Modeling working memory: a computational implementation of the time-based resource-sharing theory. *Psychonomic Bulletin & Review*, *19*, 779-819.

Oberauer, K., Lewandowsky, S., Farrell, S., Jarrold, C., & Greaves, M. (2012). Modeling working memory: an interference model of complex span. *Psychonomic Bulletin & Review*, *19*, 779-819.

Plancher, G., & Barrouillet, P. (2013). Forgetting from working memory: Does novelty encoding matter? *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *39(1)*, 110-25.

Portrat, S., Barrouillet, P., & Camos, V. (2008). Time-related decay or interference - based forgetting in working memory? *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *34*, 1561-1564.

Portrat, S., & Lemaire, B. (in press). Is refreshing in working memory sequential? a computational modeling approach. *Cognitive Computation*.

Salthouse, T. (1996). The processing-speed theory of adult age differences in cognition. *Psychological Review*, *103*, 403-428.

Vaughan, L., Basak, C., Hartman, M., & Verhaeghen, P. (2008). Aging and working memory inside and outside the focus of attention: dissociations of availability and accessibility. *Neuropsychology, Development, and Cognition. Section B, Aging, Neuropsychology and Cognition*, *15*, 403-703-724.

Vergauwe, E., Barrouillet, P., & Camos, V. (2010). Do mental processes share a domain-general resource? *Psychological Science*, *21(3)*, 384-90.

Verhaeghen, P., & Basak, C. (2005). Aging and switching of the focus of attention in working memory: Results from a modified n-back task. *Quarterly Journal of Experimental Psychology*, *58A*, 134-154.

Stability of Individual Parameters in a Model of Optimal Fact Learning

Florian Sense (f.sense@rug.nl)
Department of Experimental Psychology and Department of Psychometrics and Statistics,
Groningen, The Netherlands

Friederike Behrens (f.behrens@student.rug.nl)
Research School of Behavioral and Social Sciences,
Groningen, The Netherlands

Rob R. Meijer (r.r.meijer@rug.nl)
Department of Psychometrics and Statistics,
Groningen, The Netherlands

Hedderik van Rijn (d.h.van.rijn@rug.nl)
Department of Experimental Psychology and Department of Psychometrics and Statistics,
Groningen, The Netherlands

Abstract

We are using an algorithm based on a computational model of human memory to optimize the scheduling and repetition of individual items within a learning session. The model estimates the rate of forgetting for each participant to determine the order in which items should be repeated and to decide when previous items have been learned well enough to introduce a novel item. To improve the model further, we conducted an experiment to test how stable the parameter estimates are over time and across different materials. We have found that estimated rates of forgetting are stable over time *within* one type of material but not across different types of material. This finding has important implications for how information about a learner should be preserved between study sessions.

Keywords: spacing effect; testing effect; cognitive model; learning; parameter stability.

Introduction

Fact learning is a big part of learning a new skill. In many school curricula, students are evaluated based on how well they learned a certain array of facts. With the advance of computers into classrooms and workplaces, tutoring systems have been developed to help learners master the required material. Over a hundred years of memory research have singled out two robust effects that developers of such systems can use to enhance that goal: the spacing effect and the testing effect. By making optimal use of both of them *and* adjusting the system to the individual learner, such systems can make learning a lot more efficient. As of now, however, most optimizing systems treat each learning session in isolation; user-specific characteristics are estimated during a learning session to optimize each learning session but are not preserved *between* learning sessions. In this study, we investigated to which extend user-specific parameters relevant to such a tutoring system are stable over time and across different materials to gauge to which extent they can be preserved between learning sessions.

The tutoring system used here works by balancing the benefits of the spacing and the testing effect. The spacing effect describes the finding that performance on tests of recall is improved when study time is distributed over multiple sessions with time in-between rather than massed study (Cepeda, Pashler, Vul, Wixted, & Rohrer, 2006; Dempster, 1988). The optimal spacing schedule ultimately depends on how much time is available and when the material is tested (Cepeda, Vul, Rohrer, Wixted, & Pashler, 2008). However, it has been shown convincingly that long-term retention can be increased by spacing items within a single learning session (Lindsey, Shroyer, Pashler, & Mozer, 2014; van Rijn, van Maanen, & van Woudenberg, 2009) as well as spacing individual learning sessions (Cepeda et al., 2006).

The testing effect, on the other hand, describes the finding that active memory retrieval during practice is more beneficial for long-term retention than passive study (Karpicke & Roediger, 2008; Roediger & Butler, 2011). That is, being forced to retrieve the answer from memory leads to better learning than simple re-studying (i.e. looking at) the cue-answer pair (Carrier & Pashler, 1992). This effect has been studied extensively in the laboratory (Cepeda et al., 2008) but also holds in more realistic classroom settings (Agarwal, Karpicke, Kang, Roediger, & McDermott, 2008; van Rijn et al., 2009).

Given our knowledge of the spacing and testing effects and the quasi-lawful behavior of memory, it seems possible to devise a learning schedule that would make optimal use of each effect's benefits. This would require balancing two seemingly opposing goals: (1) maximizing time between repetitions of an item to get the biggest spacing effect, and (2) minimizing time between repetitions of an item to make sure it can still be retrieved from memory to take advantage of the testing effect. Such computer adaptive practice models have been developed and have been shown to outperform flashcard control conditions (Nijboer, 2011; van Rijn et al., 2009).

As a starting point for the development of such models, Anderson and Schooler (1991) showed that data on memory performance (i.e. practice and retention) across time courses ranging from seconds to years can be fit nicely by power functions. Interestingly, this corresponds closely with environmental relationships (Anderson & Milson, 1989). That is, the likelihood that people still remember a non-sense syllable they learned today at a certain point in the future (i.e. the original Ebbinghaus data) can be described with the same power functions that can be used to describe the likelihood of receiving an e-mail (Anderson & Schooler, 1991). This leads Anderson and Schooler (1991) to conclude that the human "memory system is adapted to the structure of the environment" (p. 400).

Based on this assumption, it is argued that the practice and retention of facts can be approximated using the same equations that can be used to describe the behavioral effects in the data. Pavlik and Anderson (2003, 2005) developed a model that formalizes this process and show how it can be used to compute the optimal schedule of practice, taking into account the effects of practice, retention, and spacing (Pavlik & Anderson, 2008). Their model assumes that there is some stable effect based on each individual's *rate of forgetting* and additional effects based on *item difficulty*. In this original work, it is assumed that these effects are stable over time and, for the rate of forgetting, across knowledge domains/materials. That is, someone's rate of forgetting is assumed to be a property of their memory and therefore stable, regardless of whether they study vocabulary, topographical information, or word definitions.

The success of such models (Nijboer, 2011; Pavlik & Anderson, 2008; Van Rijn et al., 2009) is very promising but the stability of participants' rate of forgetting across time and knowledge domains has never been demonstrated empirically. The goal of the present study is to investigate to which extent participants' rate of forgetting varies over the course of three weeks as well as across four different types of material.

Methods

The Model

The model used in this experiment is based on ACT-R's declarative memory equations (Anderson, 2007). In the ACT-R framework, each item that is learned is assigned an *activation* value. Activation is highest at the moment an item is encountered and then decays as a function of time. The activation of an item at any point in time can be computed using the following equation:

$$A_i(t) = \sum_{j=1}^{n} (t - t_j)^{-d_j} \qquad \text{Eq. 1}$$

According to this equation, the activation of item *i* at time point *t* depends on all previous time points at which item *i* has been encountered. After each previous encounter *j* the activation associated with that encounter decays with d_j, which translates to a smaller contribution to the current activation if encounter *j* has occurred long before time point *t*. The rate with which the activation decays after each encounter is calculated as follows:

$$d_{ji} = ce^{A_i(t_j)} + \alpha_i \qquad \text{Eq. 2}$$

In this equation, *c* is the decay scale parameter that determines the relative contribution of the activation component. Alpha represents the decay intercept, which represents a minimum decay value (and will be used as the decay value for the first encounter). This equation has been developed by Pavlik and Anderson (2008) to deal with the spacing effect. In the ACT-R framework, an activation value can be directly converted to an estimated response time by scaling the activation and adding a *fixed time* that accounts for non-memory related processes. The following equation is used to convert the activation of item *i* at time point *t* to an estimated reaction time:

$$RT_i(t) = Fe^{-A_i(t)} + \text{fixed time} \qquad \text{Eq. 3}$$

Pavlik and Anderson (2003, 2005, 2008) have shown that the three equations outlined here can be used to fit a wide range of data from learning-related experiments and can account for additional benefits gained through the spacing effect. The system has not only been used to describe collected data but also to devise a system that predicts, in real-time, the order in which items should be repeated to yield optimal retention. More recently, Van Rijn and colleagues (2009) and Nijboer (2011) have developed the system further and showed that a scheduling algorithm that compares observed with predicted reaction times (derived from an item's estimated activation) leads to even better learning than the Pavlik and Anderson (2008) model. The same algorithm is used in this study and a graphical representation of the procedure is depicted in Figure 1.

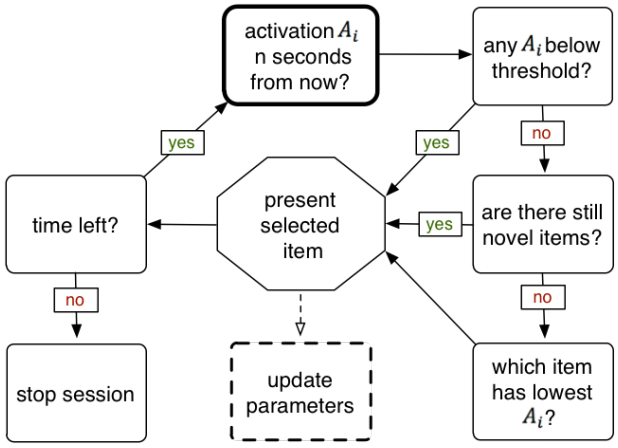


Figure 1: A graphical representation of how the model determines the order in which to repeat old and present new items.

The algorithm selects the order in which items are presented to the learner dynamically and adjusts the order of repetitions based on the learner's behavior. This is done as follows: The model simulates the activation of all items that have already been encountered *n* seconds from now using

the equations described above. If *n* seconds from now, the activation of any item is below the retrieval threshold, that item will be presented next (because this indicates that the item is about to be). If no item is below the threshold, a new item is presented as long as novel items are still available. Otherwise, the item with the lowest activation *n* seconds from now is presented. At the time the selected item is presented, the model uses the estimated activation of the item in the learner's memory to compute the estimated reaction time (see Eq. 3). The item's alpha parameter is updated by comparing the estimated reaction time with the observed reaction time. If the estimated reaction time was too slow, this indicates that the estimated activation was too low. That, in turn, indicates that the decay value for the previous encounter was estimated to be too high. To compensate for this discrepancy, the alpha parameter for the given item is adjusted in a step-wise procedure to improve the model's estimate on the following trial (see Nijboer (2011) for details). After the parameter has been updated, the model checks whether the learning session should continue and then either stops or starts the next repetition. The mechanism is depicted graphically in Figure 1.

Procedure

Each person participated in the study for three sessions on three days, each session spaced one week apart. Within each session, there were two blocks. Each block was made up of a 20-minute study session, a five-minute distraction task, and a test of the studied material that took about five more minutes. At the beginning of the first session, each participant also completed a short questionnaire regarding demographic information (age, gender, nationality, and language skills). The five-minute distraction was a simple variation of the puzzle game Tetris which participants played until they were automatically re-directed to the test that concluded each block.

During a study block, novel items were presented on *study trials* and subsequent repetitions were presented on *test trials*. On a *study trial*, participants saw both the cue and the correct response and had to type in the correct response to proceed. On a *test trial*, participants only saw the cue and had to type in the correct response. Feedback was provided in both trial types and lasted 0.6 and 4 seconds for correct and incorrect answers, respectively. The feedback always resembled a *study trial* and displayed both the cue and the correct response. Jang, Wixted, Pecher, Zeelenberg, & Huber (2012) have shown that for non-retrievable items, an additional *study trial* is very effective because participants do not benefit from the testing effect (but unsuccessful retrieval attempts can still enhance learning (see Kornell, Hays, & Bjork, 2009). Furthermore, they showed that four-second *study trials* yield the highest benefit. During the test at the end of each block, participants were provided with a list of all possible items and could provide their responses in any order they preferred.

Material

For each block, a list of 25 items was compiled. The lists of items were identical for all participants but during each study block, the model randomized the order in which items were presented based on their participant numbers. There were four types of material that were studied by each participant:

Vocabulary. There were 75 Swahili-English word pairs that were taken from Van den Broek, Segers, Takashima, & Verhoeven (2014). Swahili-English word pairs are common stimuli in vocabulary learning (e.g. Carpenter, Pashler, Wixted, & Vul, 2008; Pyc & Rawson, 2010; Van den Broek et al., 2014) because most university students do not have any prior knowledge.

Flags. A list of 25 items was compiled from Wikipedia's list of sovereign states. The authors strived to pick the flags of countries that were not likely to be known by the participants, using their own familiarity with the countries' flags and a pilot study as a benchmark.

City Locations. A list of 25 items was compiled by searching for smallish cities on Google Maps, making sure the cities are more or less evenly spaced across the continental United States of America. Cities were picked so their names are unique, not too difficult to spell, and do not contain information about their geographical location.

Bio-Psychology Facts. A list of 25 bio-psychology facts was compiled from the Glossary in Kalat (2012). The facts were chosen so that the answer would always be a single word and that there is some variations in how difficult the words are to spell.

Participants

Of the 76 first-year psychology students from the participant pool of the University of Groningen that signed up for this study, 71 completed all three sessions and 70 fulfilled the minimum requirement of having seen at least 10 unique items in each study block. It was assumed that if the participant was not able to perform well enough to be presented with at least 10 unique items within a 20-minute study block, there would likely be a problem that was beyond them being a poor learner so their data was dismissed. Participants were also removed when they answered less than 25% of the items they had seen during the study block correctly on the subsequent test, which applied to 3 participants. Of the remaining 67 participants, 50 were female and the median age was 20 (SD_{age} = 1.73; $range_{age}$ = [17; 26]). No one indicated familiarity with Swahili, 35.8% were Dutch, and 52.2% were German. All participants indicated to be fluent in English and gave informed consent.

Results

Figure 2 summarizes the performance on the final test that concluded each block. The black bars in the plot are traditional box plots and the white dots highlight the group's median. The colored areas are scaled density plots that

depict the distribution of each group's values. The data suggest that performance was very high overall. Especially in the three vocabulary sessions, performance was very high for most participants with a little more variation in the other blocks. The city location block (CI) seems to have been the most difficult followed by the bio-psychology fact block (BIO). The overall excellent performance suggests that participants actively engaged with the material during the study session, which makes us confident that the alpha parameters that were obtained during the study sessions contain meaningful information about the participants' engagement with and acquisition of the studied material.

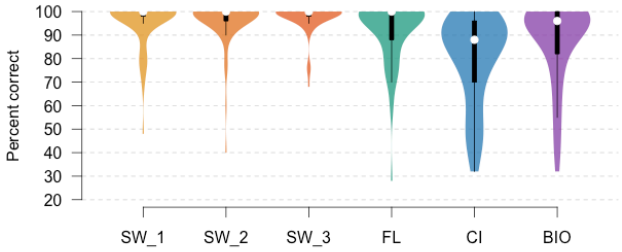


Figure 2: Performance on the final test across the six blocks. SW_1 and FL were tested in the first session, SW_2 and CI in the second session, and SW_3 and BIO in the third session. The sessions were spaced one week apart.

As described in the sub-section The Model, each item that each participant studied is assigned one alpha value. Each item starts with an alpha value of 0.3 but then the alpha value is adjusted on each repetition of trial, depending on how the participant responds to the item and how well that response matches up with the model's prediction. To address the research question of whether the estimated rate of forgetting is stable (1) over time and (2) over materials, we looked at the variation in alpha values across time and materials. For this analysis, the alpha values of items that have been presented at least three times have been included. That is, each participant contributed multiple alpha values and the exact number depended on how many items that participant encountered at least three times within each block.

For the analysis, the alpha values were log-transformed to satisfy assumptions of homoscedasticity and normality. The aim of the analysis was to check whether alpha scores differed across time and materials and whether both factors influenced each other in their effect on the alpha values. To test this, we used linear mixed-effects model regression with dummy coding. The mixed-effects model allows accounting for the interdependency between observations due to by-subject and by-item variation. Three variables were included in the model to test our research question: The first variable coded the *session* (that is, the day) on which the blocks were completed. This allowed us to check whether there is any significant variation over time across all blocks. The second variable was coded 0 for blocks in which participants studied Swahili words and 1 for those in which non-Swahili material was studied. This allowed us to directly compare the differences between multiple blocks of learning Swahili to non-Swahili blocks. The third dummy was coded -0.5 for the flags block (FL), 0.5 for the city location block (CI), and 0 for all other blocks. This allows us to compare the individual blocks (that is, types of material) in more detail. The results of the analysis are shown in Table 1.

Table 1: Results of the linear mixed-effects regression.

	beta	SE	df	| t |	p
intercept	-1.394	0.040	112	34.38	<0.001
session	-0.036	0.028	73	1.30	0.198
SW vs. ¬SW	0.182	0.011	9506	16.50	<0.001
FL vs. CI	0.364	0.013	9457	27.87	<0.001
session * SW v. ¬SW	-0.051	0.028	82	1.81	0.074

The alpha scores do not significantly differ between sessions ($t(73)=1.3$, $p=0.198$). However, the contrasts between the Swahili and non-Swahili blocks and between the flags and city location blocks significantly influence the alpha values ($t(9506)=16.5$, $p<0.001$; $t(9457)=27.87$, $p<0.001$, respectively). More specifically, participants had smaller alpha values in the Swahili blocks compared to the flag and city location blocks (a decrease of 0.049) indicating a faster forgetting rate for the latter two blocks. This effect was stronger for the city location block, which is suggested by the positive coefficient of the flags vs. city locations contrast. Specifically, the forgetting rate increases by 0.109 in the city compared to the flags block. The interaction between the sessions and the contrast Swahili vs. non-Swahili is not significant ($t(82)=1.81$, $p=0.074$). In other words, the increase of the forgetting rate in performing the flag or city task compared to the Swahili task is independent of when (that is, in which session) one performs the task.

While the regression analysis examined overall effects of difference in alpha values, it might also be informative to take a closer look at the development of estimated alpha values throughout the course of a study session. Figure 3 shows how the alpha values for each item change as a function of time. The items are color-coded (the legend is shown at the top of the graph) and it can be seen that each item has an alpha value of 0.3 when it is first introduced. On each subsequent repetition, the alpha value is adjusted and the magnitude of the change depends on the discrepancy between the estimated and the observed reaction time. The data shown in Figure 3 come from a very good participant so that many items end up with an alpha value lower than the default they started with. It can be seen, however, that there are substantial differences in alpha values within this participant, indicating that some items were more difficult to learn than others. The peak and frequent rehearsal of the 25th item is particularly obvious. This pattern was likely caused by a series of incorrect responses, which led the model to believe that the item was not learned yet, in response to which the alpha was corrected upwards step-by-step. The higher alpha than resulted in a more frequent rehearsal (see Figure 1). The plot also makes clear that the

model does a good job of interleaving items that were learned early in the session with those learned later on.

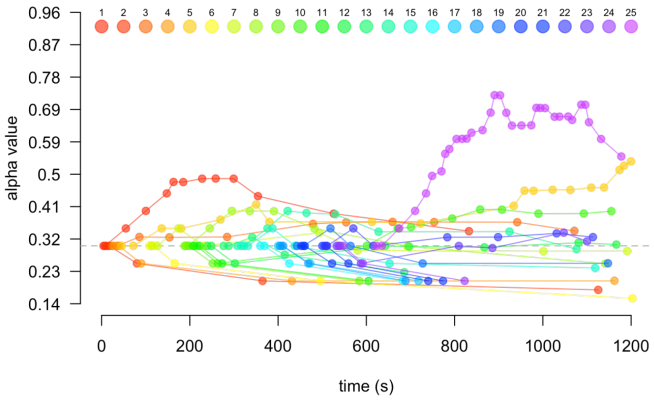


Figure 3: Development of the alpha values for each item for one participant in one block as a function of time.

Discussion

In this study, we investigated the stability of individual rate of forgetting parameters in a model of optimal fact learning. The emphasis is on scrutinizing the stability of the parameter values across time and across different materials. Knowing more about the circumstances under which a learner's estimated rate of forgetting is stable in time and across materials will enable us to further develop the model by carrying over what we learned about the participant in one learning session to the next.

The results of the analysis demonstrate that the estimated rates of forgetting do not differ significantly over time. There is a difference in estimated rates of forgetting, however, when different types of material are studied. Given the non-significant interaction between time point of study and type of material, differences between materials seem to be independent of time.

When looking at the data of the performance on the final test depicted in Figure 2, one can see that there was a clear ceiling effect. The effect is especially pronounced in the three Swahili blocks and the block in which participants learned flags. This might be considered to be an issue because it would facilitate the stability of results within those Swahili-learning blocks. It should be noted, however, that by using the parameter values that were estimated throughout the learning session instead of the *results* of the learning session (test performance), one gets a much more fine-grained view on the differences between conditions. There is much more variation in estimated rates of forgetting than the corresponding results on the test suggest. This conclusion is further supported by the fact that there was no significant difference across the three sessions (see Table 1) even though the comparison *did* include the blocks for which final performance was not at ceiling. In addition to that, using the estimated rates of forgetting for each item from each block can also serve as a diagnostic tool to get a better idea of the inner mechanics of the model and detect ways in which the model might not perform optimally and why. By plotting the development of the parameters over time for a single participant in one of the six blocks (see Figure 3) can indicate problems that would not be apparent from measures taken at the end of a learning session. Therefore, we think an analysis based on the estimated parameter values is much more interesting and insightful than one based on the performance on the final test.

As discussed in the Introduction, the model does not currently preserve estimated parameter values across multiple study sessions. That is, when a learner uses the model to study a number of Swahili-English word-pairs and then returns to the system the day after and starts another study session, the model will revert back to the default parameter values at the beginning of the second session. This seems both wasteful and inefficient. One would think that by observing the learner's behavior in the first session and comparing it to the model's estimates (which are based on the current parameter values), we have learned something about that particular learner. And updating the internal parameters of the model dynamically to capture this learning-about-the-learner is an essential part of the model. Therefore, it would be a logical next step to determine a way in which we could preserve what we have learned about the learner in the first session. That way, we can give the model a head start at the beginning of the second session instead of forcing the model to start from scratch.

The data reported here show that there is substantial stability of parameter values over time, especially if the same type of material is studied: Swahili vocabulary. We reckon it is reasonable to assume that these findings generalize to languages other than Swahili. It would be interesting, however, to test whether a transfer from Swahili to, for example, French is better than the transfer from Swahili to bio-psychology. A challenge for the future will be to determine the optimal transfer of parameter values between sessions that do not deal with the same type of material. In this study, we made an effort to devise material that is very different from each other (word-pairs (Swahili), visual information (flags), topographical information (city locations), and factual knowledge (bio-psychology)) but this leaves open the question of how similar material has to be to still allow smooth transfer of suitable parameter values.

Conclusion

The data presented here suggest that participants' rate of forgetting varies between materials but is relatively stable within a domain over time. This indicates that rate of forgetting is not purely a feature of a learner's memory system but also influenced by the type of material studied. If the same material is studied, though, the data suggest that the rate of forgetting is stable over time. Therefore, we should be able to improve the model further by carrying over what we learned about a learner from one session to the next, given that the sessions deal with the same type of material.

References

Agarwal, P. K., Karpicke, J. D., Kang, S. H. K., Roediger, H. L., & McDermott, K. B. (2008). Examining the testing effect with open- and closed-book tests. *Applied Cognitive Psychology*, *22*, 861–876.

Anderson, J. R. (2007). *How can the human mind occur in the physical universe?*. New York: Oxford University Press.

Anderson, J. R., & Milson, R. (1989). Human memory: An adaptive perspective. *Psychological Review*, *96*(4), 703–719.

Anderson, J. R., & Schooler, L. J. (1991). Reflections of the environment in memory. *Psychological Science*, *2*(6), 396–408. doi:10.1111/j.1467-9280.1991.tb00174.x

Butler, A. C., & Roediger, H. L. (2007). Testing improves long-term retention in a simulated classroom setting. *European Journal of Cognitive Psychology*, *19*(4-5), 514–527. doi:10.1080/09541440701326097

Carpenter, S. K., Pashler, H., Wixted, J. T., & Vul, E. (2008). The effects of tests on learning and forgetting. *Memory & Cognition*, *36*(2), 438–448. doi:10.3758/MC.36.2.438

Carrier, M., & Pashler, H. (1992). The influence of retrieval on retention. *Memory & Cognition*, *20*(6), 633–42.

Cepeda, N. J., Pashler, H., Vul, E., Wixted, J. T., & Rohrer, D. (2006). Distributed practice in verbal recall tasks: A review and quantitative synthesis. *Psychological Bulletin*, *132*(3), 354–80. doi:10.1037/0033-2909.132.3.354

Cepeda, N. J., Vul, E., Rohrer, D., Wixted, J. T., & Pashler, H. (2008). Spacing effects in learning: a temporal ridgeline of optimal retention. *Psychological Science*, *19*(11), 1095–102. doi:10.1111/j.1467-9280.2008.02209.x

Dempster, F. N. (1988). The spacing effect: A case study in the failure to apply the results of psychological research. *American Psychologist*, *43*(8), 627–634.

Donovan, J. J., & Radosevich, D. J. (1999). A meta-analytic review of the distribution of practice effect: Now you see it, now you don't. *Journal of Applied Psychology*, *84*(5), 795–805. doi:10.1037//0021-9010.84.5.795

Ebbinghaus, H. (2013). Memory: A contribution to experimental psychology. *Annals of Neurosciences*, *20*(4), 155–156. doi:10.5214/ans.0972.7531.200408

Jang, Y., Wixted, J. T., Pecher, D., Zeelenberg, R., & Huber, D. E. (2012). Decomposing the interaction between retention interval and study/test practice: the role of retrievability. *Quarterly Journal of Experimental Psychology*, *65*(5), 962–75. doi:10.1080/17470218.2011.638079

Jastrzembski, T., Gluck, K., & Gunzelmann, G. (2006). An ACT-R predictive model of performance: Applications in education and training. In *Proceedings of the Society for Mathematical Psychology Annual Meeting*. Vancouver, Canada.

Kalat, J. W. (2012). *Biological psychology* (11th ed.). Wadsworth, Cengage Learning.

Karpicke, J. D., & Roediger, H. L. (2008). The critical importance of retrieval for learning. *Science*, *319*(5865), 966–8. doi:10.1126/science.1152408

Kornell, N., Hays, M. J., & Bjork, R. A. (2009). Unsuccessful retrieval attempts enhance subsequent learning. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *35*(4), 989–98. Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/19586265

Lindsey, R. V, Shroyer, J. D., Pashler, H., & Mozer, M. C. (2014). Improving students' long-term knowledge retention through personalized review. *Psychological Science*, *25*(3), 639–47.

McDaniel, M. A., Anderson, J. L., Derbish, M. H., & Morrisette, N. (2007). Testing the testing effect in the classroom. *European Journal of Cognitive Psychology*, *19*(4-5), 494–513.

Nijboer, M. (2011). *Optimal fact learning: Applying presentation scheduling to realistic conditions*. University of Groningen.

Pavlik, P. I., & Anderson, J. R. (2003). An ACT-R model of the spacing effect. In *Proceedings of the 5th International Conference on Cognitive Modeling* (pp. 177–182). Bamberg.

Pavlik, P. I., & Anderson, J. R. (2005). Practice and forgetting effects on vocabulary memory: An activation-based model of the spacing effect. *Cognitive Science*, *29*(4), 559–86. doi:10.1207/s15516709cog0000_14

Pavlik, P. I., & Anderson, J. R. (2008). Using a model to compute the optimal schedule of practice. *Journal of Experimental Psychology: Applied*, *14*(2), 101–17. doi:10.1037/1076-898X.14.2.101

Pavlik, P. I., Bolster, T., Wu, S., Koedinger, K. R., & MacWhinney, B. (2008). Using optimally selected drill practice to train basic facts. In B. Woolf, E. Aimer, & R. Nkambou (Eds.), *Proceedings of the 9th International Conference on Intelligent Tutoring Systems* (pp. 593–602). Montreal, Canada.

Pyc, M. A., & Rawson, K. A. (2010). Why testing improves memory: Mediator effectiveness hypothesis. *Science*, *330*, 335.

Roediger, H. L., & Butler, A. C. (2011). The critical role of retrieval practice in long-term retention. *Trends in Cognitive Sciences*, *15*(1), 20–7.

Van den Broek, G. S. E., Segers, E., Takashima, A., & Verhoeven, L. (2014). Do testing effects change over time? Insights from immediate and delayed retrieval speed. *Memory*, *22*(7), 803–12.

Van Rijn, H., van Maanen, L., & van Woudenberg, M. (2009). Passing the test: Improving learning gains by balancing spacing and testing effects. In *Proceedings of the 9th International Conference on Cognitive Modeling* (pp. 110–115).

Spontaneous Retrieval for Prospective Memory: Effects of Encoding Specificity and Retention Interval

Justin Li and **John Laird**
University of Michigan
2260 Hayward Street
Ann Arbor, MI 48109-2121
{justinnh, laird}@umich.edu

Abstract

This paper explores the role of spontaneous retrieval in prospective memory, in an agent implemented in the Soar cognitive architecture. At goal initiation time, spreading activation causes the goal to be the most activated element in long-term memory, at which point it is spontaneously retrieved into working memory and pursued. We show that goal encoding specificity increases prospective memory performance, while a lengthier retention interval decreases performance if the percepts are differentially presented; both trends qualitatively resemble results described in psychology literature. However, a large space of possible spontaneous retrieval implementations remain unexplored, and much work remains to be done before spontaneous retrieval in a cognitive architecture can be fully understood.

Keywords: spontaneous retrieval; prospective memory; encoding specificity, cognitive architecture; Soar

Introduction

Prospective memory is the ability to remember to do something in the future, often while other activities are being performed during the delay. Such tasks are common in everyday life: from passing a message to a colleague, to taking medication before bed, these tasks all require the subject to perform particular actions (giving the message, swallowing a pill) under particular conditions (when the colleague is in sight, at bedtime). One of the main research questions in prospective memory is how people recall that they have an action to perform when the goal conditions are met. Two general classes of strategies have been suggested: monitoring, where someone deliberately checks if the conditions are satisfied, and spontaneous retrieval, where the need to act somehow "pops" into mind. Cognitive architectures are well-suited to create models of monitoring, since agents created in that framework have fine deliberate control over the use of memory. The same, however, cannot be said of spontaneous retrieval strategies, as they require automatic memory mechanisms, which have thus far received little attention in cognitive architectures.

This paper presents a preliminary exploration of the use of spontaneous retrieval for prospective memory. We implemented an automatic, uncued, activation-based retrieval mechanism in the Soar cognitive architecture, and demonstrate that the mechanism provides agents with a robust prospective memory ability. In an abstract domain that presents agents with randomly-generated goal conditions, the use of spontaneous retrieval allows the agent to achieve its prospective goals across a wide range of environmental and agent parameters. Furthermore, the performance of the agent changes with encoding specificity and retention interval length in ways that qualitatively resemble those of people. This serves as one step in building a complete model of how people perform prospective memory tasks, and the factors that must be taken into consideration when selecting between strategies.

Background

Prospective Memory

Although prospective memory has gotten increasing attention from psychologists in the last twenty years, the capability is only defined as a "fuzzy set" of intuitions around "remembering to *do something* at a particular *moment (or time period) in the future*" (emphasis in original) (McDaniel & Einstein, 2007). For clarity, we define a prospective memory task as represented by the *target* — the conditions under which the goal is applicable — and the *action*, which the agent must take to achieve the goal. Within this framework, previous literature has identified the five stages of completing a prospective memory task (Ellis, 1996). To use message-passing as an example, the stages are:

Encoding The goal is created and stored in long-term memory; this occurs when the message is given to the agent and asked to be passed on to the colleague.

Retention This stage is the delay between the storage of the goal and when the target conditions are met, such as between when the message was received and when the colleague is seen.

Initiation The target conditions of the goal are fulfilled, and the goal must be retrieved from long-term memory to working memory. In the example, the colleague is in sight.

Execution The action of the goal is taken; in this case, the colleague is given the message.

Completion Long-term memory must be changed such that the goal will not be repeated; that is, when the colleague is next seen, another attempt to pass on the message would not be made.

The crux of prospective memory is during the initiation stage, which hides a knowledge dependency problem (Li & Laird, 2013a). Since people cannot directly act on knowledge in long-term memory, the goal must be retrieved for the sight of the colleague to be considered significant; at the same

time, since retrieval from long-term memory can only be done deliberately, the goal is not retrieved without a recognition of significance in the first place.

The psychology literature identifies two classes of human strategies to avoid this dependency problem. The first is *monitoring*, which is characterized by the continual expenditure of attentional resources (Smith & Bayen, 2004); this can be modeled by the agent periodically retrieving and checking the relevance of goals that are related to its current situation, what is called a *preemptive strategy* in prior work (Li & Laird, 2013b). This class of strategies breaks the dependency cycle by retrieving goals without determining their relevance. The second category of strategies breaks the cycle the other way, by removing deliberation in the retrieval of goals; these require the long-term memory system to signal the agent in some way. One possibility is to signal that there is a relevant goal to prompt a deliberate memory search (what is called a *noticing-plus-search* strategy), while an alternative is to have the knowledge of the goal be spontaneously retrieved into working memory (a true *spontaneous retrieval* strategy).

These two classes of strategies are not mutually exclusive, but complimentary. Although early work on prospective memory debated whether monitoring or spontaneous retrieval is the better description of human behavior, recent work has shifted towards determining the factors that influence which strategy is used for a particular prospective memory task (Einstein et al., 2005). Monitoring strategies are preferred when the goal conditions are non-focal or when the goal is important, while spontaneous strategies are preferred when the delay is long, when working memory resources are low, or when the interim task is cognitively demanding (McDaniel & Einstein, 2007). This suggests that spontaneous retrieval is more effective than monitoring at completing prospective tasks when these properties are present in the environment and the goal. Both strategies are also affected by the encoding of the goal which, following the encoding specificity principle, must match the percepts at the time of initiation (Einstein & McDaniel, 2010).

Although computational models of prospective memory have been built, they tend to sidestep initiation and focus on retrieving the correct goal from long-term memory. One model explored the Intention Superiority Effect (ISE), which states that unachieved goals are retrieved more quickly than achieved goals (Lebiere & Lee, 2002). Another model looked at different accounts of finding the correct goal, and correlates the timing results to human data (Elio, 2006). Crucially, both models assume that the agent knows that a goal must be retrieved, while the difficulty of the initiation stage lies in how that fact is recognized. Since both models require deliberate use of memory, they more closely resemble monitoring strategies. Modeling spontaneous retrievals would require an automatic memory mechanism, which is discussed below.

Spontaneous Retrieval

Spontaneous retrieval from long-term memory has been acknowledged since the first studies of memory (Ebbinghaus, 1913). In contrast to deliberate or voluntary retrieval, which requires executive functions for search control, spontaneous retrieval is an associative process that requires little to no cognitive effort, often resulting in memories that overlap in features with the current situation (Berntsen, 2010). There is often a distinction between retrieval from semantic memory (i.e., retrievals of facts) and retrievals from episodic memory (i.e., retrievals of experiences) (Kvavilashvili & Mandler, 2004; Berntsen, 2008); although it is possible for prospective memory to use either mechanism, here we focus on retrievals from semantic memory.

Computationally, designing a spontaneous retrieval mechanism requires answering two questions: When is a memory retrieved? And which memory is retrieved? The answers to these two questions define a space of spontaneous retrieval mechanisms, a more thorough exploration of which can be found elsewhere (Li & Laird, 2015). Here we only note that there are additional constraints on the second question, namely, that the retrieved memory should be relevant to the current situation. In most cognitive architectures, retrieval from long-term memory requires a description of the features of the desired memory element. This description ensures that the retrieved element can be used for further reasoning. With spontaneous retrieval, however, the agent cannot deliberately create this description; a different mechanism for ensuring relevance must be used. One solution is to use a spreading activation mechanism, such that the knowledge in working memory influences which long-term memory elements are highly activated and are thus more likely to be retrieved. Again, the full space of spreading activation mechanisms is beyond the scope of this paper.

Implementation in Soar

Soar (Laird, 2012) represents all declarative knowledge as edge-labeled directed graphs. Knowledge in *working memory* is matched by *procedural if-then rules*, which in turn modify working memory. In addition to buffers that represent the perceptual input and motor output of the agent, working memory also contains buffers that allow agents to access long-term memory. In particular, a Soar agent can *store* a single element (a graph node plus all its outgoing edges) into *semantic memory*. Before the current work, the only mechanism for *retrieval* from semantic memory was for the agent to (deliberately) create a *cue* — a set of features of the desired memory element. Semantic memory then finds all elements that contains the entire set of features, and then places the element with the highest *activation* into working memory. This semantic memory element activation is *boosted* when the element is stored or is the result of a retrieval (similar to ACT-R), and *decays* over time as controlled by a *decay rate* parameter.

Spontaneous retrieval extends the capabilities of Soar's semantic memory. From the perspective of the agent's interface to memory, the biggest change is that whenever there is no deliberate retrieval, semantic memory automatically selects

an element to be placed into the semantic memory buffer. As with the bias from deliberate cued retrieval, semantic memory selects the most highly activated element, with the caveat that it skips over any element that is already in working memory; this ensures that spontaneous retrieval is not attempting to retrieve knowledge that the agent has already retrieved.

To ensure that the spontaneously retrieved memory is relevant, spreading activation is used to boost the activation of elements related to the contents of working memory. Our implementation of spreading activation is different from the spreading activation in ACT-R (Anderson, 2007). In the latter, spreading is only a term in determining the bias for retrieval, but has no long-term effect on a long-term memory element's base-level activation. This is undesirable, since this means spreading is ahistoric — spreading makes an element more likely to be retrieved at this current time step, but has no effect on whether it is likely to be retrieved at the next time step.

Instead, we created a spreading activation mechanism that directly changes base-level activation. The mechanism is defined by two parameters: what triggers the initial activation boost occurs, and which elements are affected by the spread. This strictly subsumes the original activation mechanism of Soar, which can be cast as a "spreading" mechanism in which the initial boost is triggered by long-term memory storage and retrieval, and in which no other element is affected. Additionally, we added a third trigger: whenever a rule puts a long-term memory element into working memory, that element also receives an activation boost. In effect, this allows input percepts to cause changes in long-term memory activation. Whereas before only the element itself is boosted, now all graph-neighbors of that element, and the neighbors of those elements and so on, also receive a boost in activation, to some parameterized depth d. Note that these boosts are identical, as though those elements were themselves retrieved into working memory. This is not ideal — a boost which decreases with distance may be more intuitive — but it serves as an initial attempt at a useful spontaneous retrieval mechanism.

Given this mechanism, a Soar agent that uses spontaneous retrieval for prospective memory works as follows:

Encoding The goal and its targets and actions are stored into semantic memory. The goal receives a boost in activation as a result, but this has little impact on its retrieval later.

Retention The goal is forgotten from working memory. The semantic memory activation of the goal also decays, but remains retrievable. When external percepts coincidentally (partially) overlap with the target, the goal will receive a boost in activation due to spreading, but in general, the activation of the goal is low, and is not spontaneously retrieved. Even if the goal is spontaneously retrieved, the agent will discover that the goal conditions are not met, and the goal is ignored.

Initiation At initiation time, all the goal target conditions are matched. Each individual condition causes a boost in the activation of the goal; together, these significantly increase the goal's activation, causing it to be spontaneously retrieved. A rule then matches the retrieved goal, which verifies that the conditions of the goals are satisfied. The agent can then choose to pursue the goal.

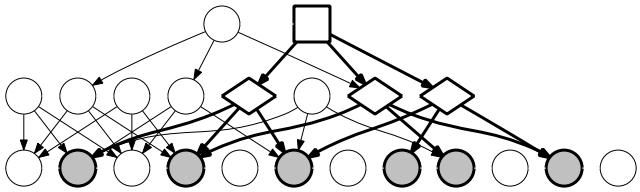


Figure 1: An example random knowledge hierarchy. See text for description.

Execution The action of the goal is then performed.

Completion Finally, once the goal is fulfilled, the agent removes the goal from semantic memory. The goal will never be spontaneously retrieved, and the agent never pursues that instance of the goal again.

Crucially, this strategy requires the goal to be within spreading distance of the agent's perceptions. This may not always be the case — the conditions of the goal may be described in abstract terms that do not directly correspond to perception. There are several ways to prevent this from occurring. One possibility is to increase the depth limit of spreading activation; due to the branching factor of semantic memory, however, this is exponentially costly. Another possibility is for the agent to encode the goal such that the conditions more closely match perceptual input; in other words, to increase encoding specificity. Finally, the agent may also have additional rules that elaborate on perceptual input (*elaboration rules*), building up to to the goal conditions; in essence, this is generalizing the percepts, and can be thought of as the flip side of encoding specificity. As we demonstrate below, these three factors are not independent, and together they determine whether the goal is boosted and retrieved.

Prospective Memory Domain

In order to evaluate the use of spontaneous retrieval for prospective memory, a domain was created that represents prospective memory tasks in the abstract.

To simplify our analysis, we restrict our work to where the structure of knowledge in long-term memory forms a recognition hierarchy, or equivalently an ontology with only `has-a` relations. For example, the agent may recognize that an object with four legs and a back is a chair, and that because there are multiple chairs and multiple tables, that the location is a classroom. For this domain, we randomly generate such hierarchies from the bottom up, where the creation of each lower-level feature has a probability of resulting in a feature one level up. This process continues until a specified number of features at a specified height is created; for example, Figure 1 shows a hierarchy of width 2 and height 3. Note that the

hierarchy is not a connected graph: lower-level features may not be part of any higher-level feature.

Within the prospective memory domain, the knowledge hierarchy determines both the target conditions of goals and the percepts of the agent. The only input the agent perceives is from the lowest level of the hierarchy, while a subset of higher nodes are designated as goals. For example, in Figure 1, if the square node is a goal, then the agent should perform the goal action when all the percept-level descendants of that goal are perceived (shaded). To generate a particular trial for the agent, the percepts for goals are first inserted into the percept sequence, with the remaining percepts interpolated using a noisy random walk.

Before each trial, the knowledge hierarchy is inserted into the semantic memory of the agent, but *without* knowledge of which features are the goals. At each time step, the agent is presented with features from the lowest level of the hierarchy, on which elaboration rules would match to create higher-level features. In addition to percepts, the agent is also presented with goals and their features (if the goal is the square node in Figure 1, its features are the diamond-shaped nodes). It is up to the agent to store the goal into memory, where it may also encode the goal more specifically by expanding the intermediate-level features into percept-level features (for example, linking the goal with the shaded nodes instead).

Within this domain, we are interested in the proportion of prospective memory tasks completed by an agent using a spontaneous retrieval strategy. We are interested in several environmental and agent parameters:

- (*) The specificity of encoding by the agent.
- The maximum spreading depth in semantic memory.
- The highest level of perceptual elaboration.
- (*) The length of the retention stage.
- The average number of conditions in a goal.
- The decay rate of semantic memory.

The parameters marked with asterisks are known to have an effect on human prospective memory. For encoding specificity, it is expected that prospective memory performance increases when the goal encoding matches that of percepts. As for the length of the retention stage, the longer the interval, the more likely that a spontaneous strategy is chosen. It is assumed that this is due to the increased cost of monitoring for long periods, but that does not preclude the possibility of spontaneous retrieval performance also changing as a function of this parameter.

Results

In general, spontaneous retrieval provides a robust prospective memory ability, allowing an agent to complete an average of 81.5% (and a median of 90%) of its goals across a range of parameter settings. We examine the effects of encoding specificity and retention interval length below.

Encoding Specificity

Encoding specificity, in this case, refers to the target conditions of the goal that is stored in long-term memory. Instead of directly storing the features of the goal, the agent instead stores the goal with its lower-level features; in Figure 1, this means the goal (the square node) is stored with the shaded nodes as its conditions instead of the diamond-shaped nodes. This encoding means the goal is now connected to the knowledge hierarchy at a lower level than it would be otherwise. Since the goal is often at the top of the knowledge hierarchy, we denote the specificity of an encoding by how many levels below the goal it is linked to; in this example, the encoding specificity would be 2.

Given this definition of encoding specificity, we perform initial analysis to determine whether a goal could be spontaneously retrieved. For goal at knowledge level g, elaboration rules that create features up to level e, and a spreading depth of d, the goal must be encoded at specificity level s that satisfies the following relationship:

$$d \geq g - e - s + 1 \tag{1}$$

That is, the spreading depth must be able to reach from the highest-level elaborated features to goal conditions (plus an extra level to spread from the conditions to the goal itself). Note that the agent cannot complete any goals when $d = 0$, since the goal would never receive an activation boost from spreading (since no spreading occurs). We can additionally calculate the maximum number of boosts a goal will receive, assuming the knowledge hierarchy has branching factor b:

$$\sum_{i=\max(1,\, s-d+1,\, g-e)}^{\min(g,\, s+d-1)} b^i \tag{2}$$

That is, every feature in the levels indicated by the index would boost the goal, a number which is exponential in the branching factor. This classification allows us to group agents across a large parameter space for comparison. The results here are from exploring $1 \geq g \geq 3$, $0 \geq e \geq 3$, $1 \geq s \geq 3$ and with $d \in \{1,2\}$ and branching factor of 3.

A number of parameter settings within this space fail in completing any goals. Upon closer examination, these are settings where the goal is at least two steps away from the elaborated features — for example, if elaborations provide features of level 3 and the goal conditions are encoded at level 4, thus requiring a two-level spread from elaboration to condition to the goal. Equivalently, this is when $e + s < g$, or where the right hand side of Equation (1) is two or more. In these cases, the lower-level features are activated more frequently, causing them to have higher activation than the goal and preventing the goal from being retrieved. This suggests that the activation boosting of a goal is not sufficient to guarantee its completion.

All other parameter settings allow the agent to complete goals. The parameter that is most correlated with higher performance is the specificity of the encoding: every increase

in specificity results in a higher proportion of goals being completed. The results in the table below are typical; the numbers represent the proportion of goals that the agent completed. In retrospect, this is not surprising: more specifically encoded goals are linked to more features, which means that there are more opportunities for the activation of the goal to be boosted.

Table 1: Representative results demonstrating the effects of encoding specificity. The numbers reported are the proportion of goals completed.

Elaboration	**Encoding Specificity**	
Level	1	2
0	0%	75%
1	70%	80%
2	65%	80%
3	65%	80%

Neither the level of elaboration nor the depth limit for spreading activation have uniform effect on the agent's performance. Although these parameters also effect the number of times a goal is boosted, the problem is that they also boost the activation of all *other* goals in addition to the goal that is being initiated. As with the low-level features from above, it is a high *relative* activation that allows a goal to be spontaneously retrieved. More specific encodings provide a large enough boost at initiation for the single goal to be retrieved, while these other parameters do not.

Overall, these results agree with the psychology literature: the best goal encoding should match both environmental parameters (such as how abstract the goal is) and agent parameters (such as the limit to spreading activation), but that more specific encodings in general lead to better performance.

Retention Interval

In our initial experiment, none of the retention interval length, the decay rate of semantic memory, nor the number of goal conditions had any individual effect; whether a trial lasts 2,000 or 10,000 time steps, or have between 1 and 20 conditions, the agent performs equally well. Learning from the experiments with encoding specificity, however, we suspect that this is due to the "density" of percepts to goals. The features that an agent perceives during the retention interval are randomly selected, and may coincidentally be one of the conditions for a goal; that goal would then receive a small boost in activation. Since all percepts are equally likely, all goals would receive roughly equal numbers of activation boosts, meaning no single goal is particularly highly activated (or particularly un-activated either).

We can frame this idea into one of "resting activation" — activation that a goal would have during the retention interval, which is determined by an equilibrium formed by the increase in activation due to spreading from random input and the decrease in activation due to decay. Changes in either would move the resting activation value; if the decay rate is increased, or if there is less activation from random input (as would be the case if the input did not contain target conditions at all), the resting activation value would decrease. Again, it is not the resting activation that directly determines the performance of the agent, but the relative activation of a goal at initiation time; this is why the decay rate has no effect, since it affects the activation of all goals. Conversely, if a goal has low resting activation compared to other goals, the activation spread from its target conditions may not be sufficient to make it the most activated element, preventing its spontaneous retrieval.

To demonstrate this, we modified the domain such that during the retention stage, the conditions of a single goal are never presented to the agent until initiation. We call this the Leave One Out percept sequence, as opposed to the Normal percept sequence. For that goal, there should be much less activation boosts from spreading as compared to other goals, leading to a lower resting activation level. In this case, a longer retention length (as activation decays after the goal is initially stored) should leave the goal uncompleted.

As expected, the Leave One Out sequence results in much more variance in the activations of goals. The least activated goal in a Normal percept sequence is 1.44 standard deviations away from the mean, while with a Leave One Out sequence, the least activated goal is 12.1 standard deviations away (the standard deviations were calculated without the outlier). This leads to the goal not being retrieved for completion as the retention interval increases, as show in the table below:

Table 2: The effect of different percept sequences. The numbers reported are the proportion of goals completed.

Mean Retention Interval	**Sequence Type**	
(time steps)	Normal	Leave One Out
105.7	96.7%	58.3%
125.0	96.7%	55.0%
142.7	97.5%	50.0%
166.2	96.7%	50.0%
194.1	97.5%	43.8%
214.0	96.7%	38.3%
246.3	97.6%	37.6%

This result could be interpreted in two ways. On one hand, for random percepts, using spontaneous retrieval for prospective memory suffers no degradations in performance, which suggests that it may be preferable to monitoring strategies. On the other hand, goals for which the conditions are never encountered outside of initiation are unlikely to be retrieved under the current mechanism, which run counter to the trends described in psychology literature. We do not know of any studies which look at the baseline frequencies of goal conditions, nor of studies which examine human prospective memory performance where performing the goal require satisfying multiple disjoint conditions. It is possible that human performance exhibit similar patterns under such situations; alternately, a better model may be a hybrid

strategy where occasional monitoring-like retrievals prevent the activation of any goal from dropping too low.

Discussion and Conclusion

This paper presented a strategy for completing prospective memory tasks, by using spontaneous retrieval to bring the goal into working memory at the right time. This strategy proves robust across a number of parameters. In particular, the change in performance over two parameters qualitatively matches human data: the increased performance when the goal is encoded more specifically, and the decreased performance when the retention interval is lengthened (where the conditions of the goal are presented differentially). These results crucially depend on the idea that the goal must have higher *relative* activation compared to other knowledge in order for the prospective memory task to succeed. This explains why other memory parameters have no effect, as they alter the activation of all goals on an absolute scale, but leave relative differences unchanged.

At the same time, a major shortcoming of this work is the unexplored space of both the spreading activation and spontaneous retrieval mechanisms, as well as in the structure of knowledge in memory. These results only hold when spontaneous retrieval is based on activation, when spreading activation has a hard limit on depth, and when long-term memory is a hierarchy. It is easy to imagine alternatives: where spontaneous retrievals are based on analogical mapping, where the size of the activation boost decays over graph distance as it spreads, or where long-term memory is a more complicated graph. None of these parameters can be easily enumerated and tested, and each requires significant evaluation on its own to determine the conditions under which they best match human data or are most useful to artificial agents.

Spontaneous retrieval is an important mechanism for cognitive architectures: it is necessary to fully model human prospective memory, and it also serves as a heuristic for when memory-search guidance knowledge is lacking in artificial agents. While this work is one step in understanding such a mechanism, much work remains to be done, and given that many algorithmic details affect the utility of spontaneous retrieval, these effects may be better explored using simpler models before being implemented in a cognitive architecture.

Acknowledgments

The authors acknowledge the funding support of the Office of Naval Research under grant number N00014-08-1-0099.

References

Anderson, J. R. (2007). *How can the human mind occur in the physical universe?* Oxford University Press.

Berntsen, D. (2008). Involuntary autobiographical memories: Speculations, findings, and an attempt to integrate them. In J. Mace (Ed.), *Involuntary memory* (pp. 20–49). Blackwell Publishing.

Berntsen, D. (2010). The unbidden past: Involuntary autobiographical memories as a basic mode of remembering. *Current Directions in Psychological Science*, *19*(3), 138–142.

Ebbinghaus, H. (1913). *Über das gedächtnis (Memory: A contribution to experimental psychology)* (H. A. Ruger & C. E. Bussenius, Trans.). Columbia University Press.

Einstein, G. O., & McDaniel, M. A. (2010). Prospective memory and what costs do not reveal about retrieval processes: A commentary on Smith, Hunt, McVay, and McConnell. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *36*(4), 1082–1088.

Einstein, G. O., McDaniel, M. A., Thomas, R., Mayfield, S., Shank, H., Morrisette, N., & Breneiser, J. E. (2005). Multiple processes in prospective memory retrieval: Factors determining monitoring versus spontaneous retrieval. *Journal of Experimental Psychology: General*, *134*(3), 327–342.

Elio, R. (2006). On modeling intentions for prospective memory performance. In *Proceedings of the 28th annual conference of the Cognitive Science Society (CogSci)* (pp. 1269–1274).

Ellis, J. (1996). Prospective memory or the realization of delayed intentions: A conceptual framework for research. In M. A. Brandimonte, G. O. Einstein, & M. A. McDaniel (Eds.), *Prospective memory: Theory and applications* (pp. 1–22). Lawrence Erlbaum.

Kvavilashvili, L., & Mandler, G. (2004). Out of one's mind: A study of involuntary semantic memories. *Cognitive Psychology*, *48*(1), 47–94.

Laird, J. E. (2012). *The Soar cognitive architecture*. MIT Press.

Lebiere, C., & Lee, F. J. (2002). Intention superiority effect: A context-switching account. *Cognitive Systems Research*, *3*(1), 57–65.

Li, J., & Laird, J. E. (2013a). The computational problem of prospective memory retrieval. In *Proceedings of the 12th international conference on cognitive modeling (ICCM)* (pp. 155–160).

Li, J., & Laird, J. E. (2013b). Preemptive strategies for overcoming the forgetting of goals. In *Proceedings of the 27th AAAI conference on artificial intelligence (AAAI)* (pp. 1234–1240).

Li, J., & Laird, J. E. (2015). Spontaneous retrieval from long-term memory for a cognitive architecture. In *Proceedings of the 29th AAAI conference on artificial intelligence (AAAI).*

McDaniel, M. A., & Einstein, G. O. (2007). *Prospective memory: An overview and synthesis of an emerging field*. Sage.

Smith, R. E., & Bayen, U. J. (2004). A multinomial model of event-based prospective memory. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *30*(4), 756–777.

Holographic Declarative Memory and the Fan Effect: A Test Case for A New Memory Module for ACT-R

Matthew A. Kelly (matthew.kelly2@carleton.ca)
Kam Kwok (kamhung.kwok@carleton.ca)
Robert L. West (robert.west@carleton.ca)
Institute of Cognitive Science, Carleton University
1125 Colonel By Drive, Ottawa, Ontario, K1S 5B6 Canada

Abstract

We present Holographic Declarative Memory (HDM), a new memory module for ACT-R and alternative to ACT-R's Declarative Memory (DM). ACT-R is a widely used cognitive architecture that models many different aspects of cognition, but is limited by its use of symbols to represent concepts or stimuli. HDM replaces the symbols with holographic vectors. Holographic vectors retain the expressive power of symbols but have a similarity metric, allowing for shades of meaning, fault tolerance, and lossy compression. The purpose of HDM is to enhance ACT-R's ability to learn associations, learn over the long-term, and store large quantities of data. To demonstrate HDM, we fit performance of an ACT-R model that uses HDM to a benchmark memory task, the fan effect. We analyze how HDM produces the fan effect and how HDM relates to the standard DM model of the fan effect.

Keywords: ACT-R; memory; cognitive modeling; cognitive architectures; artificial general intelligence; integrated cognition; holographic reduced representations; vector-symbolic architectures

Introduction

Computational cognitive architectures provide the formal, unified theories of cognition necessary for cognitive scientists to achieve an understanding of the mind. ACT-R (Anderson & Lebiere, 1998) is a widely used cognitive architecture that can model diverse aspects of cognition. As an integrated architecture, ACT-R is a good choice for modelling complex tasks. However, the ACT-R Declarative Memory system (DM) was designed for modelling the results of psychology experiments and as such presents certain limitations for modelling complex, real world behaviour. In what follows, we present Holographic Declarative Memory (HDM), a new module for the ACT-R cognitive architecture that addresses some of DM's limitations. To help establish that HDM can provide the same functionality as ACT-R's DM, we have modelled the fan effect task (Anderson, 1974), analyzed how HDM generates the fan effect, and used this analysis to compare the HDM and DM models.

Holographic Declarative Memory (HDM) replaces ACT-R's symbols with holographic vectors. Holographic vectors retain the expressive power of symbols but have a similarity metric, allowing for shades of meaning, fault tolerance, and lossy compression of stored information.

HDM is based on BEAGLE (Jones & Mewhort, 2007), a learning algorithm that models how people abstract the meaning of words from their lifetime language experience, and DSHM (Rutledge-Taylor, Kelly, West, & Pyke, 2014), a model that uses a similar approach to BEAGLE but repurposes and extends the algorithm as a general memory model. HDM is implemented for Python ACT-R and the code for both Python ACT-R and HDM are available through GitHub[1]. Our intent with HDM is to replicate the basic functionality of DM and provide new capabilities.

First, we provide an introduction to holographic models of memory and the fan effect. Next, we detail Anderson and Reder's (1999) ACT-R model of the fan effect. We then describe HDM and the ACT-R HDM model of the fan effect. We contribute a novel analysis of how holographic models produce the fan effect and relate to Anderson and Reder's model. Finally, we outline future work.

Holographic Models of Memory

First proposed by Longuet-Higgins (1968) and Gabor (1969), a holographic memory is a type of computational associative memory based on the mathematics of holography. Holographic memory has been of interest to cognitive psychologists because of the following:

(i) Associative memories are content-addressable, allowing for memory retrieval without search.

(ii) Holographic memories can compactly store complicated and recursive relations between ideas.

(iii) Holographic memories have "lossy" storage, which is useful for modelling human forgetting.

Cognitive models based on holographic memory can explain and predict a variety of human memory phenomena, such as the serial position curve in free recall (Franklin & Mewhort, 2015). Holographic memory has also been used to model analogical reasoning (Plate, 2000; Eliasmith & Thagard, 2001) and how humans perform simple problem-solving tasks such as playing rocks, paper, scissors (DSHM; Rutledge-Taylor et al., 2014) or solving Raven's progressive matrices (Eliasmith, 2013). Knowledge in SPAUN, the world's largest functional brain model (Eliasmith, 2013), is represented using holographic memory.

ACT-R DM is not designed for modelling tasks that involve large databases, such as language comprehension. Conversely, BEAGLE (Jones & Mewhort, 2007) and

[1] A Python ACT-R distribution with HDM included can be downloaded from <https://github.com/MatthewAKelly/ccmsuite> and the fan effect model, which requires Python ACT-R and HDM, can be downloaded from <https://github.com/MatthewAKelly/faneffect>. A guide to using Python ACT-R can be found at <https://sites.google.com/site/pythonactr/>.

DSHM (Rutledge-Taylor, Vellino, & West, 2008) are holographic models that have been used, respectively, to infer word meanings from a corpus and to infer patterns of movie preferences from a database of user movie scores.

Holographic memory models have also been previously used to model the fan effect. Specifically, Dynamically Structured Holographic Memory (DSHM; Rutledge-Taylor et al., 2014; Rutledge-Taylor, Pyke, West, & Lang, 2010) has been used to model two versions of the fan effect task.

Though HDM is based on DSHM, the HDM module for ACT-R differs sufficiently from DSHM that it is worth demonstrating that HDM can, in fact, model the fan effect task. The differences between HDM and DSHM stem from HDM's integration into ACT-R. As a module for ACT-R, HDM makes commitments as to the cognitive structure that the memory system is situated in. To interface with ACT-R, HDM commits to a particular way of encoding information and to a particular way of calculating reaction times that are distinct from the DSHM model.

Fan Effect

The *fan effect task* (Anderson, 1974) is a recognition memory task. During the study phase of the task, participants memorize a set of sentences that vary on some number of dimensions. In the original fan effect task (Anderson, 1974), each sentence is of the form "the person is in the location" where the person and location vary from sentence to sentence (e.g., "the hippy is in the park").

Once the participants have the sentences memorized, they are given a recognition task. In the recognition task, some sentences are from the study set (*targets*), and some sentences are novel combinations of the people and locations from the study set (*foils*). Participants are instructed to identify as quickly as possible which combinations of person and location were in the study set and which were not.

The *fan* of a concept is the number of different sentences in the study set that contained that concept. For example, if "the hippy is in the park" is the only sentence in the study set that mentions the hippy, then *hippy* has a fan of one. If participants learn that there are four people in the park during the study phase, then *park* has a fan of four.

The *fan effect* refers to the finding that participants are slower to recognize or reject sentences that contain concepts that have a higher *fan.* The more people in the park, the slower participants are to decide if the phrase "hippy is in the park" was in the study set. Likewise, if participants learn that the hippy is in several different locations, they are slower to decide if the hippy was in a particular location.

The fan effect illustrates a fundamental principle of human memory: the availability of a piece of information in memory with respect to a cue is a function of the probability of that piece of information conditional on the cue. If the participants learn four facts about the park, then given the cue *park*, each of those facts have only one chance in four of being the relevant fact to retrieve. The retrieval time from memory will reflect that one in four chance. Conversely, if the participants know only one fact about the park, given the cue *park*, retrieval time will be rapid, reflecting the 100% chance that the fact will be relevant.

ACT-R's Declarative Memory (DM)

In ACT-R, knowledge is represented in Declarative Memory (DM) as lists of *slot:value* pairs called *chunks*. Each slot is a task-relevant dimension of the stimulus, such as "colour" or "location". For example, a red square could be described by the chunk "colour:red shape:square". In the fan effect task, each sentence is represented by a chunk, e.g., "person:hippy place:park". In Python ACT-R, chunks can also be ordered lists of values without slots, "red square" or "hippy park". When the slots are omitted from a chunk, the order of the values in the chunk is used as the organizing principle.

Each chunk in DM has an activation. According to Anderson's (1991) rational analysis, the activation of a chunk in memory is an estimate of the likelihood of the information in the chunk being useful in the current situation. Given a cue that describes the current situation, ACT-R retrieves the chunk in DM with the highest activation. Activation is a sum of a base level activation and a measure of the similarity between the chunk and the cue. Base level activation is a measure of how frequently and how recently the chunk has been used. For a chunk i, the activation of that chunk, A_i, is

$$A_i = B_i + \sum_{j=1}^{n} W_j \, S_{ji} \qquad (1)$$

where B_i is the baseline activation of the chunk, n is the number of slot-value pairs in the cue, W_j is the attention paid to slot-value pair j of the cue, and each S_{ji} is an association strength: a measure of the probability that chunk i is relevant given that the cue contains slot-value pair j.

DM can be understood by analogy to a hydraulic system. Activation is like water and connections between cues and chunks are like pipes. Activation spreads from the cue to the chunks in DM. Chunks with stronger associations to the cue receive more activation. The chunk that receives the most activation is selected and retrieved from memory. The time, T, to retrieve a chunk, i, is a function of the chunk's activation, A_i, and two fitting parameters I and F,

$$T = I + Fe^{-A_i} \qquad (2)$$

The higher the activation, the shorter the retrieval time.

Although ACT-R has a mechanism for learning the association strengths, this has not been tested with the fan effect. Instead, each S_{ji} for chunk i and slot-value j is

$$S_{ji} = \mathrm{S} + \ln(\, \mathrm{P}(i|j)\,)$$

where S is a fitting parameter and $\mathrm{P}(i|j)$ is the probability that chunk i will be useful given the presence of the concept j in the cue. In the fan effect, the chunk i might be "hippy park" and j might be *park*. If there are four people in the park then *park* has a fan of four. The probability that "hippy park" is the correct chunk given *park* is then 1/4 or, more generally, $1/f$ where f is the fan.

In the fan effect task, the experimental design is supposed to control for frequency and recency effects, and so the ACT-R model of the fan effect assumes all chunks have the same baseline activation, B_i, and thus baseline activation can be removed from the equation.

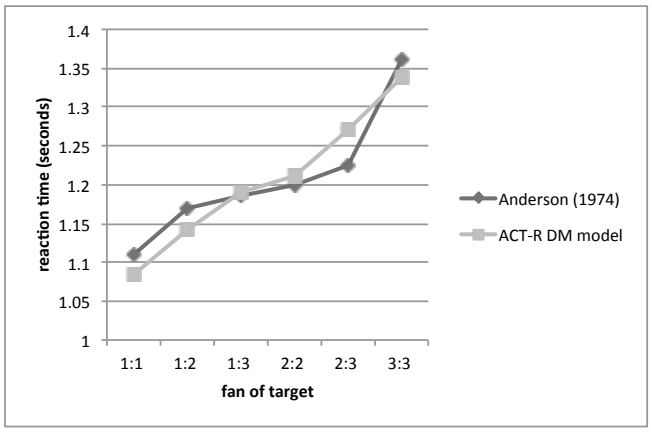


Figure 1: Real versus simulated reaction times for the fan effect from Anderson's (1974) data and Anderson and Reder's (1999) ACT-R DM model.

In Anderson and Reder's (1999) ACT-R model of the fan effect, reaction time for correctly identifying a target as belonging to the study set is calculated in milliseconds with the parameters $S = 1.45$, $W_j = 1/3$, $I = 845$, and $F = 613$. The target retrieval time for the fan effect model works out to be:

$$T = 239\, f_{\text{person}}^{1/3}\, f_{\text{place}}^{1/3} + 845$$

where f_{person} is the person's fan and f_{place} is the place's fan. This model provides a good fit to participant reaction times to targets in the fan effect task, $r = 0.95$ (see Figure 1).

Holographic Declarative Memory (HDM)

In Anderson and Reder's (1999) ACT-R DM model of the fan effect, it is necessary to set the correct association strength S_{ji} for each concept j and chunk i. However, Holographic Declarative Memory (HDM) produces the fan effect by learning the study set. The association strengths are not explicitly programmed. The studied items are presented to HDM as ACT-R chunks. HDM uses holographic reduced representations (Plate, 1995), a technique for instantiating and manipulating symbolic structure in high-dimensional vectors. To interface with ACT-R, HDM translates chunks into vectors, and vectors into chunks.

In HDM, a *value* is represented by a vector of n numbers randomly sampled from a normal distribution. These randomly generated vectors are referred to as *environment vectors*. Any two vectors chosen at random in a high dimensional space will tend to be approximately orthogonal. In HDM, angles indicate degrees of similarity. Orthogonality indicates complete dissimilarity. If we wanted to represent values with intrinsic similarity (e.g., *brother* and *sister*) we could choose non-orthogonal vectors, but for the purposes of modelling the fan experiment, we assume that the persons and locations are dissimilar.

In HDM, a *slot* is represented by a random permutation: a randomly selected reordering of a vector's elements. A *slot-value* pair is represented by reordering the elements of the *value* vector by the *slot* permutation.

Information storage in HDM is based on BEAGLE (Jones & Mewhort, 2007) and DSHM (Rutledge-Taylor et al., 2014). HDM is a *concept-based* memory system. Rather than storing chunks per se, HDM stores relationships between concepts, i.e., the *values* from an ACT-R chunk. Each concept is represented by two vectors: an environment vector $\mathbf{e}_{\text{concept}}$ that represents the percept of that concept, and a memory vector $\mathbf{m}_{\text{concept}}$ that stores the relationship between that concept and other concepts.

As information storage in HDM differs from DM, so too does the process of retrieval. To recall from DM, DM is given a retrieval cue that is a description of a chunk and DM retrieves a chunk that matches that description. Conversely, in HDM, a cue is a question, represented by a vector, and HDM retrieves the concept that best answers that question.

A memory vector for a concept, $\mathbf{m}_{\text{concept}}$, stores a list of questions to which HDM knows, from experience, that the concept is a candidate answer. When cued, that is, posed a question, HDM selects the memory vector with the greatest similarity to the cue and gives as answer the concept represented by that memory vector.

In Python ACT-R, a cue may contain the value question mark, '?', to indicate a 'wildcard', that is, an unknown value. DM can retrieve more than one unknown value at a time because it is retrieving a complete chunk. Whereas in HDM, each unknown value requires a separate retrieval because HDM retrieves a value rather than a chunk (though we are open to the possibility that these retrievals could be performed in parallel). A chunk used as a cue for recall in HDM must contain exactly one '?' to indicate the concept (i.e., value) that HDM should retrieve.

Memory Encoding and Recall with Slots

In HDM, there are two ways to structure knowledge corresponding to the two kinds of chunk in Python ACT-R: lists of *values* or unordered lists of *slot-value* pairs. We first discuss storing unordered slots-value pairs in HDM.

To store in HDM the chunk "colour:red shape:square size:large", we update the memory vector for each concept in the chunk: $\mathbf{m}_{\text{red}}$, $\mathbf{m}_{\text{square}}$, and $\mathbf{m}_{\text{large}}$. To update the memory vector for red, $\mathbf{m}_{\text{red}}$, we need to construct a vector representing the relationship between the concept red and all other concepts in the chunk and then add that vector to $\mathbf{m}_{\text{red}}$. In other words, we need to describe the set of questions for which *red* is an appropriate answer given "colour:red shape:square size:large" and add those questions to $\mathbf{m}_{\text{red}}$. Those questions are "What colour is it?", "What colour is the large thing?", "What colour is the square?" and "What colour is the large square?".

The question "What colour is it?" can be represented by the chunk "colour:?", "What colour is the large thing?" by the chunk "colour:? size:large", and "What colour is the large square?" by "colour:? size:large shape:square".

When the cue is translated into a vector, the '?' becomes the *placeholder* (Jones & Mewhort, 2007). The *placeholder*, denoted by $\mathbf{\Phi}$, is a vector used to encode all associations and thus serves as a universal retrieval cue. The placeholder is randomly generated like an environment vector. Using the placeholder, the cue "colour:?" is translated into the vector

$\mathbf{q}_{colour:?} = (\mathbf{P}_{colour}\ \mathbf{\Phi})$, where $\mathbf{P}_{colour}$ is the permutation representing the slot *colour*.

In holographic reduced representations (Plate, 1995), there are two ways of combining a pair of vectors to create a new vector: + vector addition and * circular convolution. An association between concepts is represented by convolving together the environment vectors representing those concepts. Addition is used to superimpose vectors representing separate information into a single vector.

The vector that represents the question "colour:? size:large" is $(\mathbf{P}_{colour}\ \mathbf{\Phi})*(\mathbf{P}_{size}\ \mathbf{e}_{large})$, i.e., the placeholder permuted by *colour* and convolved with *large* permuted by *size*. This vector will only match the memory vectors of concepts that are colours associated with large objects.

By default, HDM allows for partial matching of cues to concepts in memory. To do so, HDM translates each cue into a set of questions: the question explicitly specified by the cue and all less specific variants of that question. The cue "colour:? size:large shape:square" is translated into a sum of vectors representing "colour:? size:large shape:square" and also "colour:? size:large", "colour:? shape:square", and "colour:?", calculated as follows:

$$\begin{aligned}\mathbf{q}_{colour:?\ size:large\ shape:square} = \\ &(\mathbf{P}_{colour}\ \mathbf{\Phi}) \\ &+ (\mathbf{P}_{colour}\ \mathbf{\Phi})*(\mathbf{P}_{size}\ \mathbf{e}_{large}) \\ &+ (\mathbf{P}_{colour}\ \mathbf{\Phi})*(\mathbf{P}_{shape}\ \mathbf{e}_{square}) \\ &+ (\mathbf{P}_{colour}\ \mathbf{\Phi})*(\mathbf{P}_{shape}\ \mathbf{e}_{square})*(\mathbf{P}_{size}\ \mathbf{e}_{large})\end{aligned}$$

Cues are used both to retrieve from memory and to add new knowledge to memory. When the chunk "colour:red size:large shape:square" is added to memory, HDM updates $\mathbf{m}_{red}$, $\mathbf{m}_{square}$, and $\mathbf{m}_{large}$ as follows:

$$\begin{aligned}\Delta\mathbf{m}_{red} &= \mathbf{q}_{colour:?\ size:large\ shape:square} \\ \Delta\mathbf{m}_{square} &= \mathbf{q}_{colour:red\ size:large\ shape:?} \\ \Delta\mathbf{m}_{large} &= \mathbf{q}_{colour:red\ size:?\ shape:square}\end{aligned}$$

Given a retrieval cue, HDM selects the memory vector with the greatest similarity to the cue's vector and the cue's chunk is returned to ACT-R with the '?' substituted for the concept that the memory vector represents.

Similarity is measured by the cosine of the angle between vectors, which can be calculated as:

$$\text{cosine}(\mathbf{q}, \mathbf{m}) = (\mathbf{q} \bullet \mathbf{m}) / ((\mathbf{q} \bullet \mathbf{m})^{0.5} (\mathbf{q} \bullet \mathbf{m})^{0.5})$$

where $\mathbf{q}$ is a cue vector, $\mathbf{m}$ is a memory vector, and • is the dot product. The cosine is the dot product normalized by the magnitudes of the vectors. A cosine of 1 means the vectors are identical and 0 means they are completely dissimilar. HDM uses DM's retrieval time equation (Equation 2), but calculates activation as similarity measured by the cosine.

Vectors without Slots

Without slots, relationships are indicated by the order of the values in the chunk. Convolution is commutative, $\mathbf{a} * \mathbf{b} = \mathbf{b} * \mathbf{a}$, so the order is not preserved. To preserve the order we use $\mathbf{P}_{before}$, a random permutation indicating that a vector occurred before another a vector. To add the chunk "large red square" to memory, we would update $\mathbf{m}_{red}$, $\mathbf{m}_{square}$, and $\mathbf{m}_{large}$. We would update $\mathbf{m}_{red}$ as follows:

$$\begin{aligned}\Delta\mathbf{m}_{red} = &(\mathbf{P}_{before}\ \mathbf{e}_{large})*\mathbf{\Phi} \\ &+ (\mathbf{P}_{before}\ \mathbf{\Phi})*\mathbf{e}_{square} \\ &+ (\mathbf{P}_{before}\ ((\mathbf{P}_{before}\ \mathbf{e}_{large})*\mathbf{\Phi})*\mathbf{e}_{square}\end{aligned}$$

which adds the questions "large ?", "? square" and "large ? square" to the memory of the concept of red.

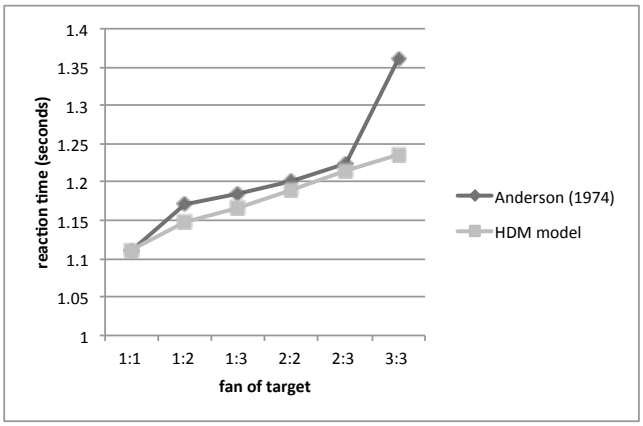


Figure 2: Real versus simulated reaction times for the fan effect from Anderson's (1974) data and the HDM model.

Recognition with Holographic Declarative Memory

In the DM model of the fan effect, the activation of a chunk is calculated as a weighted sum of the association strengths of the chunk's constituent concepts. In HDM, association strengths are measured by vector cosine, so we can calculate that activation in HDM as a weighted sum of cosines.

When determining whether HDM recognizes a cue, the cue chunk must contain no unspecified values '?'. For each value in the cue, HDM creates a new cue with that value substituted for '?', performing one retrieval for each value in the original cue. Activation is calculated as the mean of the cosines between each of these cues and the memory vector of the concept that was substituted out to create the cue. This method for calculating activations in the fan effect has been used before by DSHM (Rutledge-Taylor et al., 2014; Rutledge-Taylor, Pyke, West, & Lang, 2010).

In the fan effect task, for the cue "hippy park", HDM does two retrievals, "hippy ?" and "? park" with the vectors $\mathbf{q}_{hippy?} = (\mathbf{P}_{before}\ \mathbf{e}_{hippy})*\mathbf{\Phi}$ and $\mathbf{q}_{?park} = (\mathbf{P}_{before}\ \mathbf{\Phi})*\mathbf{e}_{park}$. Activation A is calculated as:

$$A = 0.5\ \text{cosine}(\mathbf{q}_{hippy?}, \mathbf{m}_{park}) + 0.5\ \text{cosine}(\mathbf{q}_{?park}, \mathbf{m}_{hippy})$$

The HDM Model of the Fan Effect

We ran the HDM model of the fan effect task 20 times, simulating 20 virtual participants, and averaged across runs. Because each run uses a different set of random vectors, the cosines and reaction times vary randomly with each run. Anderson's (1974) experiment had 18 participants. The model fits the human participant data reported by Anderson (1974) with a correlation of $r = 0.91$ (see Figure 2). The fit

was obtained using the exact same values for the fitting parameters as Anderson and Reder's (1999) ACT-R fan effect model. The only change was to compute activation as a mean of cosines, as described in the previous section.

Anderson and Reder's (1999) model and the HDM model are strongly correlated, $r = 0.99$. While there are slight differences in the predictions made by the two models, both the DM and HDM models are within the range of human variability for performance on this task. These results show that HDM replicates DM's ability to model the fan effect, but HDM does so in a radically different way: by measuring the cosine between vectors in a high-dimensional space.

Why does the cosine model the fan effect so well? The cosine acts as an estimate of the conditional probabilities that the Anderson and Reder's (1999) fan effect model uses to compute association strengths. The memory vector for a concept keeps a fuzzy count of the number of times that concept has co-occurred with each other concept. Taking the dot product of the cue with a memory vector gives you an estimate of the frequency with which that cue has been added to that memory vector, that is, the number of times the relationships described in that cue have occurred with that concept. The cosine is a dot product normalized by the magnitudes of the vector, which in this case, is a frequency normalized by the total number of instances, that is to say, the cosine is roughly the probability.

We can imagine all vectors in HDM as points on a *n*-dimensional hypersphere. For the HDM fan effect model, we used 256 dimensions, but for the sake of visualization, imagine a 3-dimensional sphere.

Let us first consider a fan of one. Suppose the model has learned only one fact about the hippy, namely, the "hippy is in the park". After learning this fact, the memory vector for hippy will be $\mathbf{m}_{\text{hippy}} = (\mathbf{P}_{\text{before}}\ \mathbf{\Phi})*\mathbf{e}_{\text{park}}$. The model is later given the cue "the hippy is in the park" during the recognition phase. To test for recognition, we take the cosine of $\mathbf{m}_{\text{hippy}}$ with the cue $\mathbf{q}_{?\text{park}} = (\mathbf{P}_{\text{before}}\ \mathbf{\Phi})*\mathbf{e}_{\text{park}}$. As $\mathbf{m}_{\text{hippy}} = \mathbf{q}_{?\text{park}}$ the angle between the cue and the memory vector is zero, the distance between them on the surface of the hypersphere is zero, and the cosine is 1.00.

Let us consider a fan of two. If the model knows "hippy is in the park" and "hippy is in the bank", then $\mathbf{m}_{\text{hippy}}$ is the sum of the park cue $\mathbf{q}_{?\text{park}}$ and the bank cue $\mathbf{q}_{?\text{bank}}$,

$$\mathbf{m}_{\text{hippy}} = (\mathbf{P}_{\text{before}}\ \mathbf{\Phi})*\mathbf{e}_{\text{park}} + (\mathbf{P}_{\text{before}}\ \mathbf{\Phi})*\mathbf{e}_{\text{bank}}$$

In high dimensional spaces, randomly chosen vectors are approximately orthogonal to each other. Let us assume that the cues $\mathbf{q}_{?\text{bank}}$ and $\mathbf{q}_{?\text{park}}$ are perfectly orthogonal. As illustrated in the left half of Figure 3, on the surface of the hypersphere, $\mathbf{m}_{\text{hippy}}$ will be halfway between the two cues at a 45° angle. The cosine is 0.71.

Let us consider a fan of three. If the model knows that the hippy is in the bank, park, and store, $\mathbf{m}_{\text{hippy}}$ will be at an equidistant point on the hypersphere between the cues for bank, park, and store. In the fan of three, $\mathbf{m}_{\text{hippy}}$ is further away from all the cues than in a fan of two. The angle between $\mathbf{m}_{\text{hippy}}$ and any cue is 55° and the cosine is 0.58.

Where f is the fan, the cosine between a cue and a memory vector is $f^{-1/2}$ if the vectors are perfectly orthogonal, or

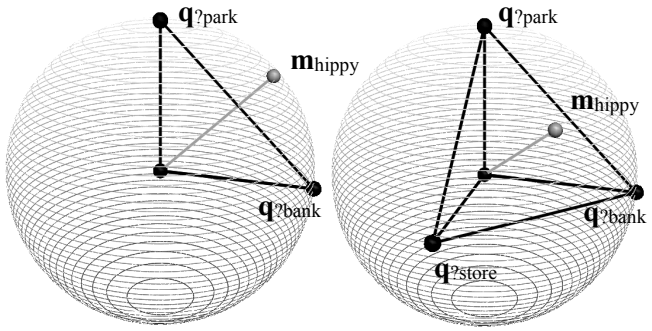


Figure 3: $\mathbf{m}_{\text{hippy}}$ with a fan of 2 (left) or 3 (right).

approximates $f^{-1/2}$ for the random vectors used by HDM. Thus HDM predicts that as the fan increases, the cosine decreases, but by diminishing amounts with each increase in fan. As the fan approaches infinity, the cosine approaches zero. HDM makes the intuitive prediction that increases in the fan has a steadily diminishing effect on reaction time, such that knowing 100 facts about the hippy is not appreciably different from knowing 101.

The cosine in HDM approximates the square-root of the probability only when the events are equiprobable. For n events with frequencies v_1 to v_n, the cosine of event i is

$$\text{cosine} = \frac{v_i}{\sqrt{v_1^2 + \ldots + v_i^2 + \ldots v_n^2}} \tag{3}$$

When given events of unequal probabilities, HDM will behave as if the most frequent events are disproportionately likely and the least frequent events are disproportionately unlikely. This is a testable and possibly erroneous prediction of HDM. The quantum probability model of human judgements (Busemeyer, Pothos, Franco, & Trueblood, 2011) also uses vector algebra to calculate probabilities, but uses the square-roots of the frequencies, then squares the cosine, such that Equation 3 is equal to classical probability. Using the square-roots of the frequencies is not possible for HDM as it would require HDM to know a priori how frequently each event will occur.

Future Work and Applications of HDM

We have presented in this paper an HDM model of the fan effect and compared it to Anderson and Reder's (1999) DM model of the fan effect. However, we have only discussed fitting to the reaction time of targets, sentences presented at the recognition phase that occurred in the study set. Anderson and Reder's (1999) model for foils, sentences that were not in the study set, fails on a variant of the fan effect task (West, Pyke, Rutledge-Taylor, & Lang, 2010). As the foil is difficult to model, we leave developing an HDM model of the foil for future research.

At present, HDM does not model recency effects, that is, more recent information is not recalled better than less recent information. However, other holographic models in the literature (e.g., Franklin & Mewhort, 2015; Murdock 1993) can account for recency effects, so such a mechanism could be incorporated into the model.

At present, interfacing with ACT-R chunks imposes an information bottleneck on HDM. Detailed sensory information cannot be feasibly stored in ACT-R chunks, but can be stored in holographic vectors (Kelly, Blostein, & Mewhort, 2013). Reimplementing the entirety of ACT-R as a holographic system would improve ACT-R's ability to interface with real world environments and to match situations to procedures. Some of that work has already been done: A holographic model similar to ACT-R's procedural memory system already exists as the basal ganglia model of the SPAUN brain model (Eliasmith, 2013; Stewart, Bekolay, & Eliasmith, 2012). However, a holographic procedural memory consistent with HDM and ACT-R would necessarily differ from SPAUN's to meet the demands of integration with a different architecture.

HDM is a powerful tool for cognitive modellers because it inherits the abilities of holographic models such as BEAGLE (Jones & Mewhort, 2007) and DSHM (Rutledge-Taylor, Vellino, & West, 2008) to store large quantities of data in memory and use it to make intelligent predictions in knowledge-heavy tasks. In Rutledge-Taylor et al. (2014) we show that DSHM can be used to model a difficult but small-scale decision-making task. HDM could be applied to a large-scale, knowledge-driven decision-making task.

Conclusion

We present a new module for ACT-R, Holographic Declarative Memory (HDM). We substitute HDM for DM in the ACT-R model of the fan effect and find that without changing any parameters HDM provides a good fit to the fan effect. We present an analysis that allows us to specify the mathematical relationship between the DM and HDM models of the fan effect.

HDM, by virtue of being a holographic model, has a number of capabilities for which DM is less suited, such as analogical or case-based reasoning, learning associations between concepts without having association strengths set by the modeller, and performing tasks that require large amounts of knowledge. We hope that by integrating a holographic memory model into ACT-R, we can bring the capabilities of vector space modelling into the ACT-R research community and enhance the capability of the ACT-R cognitive architecture to model human cognition.

Acknowledgments

This research is supported by an Ontario Graduate Scholarship awarded to the first author and a grant from the National Science and Engineering Research Council of Canada to the third author.

References

Anderson, J. R. (1974). Retrieval of propositional information from long-term memory. *Cognitive Psychology, 6,* 451-474.

Anderson, J. R. (1991). The place of cognitive architectures in a rational analysis. In K. Van Len (Ed.), *Architectures for Intelligence*. Hillsdale, NJ: Erlbaum.

Anderson, J. R., & Lebiere, C. (1998). *The Atomic Components of Thought*. Mahwah, NJ: Lawrence Erlbaum Associates.

Anderson, J. R., & Reder, L. M. (1999). The fan effect: New results and new theories. *Journal of Experimental Psychology: General, 128,* 186-197.

Busemeyer, J. R., Pothos, E. M., Franco, R., & Trueblood, J. (2011). A quantum theoretical explanation for probability judgement errors. *Psychological Review, 118,* 193-218.

Eliasmith, C. (2013). *How to build a brain: A neural architecture for biological cognition*. Oxford University Press.

Eliasmith, C., & Thagard, P. (2001). Integrating structure and meaning: a distributed model of analogical mapping. *Cognitive Science, 25*, 245-286.

Franklin, D. R. J., & Mewhort, D. J. K. (2015). Memory as a hologram: An analysis of learning and recall. *Canadian Journal of Experimental Psychology, 69*, 115-135.

Gabor, D. (1969). Associative holographic memories. *IBM Journal of Research and Development, 13*, 156–159.

Jones, M. N., & Mewhort, D. J. K. (2007). Representing word meaning and order information in a composite holographic lexicon. *Psychological Review, 114*, 1-37.

Kelly, M. A., Blostein, D., & Mewhort, D. J. K. (2013). Encoding structure in holographic reduced representations. *Canadian Journal of Experimental Psychology, 67*, 79-93.

Murdock, B. B. (1993). TODAM2: a model for the storage and retrieval of item, associative and serial-order information. *Psychological Review, 100*, 183–203.

Longuet-Higgins, H. C. (1968). Holographic Model of Temporal Recall. *Nature, 217*, 104.doi: 10.1038/217104a0

Plate, T. A. (1995). Holographic reduced representations. *IEEE Transactions on Neural Networks, 6*, 623–641. Plate, T. A. (2000). Analogy retrieval and processing with distributed vector representations. Expert Systems: The International Journal of Knowledge Engineering and Neural Networks, 17, 29-40.

Rutledge-Taylor, M. F., Pyke, A. A., West, R. L., Lang, H. (2010). Modeling a three term fan effect. In D. D. Salvucci & G. Gunzelmann (Eds.), *Proceedings of the 10th International Conference on Cognitive Modeling* (pp. 211-216). Philadelphia, PA: Drexel University.

Rutledge-Taylor, M. F., Kelly, M. A., West, R. L., & Pyke, A. A. (2014). Dynamically structured holographic memory. *Biologically Inspired Cognitive Architectures, 9*, 9-32.

Rutledge-Taylor, M. F., Vellino, A., & West, R. L. (2008). A holographic associative memory recommender system. In *Proceedings of the 3rd International Conference on Digital Information Management* (pp. 87-92). London, UK.

Stewart, T. C., Bekolay, T., and Eliasmith, C. (2012) Learning to select actions with spiking neurons in the basal ganglia. *Frontiers in Neuroscience, 6:2*, 1-14.

West, R. L., Pyke, A. A., Rutledge-Taylor, M. F., & Lang, H. (2010). Interference and ACT-R: New evidence from the fan effect. In D. D. Salvucci & G. Gunzelmann (Eds.), *Proceedings of the 10th International Conference on Cognitive Modeling* (pp. 277-281). Philadelphia, PA: Drexel University.

Modeling Two-Channel Speech Processing with the EPIC Cognitive Architecture

David E. Kieras (kieras@umich.edu)
Electrical Engineering & Computer Science Department, University of Michigan
2260 Hayward Street, Ann Arbor MI 48109-2121, USA

Gregory H. Wakefield (ghw@umich.edu)
Electrical Engineering & Computer Science Department, University of Michigan
2260 Hayward Street, Ann Arbor MI 48109-2121, USA

Eric R. Thompson (eric.thompson.22.ctr@us.af.mil)
Ball Aerospace & Technologies Corp.
2610 Seventh St. B441, Wright-Patterson AFB, Ohio 45433

Nandini Iyer (nandini.iyer.2@us.af.mil)
Battlespace Acoustics Branch, Air Force Research Laboratory
2610 Seventh St. B441, Wright-Patterson AFB, Ohio 45433

Brian D. Simpson (brian.simpson.4@us.af.mil)
Battlespace Acoustics Branch, Air Force Research Laboratory
2610 Seventh St. B441, Wright-Patterson AFB, Ohio 45433

Abstract

An important application of cognitive architectures is to provide human performance models that capture psychological mechanisms in a form that can be "programmed" to predict task performance of human-machine system designs. While many aspects of human performance have been successfully modeled in this approach, accounting for multi-talker speech task performance is a novel problem. This paper presents a model for performance in a two-talker task that incorporates concepts from the psychoacoustic study of speech perception, in particular, masking effects and stream formation.

Keywords: Cognitive architecture; two-channel speech; auditory perception; auditory streams

Introduction

A classic problem in cognitive psychology is the "cocktail party effect" in which a person is surrounded by several people speaking simultaneously, and is nonetheless able to follow a single speaker well enough to maintain a conversation, although some information about what the other speakers are saying appears to be available under some conditions. The early study of these phenomena (e.g. Cherry, 1953) led to a body of additional studies and theoretical work that defined the current concept of selective attention; the human listener was said to be able to selectively attend to one of the signal sources and "filter out" the others. The most common experimental paradigm is that the subject must listen to simultaneous speech inputs from two or more talkers (human speakers), but respond to the information provided by only one of them. Some more recent research over the last decade has used more precise procedures to help characterize the determinants of performance; in particular many experiments have been done using the *coordinate response measure* (CRM) speech corpus which represents a highly simplified form of the command and control messages used in military settings (Bolia, Nelson, Ericson, & Simpson, 2000).

The mainstream psychoacoustic work on this problem applied the mathematical tools of signal analysis that have been successful in characterizing human ability to detect and discriminate sounds. A less formal but influential concept was *auditory streams* (Bregman, 1990), the notion that we perceive separate sound sources based on the detailed properties of the incoming sounds. In a two-talker task, each talker would be perceived as a stream, and the listener's task is to determine which sounds go with which stream and choose the appropriate response. This process must involve a combination of perceptual mechanisms and cognitive strategies. However, psychoacoustic accounts of the task have focussed on "front end" processes of signal detection and processing and did not have a well-defined way to take into account the possibly complex "back end" processes of cognitive strategies involved in the task. In contrast, cognitive architecture research developed powerful theoretical mechanisms for the "back end" processing, especially using production systems, but tended to ignore difficult details of perceptual processes.

The present paper combines mathematical models of speech perception with a cognitive architecture to model human performance in a two-talker listening task. EPIC (Executive/Process-Interactive Control) is one among several architectures whose goal is to provide an integrated account of human abilities and limitations in perception, cognition, and action. A psychoacoustic speech perception model was incorporated into the EPIC cognitive architecture to provide an integrated account of performance in a well-studied two-talker speech perception task. We devised a relatively simple speech perception model and a strategy which together account for important factors that determine performance.

An earlier form of this model appears in Kieras, Wakefield, Thompson, Iyer, & Simpson (2014); the model presented here has the same strategy component, but the

perceptual models are considerably improved, taking into account how pitch differences affect detection and stream segregation. The result is a model with far fewer parameters that must be estimated from the data. A detailed comparison of the improved perceptual model with the previous one is not possible in the available space here; the reader can compare this model with the one in Kieras, et al (2014).

Following a review of the two-talker CRM listening task, an overview of EPIC will be presented and key extensions of the auditory processing module will be introduced. Within the framework imposed by these extensions, a model for the two-talker CRM listening task will be proposed and fit to the human data.

Replication of a Two-Talker Dataset

The CRM corpus is a collection of recorded command utterances in the form of

Ready <Callsign> go to <Color> <Digit> now

spoken by one of four females or four males, where the Callsign, Color, and Digit are drawn from sets of 8, 4, and 8 items, respectively. The corpus was recorded and edited to maintain a high degree of temporal overlap among the spoken Callsigns, Colors and Digits (Bolia, et. al., 2000).

In the two-talker CRM listening task, participants respond to commands by selecting the appropriate Color/Digit pair from a display. A particular Callsign is designated as the Target Callsign, which was always *Baron* in the studies used in this paper. On each trial, a *Target* message is drawn from those utterances bearing the Target Callsign and is presented simultaneously with a randomly selected Masker message, with the restriction that the Callsign, Color and Digit of the Masker differ from those of the Target. The participant thus hears two messages whose words are simultaneous, and must choose the color-digit pair associated with the Target callsign, and was instructed to ignore the Masker message. The responses are scored as matching the Target message, the Masker message, or Neither.

An important study by Brungart (2001) stimulated our first modeling. He manipulated the acoustic similarity of the two talkers, varying from Different Sex (DS), to Same Sex (SS), to Same Talker (ST), and also manipulated the relative loudness of the two messages, with a Signal-to-Noise ratio (i.e. the Target-to-Masker ratio) ranging from -12 to +15 dB. This study is important because in addition to reporting the proportion of *completely correct* responses (both Color and Digit are Target), he also reported the proportions of responses that matched Target, Masker, or Neither separately for Color and Digit.

Rather than show his results in this paper, however, we present the results for a methodologically improved replication which is very similar in design and results to Brungart (2001). The replication followed the conditions and procedures of Brungart (2001) in all respects except two: (1) The SNR, which ranged from -12 to +15 dB in the original study, was shifted to a lower range (-18 to +9 dB) in the interest of studying performance at SNRs closer to

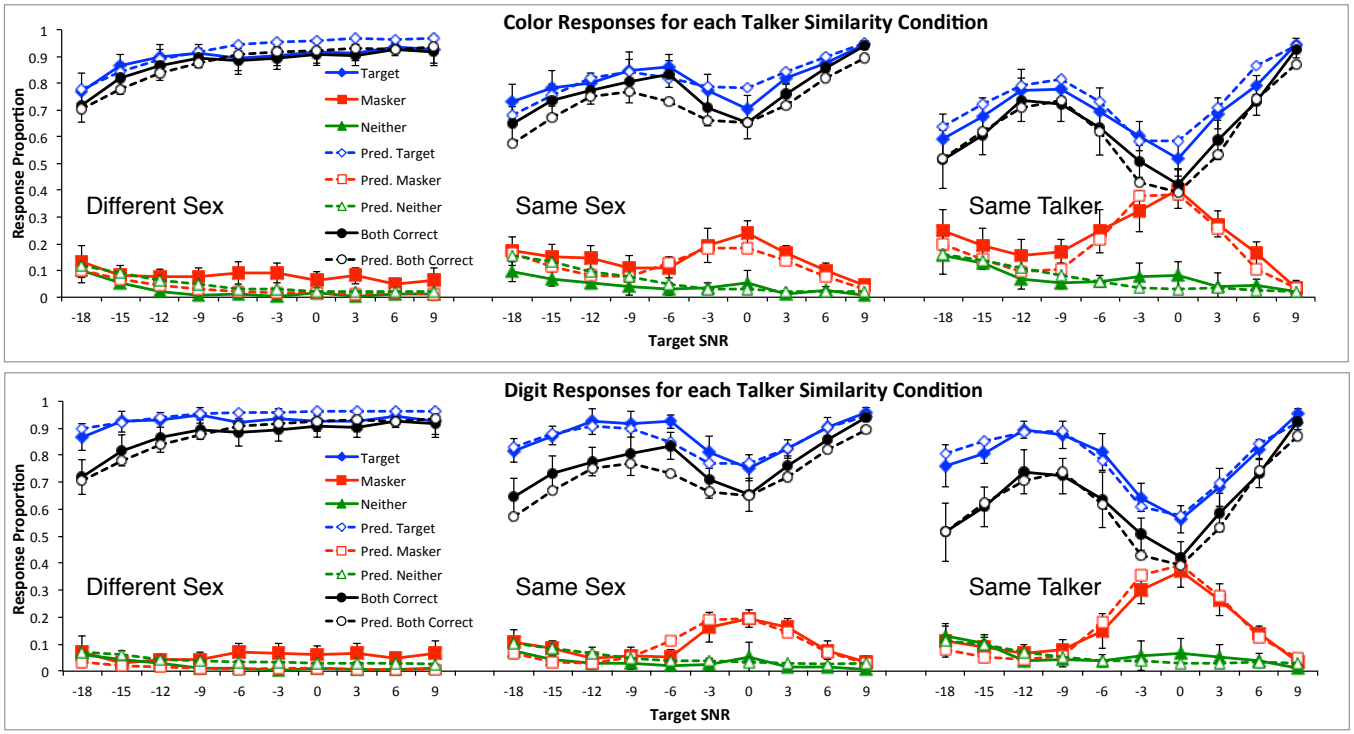


Figure 1. Observed (solid points and lines) and Predicted (open points and dotted lines) proportion of responses as a function of SNR and talker similarity. Top panel shows Color responses, bottom panel shows Digit responses. In order from the top down, the curves are as follows: Blue curves with diamond points are for Target responses, black curves with circle points are for completely correct responses (both color and digit from the Target), and are the same in the top and bottom panels; red curves with square points are for Masker responses, and green curves with triangles for neither Target nor Masker. Error bars show 95% confidence intervals for the means averaged over individual subject proportions.

masked detection thresholds; (2) the replication clarified the task instructions with a point reward system for correct performance, and provided performance feedback at the end of each trial and during the experiment.

Results

The six panels of Figure 1 show these somewhat complex experiment results as the *observed* points (solid points and lines; the *predicted* points will be explained later). Each panel plots the proportion of Target, Masker, and Neither responses as a function of the signal-to-noise (SNR) ratio in dB. The upper and lower panels display the proportion for Color and Digit responses separately. In addition, the panels show the proportion of Both-Correct responses in which both Color and Digit are from the Target message. These black curves are the same in the upper and lower panels. The left-to-right panels display the results based on the similarity of the Target and Masker talkers. From left to right, the stimulus conditions are Different Sex, Same Sex but different talkers, and Same Talker.

The basic effects are as follows: overall, with increasing positive SNR, the completely correct and Target Color and Digit responses are chosen more often, and Masker and Neither content are chosen less often. The overall performance when the messages are delivered by Different-Sex talkers is better than that for Same-Sex talkers, which is turn is better than that when the two messages are from the Same Talker. For the Same-Sex and Same-Talker conditions, accuracy is very poor at very low (negative) SNRs, but then improves, and then declines again in the vicinity of 0 dB SNR, and then improves again.

A key empirical fact is that the incorrect responses were almost always from the Masker message, which places a basic constraint on the cognitive processes in any model, in that it implies that Masker message content was being perceived and remembered, and then chosen as a response, rather than being simply filtered out, as would be expected from a simple selective attention model.

Accounting for the Phenomena

To date, a theoretical account of the two-talker CRM results remains incomplete. Discussions have focused on the relative importance of informational masking over energetic masking, the roles of selected and divided attention, and the formation and maintenance of auditory streams. However, none of these concepts have been operationalized to the point of providing strong predictions of experimental outcomes. What follows is an attempt to help bridge this gap.

The focus of our work was to account for these results in terms of a basic concept of human cognitive architecture and a quantitative model based on that concept. The resulting model incorporates mechanisms that resemble both energetic and informational masking, but do so with considerably more theoretical precision; most importantly, the strategy that the subject follows to perform the task is directly represented, and this turns out to be critical in accounting for the specific effects in this data.

The Architecture and Model

An EPIC architecture model comprises a simulated task environment which interacts with a simulated human; the architecture describes the fixed components of the simulated human, controlled by a task-specific strategy represented as production rules. Due to space limitations, the usual description of the architecture is not provided here; see Meyer and Kieras (1997, 1999) or Kieras (in press) for more discussion. The focus of this presentation is on the mechanisms of the auditory processor that have been added to the architecture, and the production-rule strategy for the task.

Model Summary

The application of a cognitive architecture to multichannel speech processing is novel, and so needs to be presented with some detail, but for brevity, low-level representational issues are not presented here. Rather, the emphasis is on the conceptual design of the architecture and model components, especially the auditory processor, taking into account that at this time many processes have to be "black boxed". The following is a compact description of the architecture and model components and processing involved in the two-talker CRM task, flowing from input to response. In some of what follows, the description is somewhat more complex because the mechanism is general enough to apply to more than two talkers.

Speech auditory input. Each utterance is pre-parsed into six segments corresponding to words (with *go to* being treated as a single word). The segments from the different sources are assumed to arrive at the auditory processor simultaneously and are each perceived as individual auditory events. Each segment pair is processed in order of arrival.

Auditory perception constructs *auditory objects* based on properties of the physical input. There are two kinds of auditory object: *word objects* represent individual perceived words that have a temporal duration; *stream objects* represent perceived sound sources for these word objects.

Word objects. Word objects have a variety of properties, but for the purposes of this model, they may or may not have *content*, which is the recognized semantic item (e.g. *red*); this allows for a word to be "heard" but not recognized. Words also have *stream attributes*, which in this model are average loudness level (specified in dB) and average pitch (in semitones, where the number of semitones is defined as $12 \cdot \log_2(\text{pitch in Hz})$), both averaged over the duration of the word. Semitones provide a logarithmic scale for pitch, analogous to decibels for loudness. This model assumes that the stream attributes are *always* perceived.[1]

Whether the content of a word object is recognized in the presence of the other word objects is assumed to be a basic masking phenomenon. The probability of content detection depends on the SNR, that is, the loudness level of the word

[1] For simplicity, we are assuming that perceived pitch and loudness correspond to physical semitones and dB.

relative to the other word objects that are simultaneously present, and the pitch difference between the two word objects. With respect to the latter, studies show that discrimination of simultaneous vowel sounds improves with pitch difference, though increasing the difference beyond about 4 semitones produces no further improvement (Assmann & Summerfield, 1990). This effect was incorporated in the model by computing an *Effective SNR* that is the weighted sum of the loudness difference in dB (the SNR) and the pitch difference in semitones capped at 4.

Stream objects and stream tracking. The stream objects also have attributes of loudness and pitch, but these represent the overall properties of the perceived sound source. In this model, a stream object carries the mean loudness and mean pitch of the words associated with the stream. For example, a typical female talker will be represented as stream percept with a higher mean pitch property than that for a typical male talker.

The auditory perceptual processor assumes that there are as many stream objects as input sources, each with a unique but arbitrary *StreamID* attribute, and attempts to assign each incoming word object to one of the streams, using the stream-related attributes of loudness and pitch to do so. Once the assignment is done, the stream percepts are updated to reflect the loudness and pitch properties of the words assigned to them, and the next pair of word objects will be assigned to the updated streams. Thus the auditory processor *tracks* the streams.

Cognitive strategy and response choice. The final output of perceptual processing, represented in the cognitive processor's working memory, is a set of word objects and a set of stream objects. Each word object will always be associated with a stream object, but it may or may not have recognized content.

Because the loudness and pitch of each word in the utterances varies within the same talker, it is possible for individual words from two different talkers to be mis-assigned to the streams, so that each stream is associated with a mixture of words from the two talkers. Figure 2 shows an example in which the Color words have been assigned to the wrong stream, while the Digit words were assigned to the correct stream. This will lead to a response with the Masker Color and the Target Digit.

The cognitive process for selecting a response makes use of the recognized content of the word objects together with the stream associated with each word object. For example, as in Figure 2, if the word object whose content is the Target Callsign *Baron* is associated with Stream2 and there are two word objects associated with the same stream whose content has been recognized as the Color *Red* and the Digit *8*, then *Red 8* will be used to specify the response to be made.

Some content might be unrecognized, but in many cases the model strategy can infer the missing information. For example, if only one of the Callsign contents was recognized, and it was a Masker Callsign, the model can infer that the unrecognized Callsign word object was the Target Callsign, and its assigned stream must be the Target stream, so the Color and Digit words associated with that same stream must be the Target Color and Digit. Thus the strategic component of the model tries to make use of partial information to perform the task.

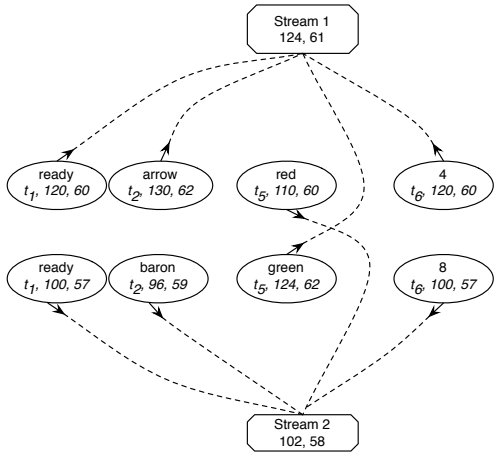


*Figure 2. Example showing contents of working memory after erroneous stream tracking. The polygonal boxes top and bottom are the two stream objects, showing mean pitch (Hz) and loudness level (dB) values. The ovals are the word objects in each message in left-to-right time order (*goto *and* now *omitted for clarity), showing the content, time stamp, pitch, and loudness. During perception, each word was associated with its closest stream, but because the Color word pitches were discrepant, they were assigned to the wrong stream.*

Theoretical summary. In terms of conventional attention theory, this is a "*very* late selection" model - *all* of the information produced by perception is available to cognition for choosing the response.

The problems of trying to handle two simultaneous messages is not represented as a failure to select the correct stream prior to cognition, but rather that masking effects and errors in stream assignments will result in a collection of perceptual information about the messages that may be incomplete or incorrect (e.g. as in Figure 2), and the task strategy must make use of this information to choose a response that meets the task requirements.

Model Details and Parameters

Corpus statistics drive the model. We computed the average loudness and pitch over each segment in each utterance in the CRM corpus, and supplied this information for each word (segment) that was "heard" by EPIC's auditory processor. An interesting result is that while female talkers had mean pitches about an octave higher than male talkers, individual talkers had somewhat different baseline pitches, which allows the stream tracking to often distinguish talkers within genders over the course of an utterance. Because this model was driven by the corpus properties, there are relatively few free parameters that affect its fit to data.

For each trial, the simulated experiment samples two utterances and then supplies EPIC's auditory system with the content, loudness, and pitch of each segment. The pitch was converted to semitones. Inside the auditory system module, pitch differences were always capped at 4 semitones, a constant value based on Assmann & Summerfield (1990) and not estimated to fit the data.

Content detection parameters. The content detection

parameters are summarized in Table 1. The *Effective SNR* is the sum of the loudness SNR and the pitch difference in semitones weighted by a parameter *w*.

The content detection process is modeled along the lines suggested by Wichman & Hill (2001). With a low probability (the lapse rate α), subjects will fail to recognize content (even at very high SNR); otherwise, the probability of content detection follows a gaussian detection function of Effective SNR, with parameters of mean μ and standard deviation σ. The parameters *w,* α and σ are assumed to be constant across the type of content word (Callsign, Color, Digit), while μ is assumed to have a different value for each type of content word (Callsign, Color, Digit). For completeness, the content detection functions for the filler words *ready*, *goto*, and *now*, were specified, but for simplicity were made the same as the Callsign detection function because the *content* of the filler words plays no role in stream tracking or response strategy.

Stream tracking details and parameters. The stream tracking parameters are also summarized in Table 1. The stream perception model in the EPIC auditory processor uses an *averaging minimum-distance* stream tracking algorithm. Each stream object accumulates the mean pitch (in semitones) and mean loudness (in dB) of the word segments that have already been assigned to that stream. The stream predicts that the pitch and loudness of the next, or new, word segment will be the same as the current means. The stream perception model then calculates the prediction error between each stream and each new word segment as the weighted cartesian distance between the (pitch, loudness) values, where pitch differences are weighted by a parameter λ (0-1) and loudness differences are weighted by (1 - λ). The pitch difference was capped at 4 semitones. The new word segments are then assigned to streams so as to minimize the total distance between all words and their assigned streams. The streams are then updated to include their newly assigned word segments, and the resulting means used to predict the segment that follows.

The stream perception model included a noise component. After determining the minimum-distance assignment, the stream perception process compares the maximum and minimum total distance; if the difference is less than or equal to a threshold value θ, an assignment is chosen at random.

Cognitive processor strategy exploration. The auditory perception components in the EPIC architecture take the input utterance segments and perform content detection and stream tracking and provide the resulting content and StreamID attributes of the individual word segments, like that shown in Figure 2, to the cognitive processor, which is running a strategy implemented in production rules. Over the course of this work, a variety of strategies were considered, and two key options were identified. The first is that in the 2-channel task, symmetrical inferences can be made; for example, if we know that one of the Color words is from the Masker stream, we can infer that the other Color word has to be from the Target stream.

The second option concerns the "guessing" strategy. Note that in this forced-choice paradigm, the subject must respond even if they have not identified the Target Color or Digit. The optimum strategy would seem to be to always avoid responding with known Masker content, and choose some Neither Color or Digit instead. However, this *Avoid-Masker* strategy failed badly to fit the data - it could not account for how there are so many Masker responses in conditions where the Masker stream should be easily identified, such as at extreme negative SNRs. We realized that subjects might adopt a "use what you heard" heuristic: If the Target callsign content was not actually detected, then there is some uncertainty about whether the two streams were correctly identified, so responding using content that was actually detected is better than a pure guess. Thus the *Use-Maskers* strategy will use content known to be from the Masker stream if Target content was not detected, but only if the identity of Target stream had been *inferred* from detection of Masker callsign content. This model used both the symmetrical inferences and the Use-Masker options.

Table 1. Best-fit parameter values

Effective SNR pitch weight *w*	2.00
Callsign content detection μ	-20.00
Color content detection μ	-18.00
Digit content detection μ	-26.00
Content detection σ	10.00
Content detection lapse rate α	0.04
Stream tracking pitch weight λ	0.80
Stream tracking distance threshold θ	0.10

Strategy summary. During the processing of the utterance, if Callsign content is present (detected), tag its StreamID as the Target or Masker stream accordingly. If not, infer the Target or Masker status from the other stream if its Callsign content is present. Then tag the Target or Masker status of each Color and Digit word, based on their assigned StreamIDs. Note that if neither Callsign is detected, it is still possible for Color and Digit words to be paired with their correct streams, but the model will not know which stream is the Target stream or the Masker stream.

When it is time to choose a response, the following rules are used for both choosing the color response and choosing the digit response, depending on what content was detected and which stream it is associated with: If the Target stream is known or inferred, then use the content from the Target stream if it is available. But if the Target stream was only inferred and the Target content is not available, then use the Masker content if it is available. Otherwise, use a color-digit content pair from the same stream if available, or use separate color and digit content if it is available; otherwise, make a pure guess.

Model Fitting and Results

The parameter values shown in Table 1 were determined by Monte-Carlo runs of the EPIC model with a grid search of the parameter values using high-performance clusters provided by AFRL through mindmodeling.org. The search goal was to maximize r^2 between predicted and observed values for the Target and Masker Color and Digit

probabilities (blue and red curves in Figure 1). Each Monte-Carlo run used 3000 trials per talker/SNR condition. There are a total of 240 empirical data points with at least 120 degrees of freedom; eight parameter values were varied in the search. The best-fit values are shown in Table 1.

Figure 1 shows the predictions from the EPIC model as open points and dotted lines. All three conditions are well handled with a small set of parameters that describe how the auditory perceptual process is affected by the acoustic properties of the input as provided by the corpus statistics based on the segmentation. It is especially noteworthy that unlike the model presented in Kieras et al. (2014), there are no parameters that are specific to talker similarity conditions - the pitch difference used in detection and tracking accounts for these effects.

As summary measures of goodness of fit, $r^2 = 0.99$ between predicted and observed values for the Target and Masker Color and Digit probabilities (blue and red curves), and $r^2 = 0.95$ for the completely-correct probabilities (black). Only a few of the predicted values lie outside the confidence intervals in the data.

However, there is a clear tendency for the completely-correct points to be generally under-predicted, probably because our simple model of the stream tracking is not "sticky" enough. That is, a detailed look shows that subjects are more likely than the model to choose the Target Digit if they have chosen the Target Color, as opposed to switching to the Masker or Neither Digit. The result is a tendency to under-predict the completely-correct responses, even though the individual Target and Masker responses are well predicted.

Conclusions

The EPIC auditory architecture has been extended to include explicit mechanisms for auditory stream perception and tracking. These mechanisms rely on acoustic properties of the speech input itself, in this case, the statistics of the corpus.

We now have a successful model of the two-talker task in which stream tracking based on basic acoustic characteristics of speech accounts very well for data from the two-talker task. Further refinement of the model for the stream tracking process may improve the fit, and there may be ways to reduce the number of free parameters in the detection functions. Work in progress suggests that this model may also scale to three- and four-talker tasks; in fact, the model as described functions in the three- and four-talker cases; the theoretical issue is how to correctly capture the substantially poorer performance produced by having multiple maskers.

In addition, the two-talker model can account for the original Brungart (2001) data if complex suboptimal mixture model strategies are implemented to represent the apparently under-constrained strategies adopted by the subjects. This last result urges that better experimental control of subject strategies, as in our replication experiment, should be used in future experiments on this topic, and that modeling should attempt to explore alternative subject strategies systematically.

Acknowledgements

This work was supported by the Office of Naval Research, Cognitive Science Program, under grant numbers N00014-10-1-0152 and N00014-13-1-0358, and the U. S. Air Force 711 HW Chief Scientist Seedling program.

References

Assmann, P.F., & Summerfield, Q. (1990). Modeling the perception of concurrent vowels: Vowels with different fundamental frequencies. *J. Acoust. Soc. Am.* 88, 680–697.

Bolia, R., Nelson, W., Ericson, M., and Simpson, B. (2000). A speech corpus for multitalker communications research. *J. Acoust. Soc. Am.* 107, 1065–1066.

Bregman, A. S. (1990). *Auditory scene analysis: the perceptual organization of sound.* Cambridge, MA: MIT Press.

Brungart, D.S. (2001). Informational and energetic masking effects in the perception of two simultaneous talkers. *J. Acoust. Soc. Am.* 109, 1101-1109.

Cherry, E. Colin (1953). Some Experiments on the Recognition of Speech, with One and with Two Ears. *J. Acoust. Soc. Am.*. 25 (5): 975–79

Kieras, D.E. (in press). A summary of the EPIC Cognitive Architecture. In S. Chipman (Ed.), *The Oxford Handbook of Cognitive Science*.

Kieras, D.E., Wakefield, G.H., Thompson, E., Iyer, N., and Simpson, B.D. (2014). A cognitive-architectural account of two-channel speech processing. In *Proceedings of the 2014 International Annual Meeting of the Human Factors and Ergonomics Society*, Chicago, October 27-31, 2014.

Meyer, D. E., & Kieras, D. E. (1997). A computational theory of executive cognitive processes and multiple-task performance: Part 1. Basic mechanisms. *Psychological Review*, 104, 3-65.

Meyer, D. E., & Kieras, D. E. (1999). Precis to a practical unified theory of cognition and action: Some lessons from computational modeling of human multiple-task performance. In D. Gopher & A. Koriat (Eds.), *Attention and Performance XVII*.(pp. 15-88) Cambridge, MA: M.I.T. Press.

Thompson, E.R, Iyer, N., Simpson, B.D., Wakefield, G.H., Kieras, D.E., & Brungart, D.S. (submitted). Payoff matrices and optimal listener strategies in speech-on-speech masking. Submitted to *J. Acoust. Soc. Am.*

Wichman, F.A., & Hill, N.J. (2001). The psychometric function: I. Fitting, sampling, and goodness of fit. *Perception & Psychophysics*, 2001, 63(8), 1293-1313.

How does prevalence shape errors in complex tasks?

Enkhbold Nyamsuren (e.nyamsuren@uva.nl)
Han L.J. van der Maas (h.l.j.vandermaas@uva.nl)
Department of Psychology, University of Amsterdam
Weesperplein 4, 1018 XA Amsterdam, Netherlands

Niels A. Taatgen (n.a.taatgen@rug.nl)
Department of Artificial Intelligence, University of Groningen,
Nijenborgh 9, 9747 AG Groningen, Netherlands

Abstract

This study shows that cause and types of errors in complex problem-solving tasks can be explained within a framework of the prevalence effect commonly studied only in simple visual search tasks. The explanation proposes that subjects make a series of probabilistic decisions aimed at balancing both speed and accuracy. Such decision is a complex process that relies not only on task instructions but also on cognitive biases established by the history of previous trials and progress of the current trial. We provide evidence based on both empirical data and cognitive modeling.

Keywords: problem-solving, cause of errors, prevalence, ACT-R

Introduction

Why and how do people make mistakes in complex problem-solving tasks? What are the primary cognitive mechanisms? We try to answer these questions using a computerized version of a board game SET[1]. Compared to typical laboratory tasks, SET is a more complex task requiring implicit and explicit strategies, coordination of bottom-up perceptual and top-down executive processes, making consecutive decisions and accumulation of evidence along several dimensions. Any of these components can be a source of errors. Despite a number of preceding studies focused on SET (Jacob & Hochstein, 2008; Mackey, Hill, Stone, & Bunge, 2011; Nyamsuren & Taatgen, 2013), none of them looked at the source of errors. However, the nature of errors can tell us a lot more about the process of problem solving than just the response times and accuracies. We employ a combination of empirical study based on Math Garden and cognitive modeling to tackle this problem. Math Garden (Klinkenberg, Straatemeier, & Van der Maas, 2011) is a web-based computer adaptive practice and monitoring system used by more than 2000 schools to train students' cognitive skills with serious games such as SET.

A SET trial starts with a number of cards dealt open (Figure 1). Each card is uniquely defined by a combination of four attributes: color, shape, shading and the number of shapes. Each attribute can have one of three distinct values. The goal is to find a unique combination of three cards, called a *set*, where values of each attribute are all same or all different. We refer to the number of different attributes in a set as the *set level*. For example, in Figure 1, a level 2 set is formed by three yellow cards. It has two same (color and shape) and two different (shading and number) attributes. In a level 4 set, all values of all attributes are different.

Jacob and Hochstein (2008) proposed that SET players use a dimension reduction strategy. They prefer to search for a set among cards that have the same attribute value thus effectively reducing the search space by one attribute dimension. For example, a subject may look for a set among cards of the same color. A later study (Nyamsuren, & Taatgen, 2013) confirmed Jacob and Hochstein's theory. Nyamsuren and Taatgen also found that dimension reduction is mostly used early in a trial. If dimension reduction strategy fails to find a set, subjects start searching for more dissimilar sets.

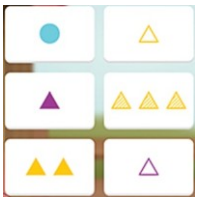


Figure 1: An example of a trial used in Math Garden. A level 2 set is formed by the yellow cards.

Experimental Results

The data was gathered in Math Garden between April 2014 and October 2014. It included 1374530 trials of 80 items (20 items per set level) played by 86964 subjects. Each item consisted of six cards and had exactly one set (e.g. Figure 1). A trial was terminated after a subject selected any combination of three cards. There was 30-seconds time limit per trial. Above sample does not include overtime trials or trials without proper responses (a subject can give up on a trial and request to shown an answer).

Accuracy and Response Time

The average accuracy[2] is around 70%. In 30% of the trials, subjects responded with wrong combinations of three cards (further referred as *triplets*). First, we study cause of errors

[1] SET is a game by Set Enterprises (http://www.setgame.com)

[2] Math Garden dynamically adjusts difficulty to maintain a 75% success rate. Therefore, relative accuracy is uninformative.

by analyzing response times (RT).

Confirmed by a linear regression carried out on trials' mean RT, Figure 2a shows that response times increase with set level for both correct and incorrect trials (β = 2.05, $t(156)$ = 20.2, p < .01). In correct trials, the increase is caused by two factors (Nyamsuren & Taatgen, 2013). Firstly, subjects tend to start a trial with search for a lower-level set and, if the set was not found, switch to search for higher-level sets. Secondly, it requires more effort to compare dissimilar attributes than similar attributes. It is likely that the same two factors are also responsible for RT increase in incorrect trials. Mean RT for correct trials is lower than mean RT for incorrect trials (β = -1.17, $t(156)$ = -4.2, p < .01). However, this difference in RT decreases as the set level increases (β = .35, $t(156)$ = 3.5, p < .01). Note that, for items with level 4 sets, mean RT for incorrect trials is lower than mean RT for correct trials.

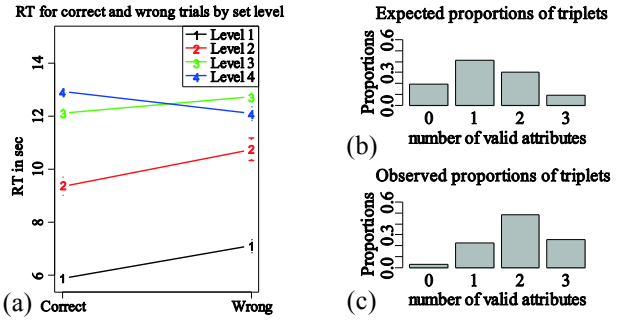


Figure 2: (a) Response times for correct and incorrect trials averaged by set level; Distributions of proportions of triplets by the number of valid attributes calculated from (b) all possible combinations of triplets in 80 items and (c) triplets provided as response by subjects.

Errors Based on Types of Triplets

Previous studies showed that perceptual aspects of SET have significant influence on subjects' decisions (e.g. Nyamsuren & Taatgen, 2013). Similarly, error types in SET may be affected by perceptual components of the task. In this section, we explore whether properties of a triplet defined by its combinations of attribute values affect subjects' decisions and error patterns.

Subsequent analyses concern incorrect trials where subjects responded with wrong triplets. Figure 2b shows a distribution of proportions of triplets by the number of valid attributes in a triplet. An attribute is valid if it follows the set rule and thus is either the same or different in all cards of the triplet. These proportions are calculated from all possible non-repeating combinations of triplets in all 80 items. They serve as a baseline. Figure 2c shows the same distribution, but with proportions calculated from wrong triplets provided as responses. According to Figure 2c, triplets with 2 or 3 valid attributes have significantly above chance probability of being chosen as a set. In other words, errors made by subjects are systematic and not random. More set-like triplets with higher number of valid attributes have higher probability of being incorrectly chosen as a set.

More importantly, there is a negative correlation between the number of valid attributes and RT. Errors with triplets with more valid attributes are made sooner than errors with triplets with fewer valid attributes. According to a linear regression analysis, RT decreases by 188 ms with each valid attribute in a triplet ($t(1518)$ = -2.26, p = .024).

Errors Based on Sameness and Difference

The previous section showed that the number of valid attributes in a triplet could have a significant impact on subjects' decisions. However, a valid attribute can be either same or different among three cards of the triplet. We found that sameness or difference of an attribute plays a substantial role in subjects' decisions.

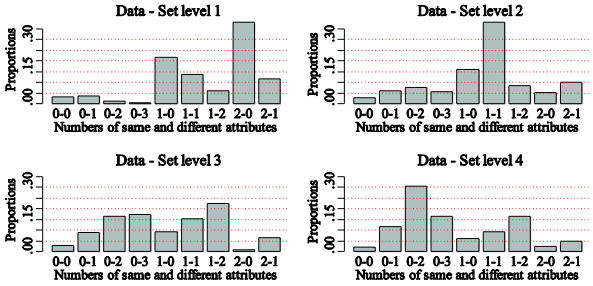


Figure 3: Distributions of proportions of errors types in trials categorized by set level. A wrong triplet consisting of three cards on the left of Figure 1 would give a 1-1 error type, since this answer contains one valid same attribute (shading) and one valid different attribute (color).

The following analysis concerns incorrect trials where subjects responded with wrong triplets. Figure 3 shows distributions of proportions of triplets with specific combinations of same and different valid attributes. The proportions were calculated separately for groups of trials of the same set levels. In trials with level 1 sets, most errors are made with triplets that had same valid attributes. For example, about 35% of all errors in level 1 trials involved triplets with two valid same attributes and no valid different attributes. The effect is completely opposite in trials with level 4 sets. In those trials, the most frequent errors involve triplets with different valid attributes. In fact, the gradual shift from sameness to difference can be observed in the distributions of proportions as set level increases. For levels 1 to 4, mean numbers of same attributes in wrong triplets are 1.4, 0.93, 0.61 and 0.46 against expected 0.67, 0.41, 0.33 and 0.21 if triplets were chosen randomly. Similar above chance preference was observed for different attributes in higher-level sets. Therefore, this shift likely represents a systematic shift in criterion against which subjects evaluate validity of attribute combinations.

Cause of Errors

An explanation of errors in SET can be derived from the prevalence effect. It is frequently observed in visual search

tasks where a target can be either present or absent. In low-prevalence conditions, subjects miss the target more often than in high-prevalence conditions (Wolfe, Horowitz, Van Wert, Kenner, Place, & Kibbi, 2007). Subjects do not explicitly try to speed up their responses using some time threshold. Instead, they adjust their internally estimated probability of a target being absent based on the sequence of previous trials (Ishibashi, Kita & Wolfe, 2012). This probability affects the decision on whether an object is a target or a distractor and the decision to quit the trial.

Within-trial Prevalence

With a proposal of within-trial prevalence, we assume that subject's internally estimated probability of finding a set changes during the progression of a trial. Subjects are aware that there is always one set present in each trial. Therefore, although probability of finding a set at the start of a trial is very low, it increases as a subject continues search and discards more distractor triplets. Wolfe and Van Wert (2010) proposed that the prevalence effect can be modeled via a drift diffusion model where decision is made when an evidence accumulation reaches a certain threshold. Similarly, we propose that subjects pick a triplet as a set when the accumulating probability reaches some threshold. During the trial, each discarded triplet increases probability of the next triplet being a set (green lines in Figure 4).

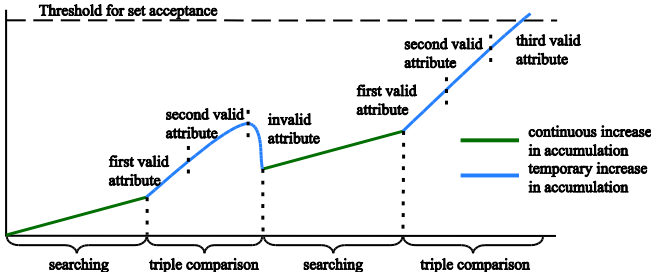


Figure 4: An evidence accumulation account of a within-trial prevalence in SET.

We know that the more set-like triplets are, the more likely they are to be chosen as a set. The effect can be explained with an assumption that an increasing similarity of a triplet to a set results in a temporary increase in within-trial probability. This process can be viewed as a large but temporary step-wise increase in accumulation caused by each new validated attribute (blue lines in Figure 4). Higher number of valid attributes will result in a larger increase in accumulation and a higher probability of exceeding the acceptance threshold. However, an attribute with invalid combination may negate the local boost in probability and results in the triplet being discarded as a potential set.

Even when a set is found early, the temporary increase in probability caused by four valid attributes is normally sufficient to exceed the threshold. On the other hand, in late trials, wrong triplets with few valid attributes will have a higher chance of exceeding the threshold due to constantly increasing probability. This process will result in incorrect trials having higher RT than correct trials (Figure 2a).

Finally, triplets with higher number of valid attributes may exceed the threshold sooner than triplets with fewer valid attributes explaining the negative correlation between RT and the number of valid attributes observed in the data.

Between-trial Prevalence

The prevalence effect also provides a framework for explaining why subjects shift from sameness to difference when validating attribute combinations. Here, changing prevalence of trials with particular set levels is a likely cause for such criterion shift. The adaptive algorithm in Math Garden ensures that next trial's difficulty is tailored to subject's skills. Therefore, new subjects start with easy trials with level 1 sets and are gradually introduced to more difficult trials. As a result, trials with level 1 and 4 sets initially have high and low prevalence respectively. However, as subjects gain more experience, prevalence of trials with level 1 and level 4 sets decreases and increases respectively causing subjects to shift their set acceptance criterion from similarity to dissimilarity. Based on data of 432 subjects who played at least 100 trials, proportions of trials with set levels 1 and 4 in first 25 trials are 63% and 7% respectively. For the fourth bin of 25 trials, the same proportions change to 18% and 28% respectively.

In terms of evidence accumulation account shown in Figure 4, different and same valid attributes make different contributions to the temporary increases in accumulation. In trials with level 1 sets, valid different attributes may not cause temporary increase in accumulation or may even have inhibitory effect on accumulation. However, as a subject is exposed more to trials with higher set levels, contributions of valid different attribute may gradually increase.

Threshold

The fact that the RT increases with set level indicates that the threshold is not the same among trials with different set levels. It is likely that subjects dynamically adjust their threshold whenever it is too low or too high, as in other visual search tasks. Chun and Wolfe (1996) showed that subjects' RT in target absent-present visual search tasks can be reproduced with a model using a dynamic threshold adjusted in a staircase manner. It was further suggested that RT in low- and high-prevalence search tasks can be modeled via adjustment of a quitting threshold (Wolfe & Van Wert, 2010). In a more recent visual foraging study, subjects adjusted in a staircase manner their probability of remaining on a patch depending on whether an instance of foraging was successful or not (Wolfe, 2013).

We draw an analogy from above examples and propose that subjects in SET are also adjusting set acceptance threshold in a staircase manner based on the result of the previous trial. After making a mistake, a subject may become more conservative and increase set acceptance threshold. The opposite will happen after a correct trial where the subject accepts a more liberal approach by

lowering threshold.

Cognitive Model

A cognitive model was used to formally test validity of the processes proposed in the preceding section. We have reused a model of a SET player developed in our earlier work. Due to space limit, we will describe only essential details of the model. The reader is referred to previous literature for a detailed description of the model (Nyamsuren & Taatgen, 2013). The model is based on ACT-R cognitive architecture (Anderson, 2007) that simulates functionality of essential cognitive resources such as declarative memory, working memory, the visual system and the production system. Within a model, task-related instructions are implemented as a set of production rules.

The overall strategy used by the model is simple. The model chooses a triplet and compares validity of four attributes one by one in a random order. This is done by having a production rule named '*compare*' repeatedly being called for each attribute. Only when all four attributes form valid combinations, the model chooses the triplet as a set ending a trial. When any attribute yields an invalid combination, the triplet is discarded and a new triplet is chosen. At the beginning of the trial, the model prefers triplets of cards having, at least, one common value (e.g. all green cards). Later in the trial, the model switches to triplets with cards that are more dissimilar.

The original model did not make mistakes. We have extended the model by implementing error-making mechanisms described in the preceding section. The next section describes those extensions.

Production Competition as a Cause of Errors

The original model took a conservative approach to set acceptance ensuring that all four attributes were valid in a triplet. The modified model adopts a more liberal approach and can accept a triplet as a set without validating all attributes. This is done by introducing a new production rule named '*valid-set*' that competes with the production rule '*compare*'. This process is shown in Figure 5. Given a triplet, the model can either validate an attribute in the triplet by calling '*compare*' production or accept the triplet as a set by calling '*valid-set*' production. A production rule with the highest utility value *U* is chosen.

Utility of '*valid-set*', $U(V)$, represents the accumulation shown in Figure 4 and indicates a probability of a triplet being a set. $U(V)$ is zero at a start of a trial but increases as the trial progresses according to $U(V) = 1/(T_t - T_c)$. T_t is the total number of unique triplets formed by six cards and equal to 20. T_c is the number of compared triplets, and $(T_t - T_c)$ is the number of remaining uncompared triplets.

$U(V)$ can temporary increase based on the number of validated attributes in a triplet. Given k validated attributes, probability of a set is $U(V) = 1/[(T_t - T_c) * P(k)]$, where $P(k)$ is a proportion of triplets that have k or more valid attributes. For each newly compared triplet, k is set to 0 but increases with each validated attribute in the triplet. It is unlikely that subjects can estimate $P(k)$ within each trial. However, it is probable that subject may able to learn prevalence of triplets with different values of k over many trials. Calculated from all unique triplets from all items, the proportions are $P(1) = .8$, $P(2) = .4$, $P(3) = .11$, and $P(4) = 1/(T_t - T_c)$ since there is only one set. Therefore, $U(V)$ increases with increasing k and is equal to 1 for $k = 4$ simulating the temporary increase in accumulation shown in Figure 4. Above proportions decrease as T_c increases.

$$T_{m_{i+1}} = \begin{cases} T_{m_i} - 6, & trial_i \text{ is correct} \\ T_{m_i} + 4, & trial_i \text{ is incorrect} \\ 12, & trial_i \text{ is overtime} \end{cases} \quad \text{(Eq. 1)}$$

The utility of '*compare*' production, $U(C)$, represents a threshold for set acceptance. The threshold for *i*-th trial is calculated as $U(C)_i = 1/(T_t - T_{m_i})$, where T_{m_i} is the minimum number of triplets to be compared in the trial. $U(C)$ remains the same during a trial, but T_m decreases or increases between trials according to the Eq. 1. For the next trial, $U(C)$ increases if the model makes a mistake by responding with a wrong triplet. If set is found, $U(C)$ decreases. If utility is too high and the model cannot find a set within a time limit then T_m is reset to 12, the minimum allowed value. This minimum value is set based on the assumption that subjects always have to perform some search. Numerical constants were fitted based on model simulations.

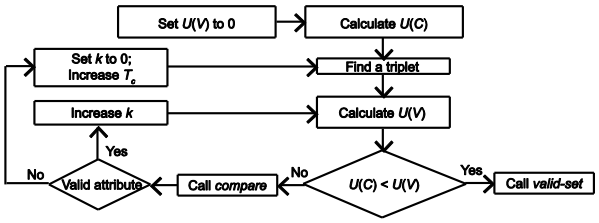


Figure 5: The competition between '*valid-set*' and '*compare*' productions is a cause of errors in the model.

Bias to similarity

Bias to similarity can occur at least at two decision points. First, bias can affect a choice of strategy. To replicate the effect of prevalence of trials with different set levels, the model was modified to be highly biased to the dimension reduction strategy while playing items with set level 1. However, this bias decreases with increasing set level following the decrease in observed proportions of trials with set level 1. Therefore, while playing items with set level 4, the model is more likely to use dissimilarity-based strategy.

Second, similarity bias can affect a decision whether a triplet is a set with subjects giving an initially higher weight to valid same attributes than to different attributes. This bias is simulated using two weights W_s and W_d that affect calculation of the value k: $k = W_s k_s + W_d k_d$. k_s and k_d are numbers of validated same and different attribute respectively. If the model is using the dimension reduction strategy then W_s and W_d are equal to 0.5 and -0.5 respectively. Otherwise, W_s and W_d are equal to -0.5 and 0.5

if the model is using a dissimilarity-based strategy.

Those changing weights represent shift in criterion for accepting a triplet as a set. More specifically, weights coupled with decreasing bias to the dimension reduction strategy simulate in the model a shift in set acceptance criterion from similarity to dissimilarity.

Simulation Results

80 items were divided into 10 blocks. Each block contained items of the same set level and same distance between set cards. The model was tested on 10000 trials in each block. The '*compare*' production had the minimum utility at the first trial of a block but was adjusted between trials of the same block.

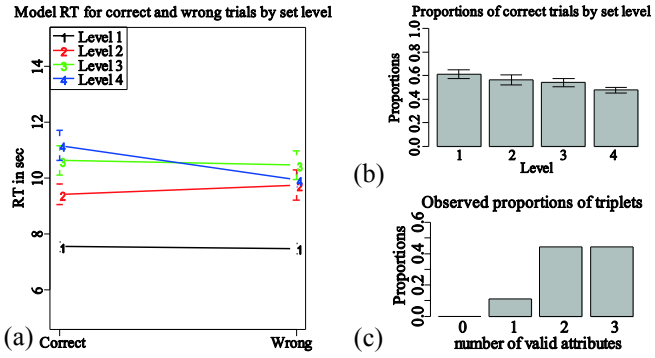


Figure 6: (a) Model's response time for correct and wrong trials of different set levels. (b) Model's accuracy by set level. (c) Proportions by the number of valid attributes of wrong triplets selected by the model.

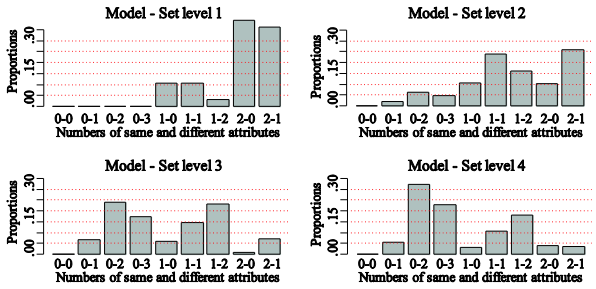


Figure 7: Distributions of proportions of errors made by the model in trials categorized by set level.

Figure 6a shows RT produced by the model. Similar to experimental data, model's RT increased as set level increased (β = 1.0, $t(156)$ = 7.5, p < .01). The main predictor for accuracy indicates that RT should be lower for correct trials (β = -.36, $t(156)$ = -.97, p = .3) with the interaction term indicating that this difference should decrease as set level increased (β = .2, $t(156)$ = -1.5, p = .14). However, both the main and interaction terms were not significant. The model was not able to simulate decreasing difference in RT between correct and incorrect trials observed in the empirical data (Figure 2a). Figure 6b shows model accuracy. The model predicts that the accuracy should decrease with set level. Most likely, this trend was not observed in subject data (Figure 2a) because Math Garden maintains 0.75 probability success by matching trial's difficulty to subject's skills. Figure 6c shows types of model-made errors defined by the number of valid attributes. Similar to subjects (Figure 2c), the model is more likely to make errors with triplets that have high number of two or three valid attributes. The increasing number of valid attributes also results in lower RT for incorrect trials (β = -1.1, $t(844)$ = -4.3, p < .01). However, the decrease of 1.1 seconds per valid attribute is much higher than 0.188 seconds observed in subjects' data.

Figure 7 shows distributions of proportions of incorrectly selected triplets with specific combinations of same and different valid attributes. The model's data closely resembles the empirical data shown in Figure 3. Correlations between the proportions in the empirical data and model data are $r(7)$ = .86, p < .01 for level 1, $r(7)$ = .84, p < .01 for level 2, $r(7)$ = .97, p < .01 for level 3, and $r(7)$ = .99, p < .01 for level 4. The model shows the same shift in criterion from similarity to dissimilarity in its decision of a triplet being a set. Overall, model simulations support our hypothesis that the prevalence effect, internally estimated probability of finding a set, can be a cause of errors.

Discussion and Conclusion

Speed-accuracy trade-off (Wickelgren, 1977) may also contribute to errors since Math Garden pressures subjects to finish a trial both quickly and accurately. However, it is unlikely to be the sole or even the main cause of errors in SET. First, the negative correlation between the number of valid attributes and RT is not easily explained by speeded responses. Second, a critical assumption behind the speed-accuracy trade-off is that errors should disappear if subjects are discouraged from giving fast responses. However, the reward system used in Math Garden severely punishes for fast incorrect responses making it more profitable to make slow correct responses. Therefore, the ideal strategy is either to give a correct response or let the time run out. The fact that subjects make early errors (Figure 2a) despite discouragement of fast responses violates the assumption behind the speed-accuracy trade-off. Wolfe at al. (2007) also explicitly differentiated the prevalence effect from speed-accuracy trade-off and showed that people resort to probabilistic decision making even in absence of a time pressure. The prevalence-based explanation assumes that that estimated probability causes changes in both RT and accuracy (Wolfe & Van Wert, 2010). Therefore, manipulations based on time should have little effect on estimated probability and, therefore, on accuracy explaining why subjects still made early errors in SET despite strong discouragement in Math Garden.

Therefore, a general question that has not been addressed in other studies is why a probabilistic decision is made despite an opportunity to verify their answers. We propose that it is due to an inherent nature of a human cognition to

pursue efficiency. Efficiency is achieved by minimizing the amount of cognitive effort to accomplish the task while still maintaining a reasonably high degree of success. This efficiency optimization is different from the common definition of optimization aimed at finding the optimal solution. Instead, in cognitive literature, efficient strategy is often referred to as heuristic (Gigerenzer & Brighton, 2009). Heuristics are simple strategies that do not guarantee absolute success rate but work most of the time. A necessity for heuristics is dictated by the framework of bounded rationality (Simon, 1972). It assumes that cognitive resources are limited and, therefore, processes utilizing the least amount of resources are favored even at the expense of accuracy. Note that a time pressure is not a required component for a formation or use of heuristics.

Above discussion suggests that the prevalence effect is a general phenomenon beyond simple visual search tasks commonly used to study the effect. Our study shows that it may play an important role in complex problem-solving tasks. For example, a similar effect is commonly observed in causal reasoning tasks. Griffiths, Sobel, Tenenbaum and Gopnik (2011) showed that subjects internally estimate Bayesian-like probabilities to judge causal relations between an effect and two possible causes. Although subjects were aware that it is possible for both options to independently cause the effect, their judgments were highly correlated with frequencies of both options causing the effect. The more prevalent option was not only likely to be classified as a cause but also decreased the probability of positive classification for the second option despite the independence of causes. Therefore, decision-making in causal task is not just frequency-based but a probabilistic process that incorporates frequency information.

Wolfe and Van Wert (2010) originally proposed that target prevalence in visual search can be modeled with a drift diffusion model with a changing starting point. Indeed, triplet comparison processes in Figure 4 can be viewed as a sequence of drift diffusion models where a consecutive model has a higher starting point than the previous one. However, Wolfe and Van Wert assumed that the starting point can only change between trials and not within trials. Our study shows that, in a complex task requiring several decisions, the starting point can and should change within a trial if there is a high expectation that the target is present. During the progression of visual search the estimated probability of finding a target should increase. This leads to our assumption that prevalence is not only a between-trial effect, but also can be observed within a trial in complex tasks such as SET.

ACT-R does not provide a suitable and standardized way to model an evidence accumulation process in the procedural system. In this study, we proposed that production rule's utility can change as a function of relevance to a changing context without the production rule being executed. It is not inconsistent with the existing utility learning mechanism, but adds an additional factor that influences utilities. As a proof of concept, the SET model used this mechanism to manipulate utility of a single production ruule during a trial to replicate a human behavior conventionally modeled with accumulation models. While we argue for the necessity for such mechanism, more studies are required for its implementation that is well integrated into ACT-R both theoretically and technically. Experimental data and the cognitive model can be downloaded from http://www.bcogs.net/models/

References

Anderson, J. R. (2007). *How can human mind occur in the physical universe?* New York: Oxford University Press.

Chun, M. M., & Wolfe, J. M. (1996). Just say no: How are visual searches terminated when there is no target present? *Cognitive Psychology*, 30 (1), 39-78.

Gigerenzer, G., & Brighton, H. (2009). Homo heuristicus: Why biased minds make better inferences. *Topics in Cognitive Science*, 1 (1), 107-143.

Griffiths, T. L., Sobel, D. M., Tenenbaum, J. B., & Gopnik, A. (2011). Bayes and Blickets: Effects of Knowledge on Causal Induction in Children and Adults. *Cognitive Science*, 35, 1407–1455.

Ishibashi, K., Kita, S., & Wolfe, J. M. (2012). The effects of local prevalence and explicit expectations on search termination times. *Attention, Perception, & Psychophysics*, 74 (1), 115-123.

Jacob, M., & Hochstein, S. (2008). Set recognition as a window to perceptual and cognitive processes. *Perception & Psychophysics*, 70 (7), 1165-1184.

Klinkenberg, S., Straatemeier, M., & Van der Maas, H. L. J. (2011). Computer adaptive practice of maths ability using a new item response model for on the fly ability and difficulty estimation. *Computers & Education*, 57 (2), 1813-1824.

Mackey, A. P., Hill, S. S., Stone, S. I., & Bunge, S. A. (2011). Differential effects of reasoning and speed training in children. *Developmental Science*, 14 (3), 582-590.

Nyamsuren, E., & Taatgen, N.A. (2013). Set as instance of a real-world visual-cognitive task. *Cognitive Science*, 37 (1), 146-175.

Simon, H. A. (1972). Theories of bounded rationality. Decision and organization, 1 (1), 161-176.

Wickelgren, W. A. (1977). Speed-accuracy tradeoff and information processing dynamics. *Acta Psychologica*, 41 (1), 67-85.

Wolfe, J. M. (2013). When is it time to move to the next raspberry bush? Foraging rules in human visual search. *Journal of Vision*, 13(3), 10.

Wolfe, J. M., Horowitz, T. S., Van Wert, M. J., Kenner, N. M., Place, S. S., & Kibbi, N. (2007). Low target prevalence is a stubborn source of errors in visual search tasks. *Journal of Experimental Psychology: General*, 136 (4), 623.

Wolfe, J. M., & Van Wert, M. J. (2010). Varying target prevalence reveals two dissociable decision criteria in visual search. *Current Biology*, 20 (2), 121-124.

When and Why Does Visual Working Memory Capacity Depend on the Number of Visual Features Stored: An Explanation in Terms of an Oscillatory Model

Krzysztof Andrelczyk (kandrelczyk@gmail.com)
Institute of Psychology, Jagiellonian University
Ingardena 6, 31-120 Krakow, Poland

Adam Chuderski (adam.chuderski@uj.edu.pl)
Institute of Philosophy, Jagiellonian University
Grodzka 52, 31-044 Krakow, Poland

Tomasz Smolen (tsmolen@up.krakow.pl)
Pedagogical University
Podchorazych 2, 30-084 Krakow, Poland

Abstract

The nature of capacity limits within human visual working memory (VWM) remains the subject of controversy: while the capacity-as-objects account predicts that what loads VWM capacity is solely the number of objects maintained, irrespectively of the number of visual features that need to be stored for each object, the capacity-as-features account predicts that also (or – primarily) the total number of features maintained in VWM loads its capacity (and leads to decreased performance). We present novel simulations of a VWM task, using our existing, oscillatory computational model that describes the binding of features into objects as resulting from the proper synchronization and desynchronization of rhythmical changes in neuronal activity. The model predicted (in line with wide evidence) that VWM performance decreases with the increasing number of objects, but also decreases (although not as sharply as predicted by the capacity-as-features account) as a function of increasing number of features. The model attempts to explain what precise characteristics of oscillatory dynamics stand behind such two sources of VWM limitation. However, the complete pattern of the model's predictions remains yet to be examined empirically.

Introduction

Working memory (WM) is a neurocognitive mechanism responsible for the active maintenance of information as well as its manipulation for the purpose of the current task. Although early research on WM was dominated by verbal paradigms and models, for the last 15 years some researchers (e.g., Luck & Vogel, 1997) have pointed at the crucial role of, relatively simpler than verbal WM, visual working memory (VWM; also called robust visual short-term memory store) in subserving functions of temporary storage, binding, and manipulation of information. VWM operates on visuospatial representations, usually called objects, that are widely thought to consist of bindings of the corresponding visuospatial features (like shape, color, orientation, size, or location). Although simple, during the evolution of the human mind this mechanism, primarily responsible for the continuity of perception as well as the spatial orientation, most probably has been adapted in service of more complex cognition, including the construction of abstract representations (see Cowan et al., 2011), encoding and processing relations (Clevenger & Hummel, 2014), as well as running mental models and simulations (hypothetical models of the world, in abstraction from its actual state; Johnson-Laird, 2006). This hypothesis is supported by the fact that WM (and VWM in particular) is the strongest known predictor of fluid intelligence – the crucial ability to solve new, complex problems, that is central to human cognitive ability (McGrew, 2009). It has been shown that VWM capacity, measured by the number of recalled or recognized visual representations, explains up to half of variance in fluid intelligence (Fukuda, Vogel, Mayr, Awh, 2010), what suggests a key role of VWM in reasoning, but also in other types of complex cognitive processing, like problem solving, spatial navigation, language use, complex learning, and decision making (i.e., those strongly correlated with fluid intelligence).

Research on VWM pertained to such issues as the role of attentional selection/filtering in VWM, the profound influence of global organization of perceptual scene (statistical regularity) on the number of objects that can be retrieved from VWM, as well as the interaction of VWM and long-term memory (for a review see Brady, Konkle, & Alvarez, 2011). Also, a lot is known about neurobiological basis of VWM, including localization of VWM subsystems responsible for maintaining object features (within superior parietal lobule) versus binding complete objects out of those features (within inferior parietal lobule; Xu & Chun, 2009), or the relation between individual capacity observed in people and the patterns of activity of these subsystems (Todd & Marois, 2004).

Indeed, one of the most important features of VWM is that its capacity is heavily limited and inter-individually varied. Usually, the average capacity of VWM equals four objects or even less, and in the population it can vary from two up to six items (Cowan, 2001). However, the major controversy in research on VWM capacity is what exactly is its "currency", that is, which aspect of maintained information limits the number of objects stored and retrieved. Early theories proposed that VWM is limited in its capacity

for storing separate objects, irrespectively of the complexity of particular objects (Luck & Vogel, 1997; for a review see Fukuda, Awh, & Vogel, 2010). The evidence for this stance came from studies in which increasing the number of presented objects drastically decreased performance (usually measured with the so-called change detection paradigm that requires remembering one visual array, as well as its later comparison with either the identical array or an array in which one item has been changed; Cowan, 2001; Luck & Vogel, 1997). At the same time, performance seemed to be robust to increasing number of visual features that had to be encoded and processed (e.g., people detected the change for several simple objects, like squares, that differed only in color with a comparable accuracy as they detected the multiple-feature objects).

However, recently many scholars (e.g., Bays, Catalao, & Husain, 2009; Bays & Husain, 2008; for a review see Brady, Konkle, & Alvarez, 2011) have objected this *capacity-as-objects account*, suggesting that in fact what constrains VWM is the total amount of information in a perceptual scene, and people remember a larger number of perceptually simpler objects than of complex, multiple-feature objects (Alvarez & Cavanagh, 2004). According to *the capacity-as-features account*, the changes imposed in the change detection paradigm are too substantial (e.g., a change from red to blue) in order to reveal the influence of VWM load on the precision of maintained representations, which however does appear when participants are asked to reproduce exact values of remembered visual features (i.e., "was it light red, dark red, or pink?"). The methodological and interpretational issues concerning new paradigms for VWM capacity measurement remain controversial (for opposing views see Brady et al., 2011, and Fukuda et al., 2010). However, a recent study (Oberauer & Eichenberger, 2013) has shown that even in the standard change detection paradigm (but probably under better experimental control than in previous studies), the VWM performance gradually dropped when three objects were presented with one, three, or as much as four or six visual features (picked up from the following features: color, shape, orientation, size, the thickness of bars inside, and the frequency of stripes inside.

Also other accounts exist, for example a recent view (Clevenger & Hummel, 2014) which suggested that the "objects" of VWM are neither complete objects or feature-object bindings, but the pairs of objects. According to this *capacity-as-paired-relations* account, the capacity of VWM is constrained by the number of representations of pairs of objects, bound with all spatial relations between these objects, and encoded in parallel.

Apart from behavioral experimentation, another way to understand the nature of VWM is to develop (and test) process models of VWM that show in simulations which stages or characteristics of visual information processing are the most vulnerable to WM load, and why VWM capacity has to be limited (because of certain processing demands). The most influential line of such models consists of *oscillatory models* (e.g., Horn & Usher, 1992; Raffone & Wolters, 2001; Usher, Cohen, Haarmann, & Horn, 2001; Vogel, Woodman, & Luck, 2001) that describe the binding of features into objects as resulting from the proper synchronization and desynchronization of rhythmical changes in neuronal activity, called *brain oscillations.* Binding of features into an object is made with synchronous oscillations, whereas the maintenance of separable objects – with asynchronous oscillations. Features of the same item fire in synchrony, whereas two features of different objects are active out of phase. Due to this mechanism, the system is able to reconstruct the object from its features. Such an oscillatory nature of VWM has recently been demonstrated in both primates (e.g., Siegel, Ward, & Miller, 2009) and humans (Kaminski, Brzezicka, & Wrobel, 2011).

The goal of the present study is to analyze how the modified version of one such model of VWM, originally proposed by Chuderski, Andrelczyk, and Smolen (2013), handles different numbers of objects and features in its memory, and to identify what characteristics of a dynamic, oscillatory system implemented in the model may be responsible for its limited capacity to store either objects or features, or some product of objects and features. We start with a concise description of the above mentioned model.

An oscillatory computational model of visual working memory

The main part of the model consists of a buffer, which contains a certain number of elements. Each element roughly approximates a neuronal assembly representing one specific feature of the world. Like in many other models (e.g., Horn & Usher, 1992; Raffone & Wolters, 2001), a level of internal activation x_i is assigned to each element i. The following equation defines the change in the level of activation x of ith element from time t to time t+1:

$$x_{it+1} = x_{it} + \frac{\omega}{1 + \frac{1}{e^{-x_{it}-0.5}}} + \alpha \sum_{|x_{jt}-x_{it}|\le\delta} e^{x_{jt}-x_{it}} - \beta \sum_{|x_{kt}-x_{it}|>\delta} e^{x_{kt}-x_{it}} + \varepsilon(n)$$

This equation consists of five components. The first component is simply the activity of element i in the preceding cycle. The second component represents the self-recurrent increase of the activation of element i, reflecting the reverberatory nature of neuronal assemblies constituting WM. The output of i is fed back to i in order to increase i's activity. The parameter ω impacts the frequency of oscillations given a particular time scale, but has no significant influence on the model's capacity. The output of the element i in time t has been defined using a commonly applied sigmoid function of x_i.

The third component of the formula represents the *coactivation* of i by the mean activity of elements j that fire in close proximity with i, that is, whose activity x_j falls within $[x_i - \delta, x_i + \delta]$. Such a range denotes the activity horizon in which all oscillating elements are treated as

forming one *binding* that integrates features into a particular object, an object with its context, or a role-argument pair. This component accounts for the known fact that neurons that all fire in synchrony with a given neuron more strongly influence its potential as well as synaptic connections than when they fire out of phase. The less active element *j* is, the less it can coactivate *i*. Moderate values of δ have no negative influence on the model's capacity, unless δ is either too small (δ < 0.01) or too large (δ > 0.1), that is, when the bound elements are either highly prone to random changes in activation (i.e., they easily fall apart) or there is no place in the activity space to add new distinct items (i.e., $x_i + \delta$ approaches *x* of another, more activated item), respectively. Parameter α regulates the amount of coactivation that is spread from *j* to *i*. Initial computational simulations showed that it has a limited influence on the model's capacity, as its low value (α = 0.0004) used in the present simulation yielded quite similar capacity (5.1 two-element items) as (5.5 items) its optimal value (α = 0.0012; no further gains in capacity were noted).

The fourth component implements the most important mechanism of the model: lateral inhibition exerted by the bindings that are treated as encoding distinct representations from a representation encoded by elements *i* and *j*; that is, the elements denoted by *k*, which fall outside the range $[x_i - \delta, x_i + \delta]$. The less active element *k* is, the less it inhibits *i*. The larger the activation of an inhibiting binding, the more it suppresses element *i*. Parameter β controls the strength of that inhibition. Previous simulations (see Chuderski et al., 2013) demonstrated that β is the main factor controlling the capacity of our oscillatory model. An increase in β negatively impacted available capacity, as higher values of β made more elements to fall out from VWM (drop permanently below the threshold arbitrarily set to .2).

The last component consists of the noise ε, drawn from the normal distribution with the mean equal to zero, and the variance dependent on the parameter *n*. Large noise (*n*>0.00005) negatively influences capacity, as oscillations of elements become more random; however, its small values (*n*<0.00001) affect the capacity only slightly.

When the output of element *i* surpasses unity (this reflects the strongest possible firing of a neuronal group), the parameter ω for that element is temporarily reversed (reflecting the well-known phenomenon of neuronal hyperpolarization), causing this element to fall quickly below the threshold of activation (0.2), what represents the mechanism of refraction (afterhyperpolarization). When x_i becomes smaller than the threshold, ω value is reset to a previous positive value, and the element starts building up its activation above the base level. However, in certain cases the inhibition signals may be so strong that the activation of *i* is too slow, and just after ω is reversed the activation of *i* decreases permanently below the threshold – the minimal activation necessary to stay in WM. If this happens, element *i* falls out of WM, meaning that it can no longer impact other elements in WM, nor can be recalled (but it may potentially be encoded in the active part of long-term memory). This latter mechanism underlies the function of emptying WM in order to encode incoming information.

Generally, the number of elements which can be bound together within one synchronic oscillation is not limited. However, in Chuderski et al. (2013) only pairs of synchronized elements (an object identity and one its feature) were added to the model's VWM. So, how the model works when an object is composed from more visual features?

Workings of the oscillatory model

The aim of the model is to maintain as many separate oscillations as necessary, for a given time interval. Two elements making one oscillating pair (e.g., an object and the information about its a shape) are added to the buffer in the same time. The pair which is added as the first one is added with a random level of activation. Subsequent pairs can be added when activation of all other pairs is less than the value of $1 - 4 \times \delta$, and those subsequent pairs are added at a level of $x = x_{max} + \delta + (1 - x_{max}) / 2$, where x_{max} denotes the *x* value of the most active pair. So, this mechanism checks if there is enough place in the activation space for new elements, and grants that at least on entering the buffer the new pairs will be sufficiently distinctive from all pairs already maintained.

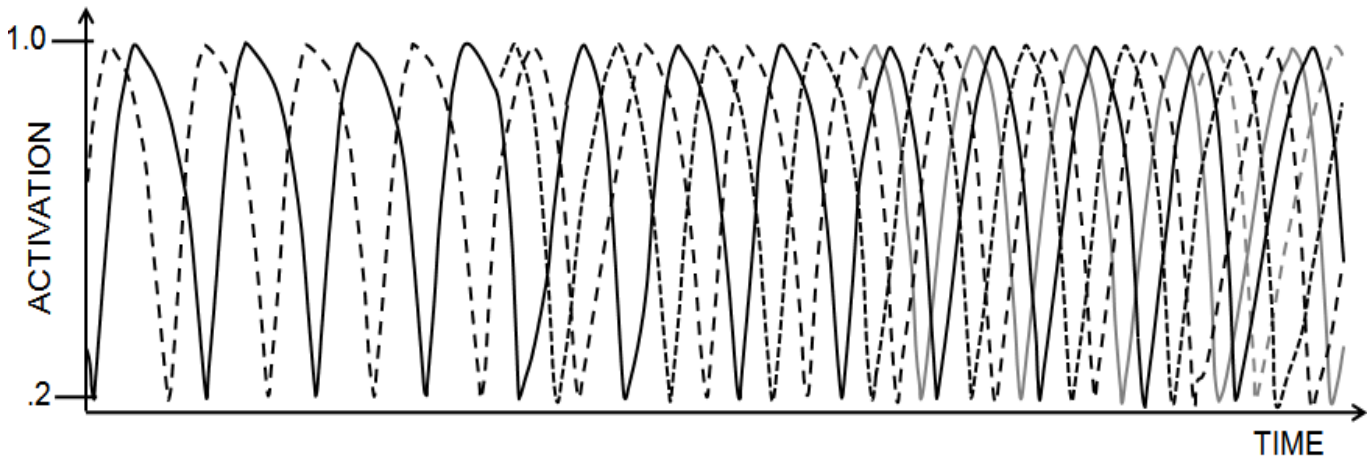


Fig.1: The model dynamics under increased VWM load.

In the model, the capacity limit arises because when the total amount of inhibition in the model is very large, it overcomes the results of activation, and the elements with the lowest activation levels start falling out of the buffer. If one element from the pair falls out, then the coactivation is no longer possible, and the chance that the other element from that pair would also fall out drastically increases. However, a certain amount of inhibition is necessary, because it secures that oscillations will evenly occupy a respective time interval, helping to separate them. So, the values of β reflect the trade-off between low inhibition (many objects can be maintained, but their bindings more easily fall apart) and high inhibition (less objects can be maintained, but their bindings are more robust).

Simulating the effects of objects and features' load on visual working memory capacity

Below, we report simulations in which, apart from the number of objects presented to our oscillatory model in the (simulated) change detection task, we varied the number of features per object. When the arrays in the task changed, than a random number of features were altered in the highlighted target (i.e., from one feature to the maximum number of features possible). Thus, the model had to maintain in its VWM all features of an object, because if it encoded only some features, and the changed feature was among remaining ones (was not encoded), the model would commit an error of omission. We compared the results of our simulations to existing empirical data. The classic study (Vogel et al., 2001; Experiments 11 & 14) yielded a large effect on the change detection accuracy of the number of objects that were required to maintain (either 2 or 4 objects), but no effect of the number of features per object (1 or 4). In contrast, Oberauer and Eichenberger (2013) used eight feature values per each featural dimension, and observed a significant drop in participants' performance when the number of featural dimension increased from one to four. So, can our oscillatory model replicate this data (see Fig.3)?

We simulated the variant of the change detection task similar to the variant applied by Oberauer and Eichenberger, as their results seem to be the most reliable data available. However, unlike their use of only sets of three objects, we manipulated the number of presented objects at three levels: two, three, and four, in order to observe whether any interaction of the numbers of objects and features occurs. First, the model was presented with symbolic descriptions (no perceptual module was modeled) of objects that contained the numerical identity of an object (e.g., Object1) plus one, two, or four values of distinct features. The task of the model was to encode the objects in its oscillatory VWM, and to maintain these objects until the description of the second array arrived. This array could be identical (in half trials) or could differ in a random number of features for exactly one object indicated (in the remaining half of trials). Finally, the model decided if the target object matched the corresponding contents of VWM, or differed. The accuracy of the model's responses was recorded.

In the following simulations, we used exactly the values of parameters fitted to data in the previous simulation (Chuderski et al., 2013). That is, we set parameter α to a value of .0004, parameter ω was drawn from the normal distribution with $\mu = 0.05$, and $\sigma = 0.005$, and parameter δ was set to $\delta = .05$. The only parameter we analyzed, and fitted in the present study, was the value of β (in the original paper the mean value of $\beta = .0026$ was used).

Simulation results

First, we analyzed how parameter β determines the overall accuracy in the current version of the model. We simulated 450 trials in the change detection task per each reasonable value of β (.0010, .0015, .0020, .0025, and .0030), using only the set sizes of three objects, as well as applying the number of features equaling one, two, or four (i.e., the exact values used by Oberauer & Eichenberger in their Exp. 3). The results, shown in Fig. 2, indicated that a moderate level of β between .0015 and .0020 was optimal. At either higher or lower values of β accuracy dropped substantially. The probable cause of the fact that too low inhibition deteriorated VWM maintenance results from stochastic effects that pertain to the activations of oscillated elements. When the model tried to maintain many bindings, low inhibition was not able to optimally separate the consecutive oscillations, the activations within one binding (object) might vary more than was the distance (in the activation space) to subsequent bindings, and some elements might fall into the attractor of another binding. Obviously, a high level of inhibition was not effective either: bindings mutually inhibited themselves so strongly that elements that entered the afterhyperpolarization phase were easily eliminated from the model.

Thus, in the following simulations we adopted the optimal levels of β, picking it up on random from [.0014 – .0021] range. The simulation contained set sizes of two, three, and four objects, which could be the bindings of two elements (an object's numerical identity + the value of one its feature), three elements (an object + two features), and five ones (an object + four features). Thus, for each set size we calculated three data points (i.e., nine data points were obtained with fitting only one parameter). In total, 1800 trials were simulated per each data point.

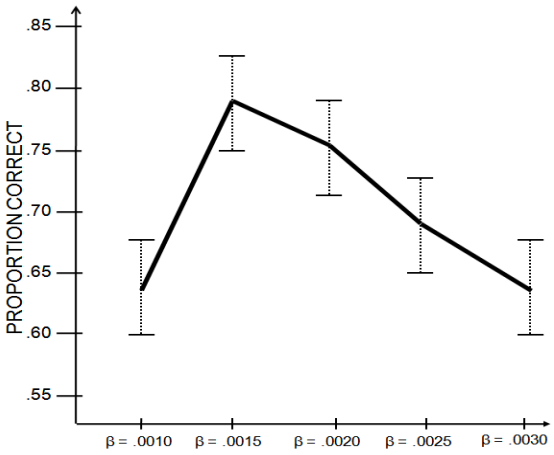


Fig. 2. The average accuracy of the model in the three-object change detection task under varying levels of β.

Not surprisingly, the model replicated the effect of the number of to-be-remembered objects on accuracy, overall matching the usual pattern of empirical data. Regarding the number of features, the results of set size equaling three nicely matched data of Oberauer and Eichenberger (2013): the model aptly mimicked the fact that accuracy decreased when the number of features increased, $F = 92.02$, $p < .001$, however the simulated decrease (19%) was slightly larger than the one observed by Oberauer and Eichenberger (10%).

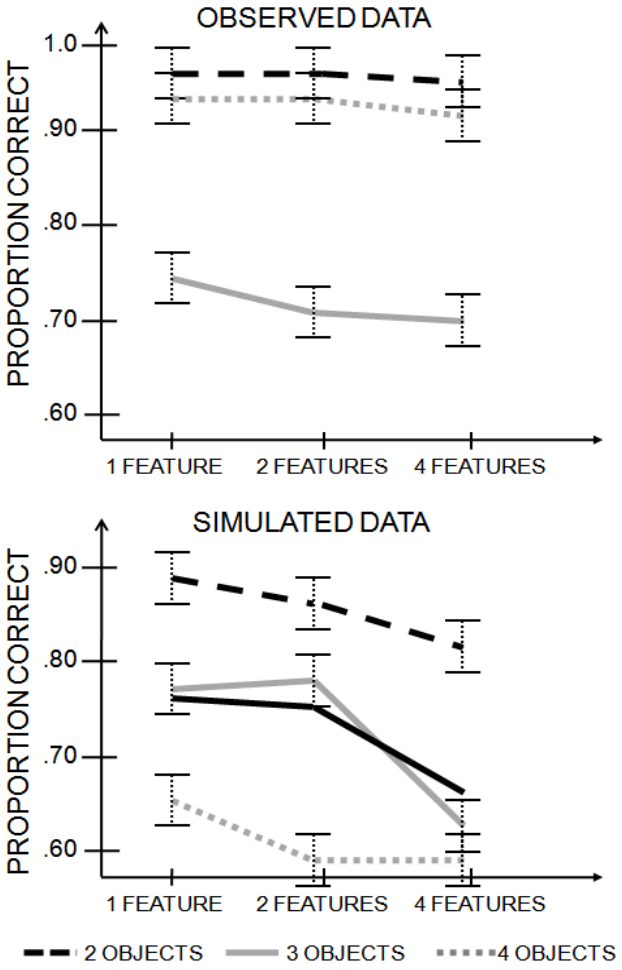


Fig. 3. Upper panel: data from Vogel et al. (2001) for set sizes two and four, and from Oberauer and Eichenberger (2013) for set size three. Bottom panel: data simulated by the oscillatory model (black line = mean from all set sizes). Bars = 95% confidence intervals.

For set sizes two and four, we could compare simulated data with the results of Vogel et al., who used exactly such set sizes. For set size four, our data diverged from their results, because there was a similar drop in accuracy as a function of the number of features, $F = 6.57$, $p = .001$, as was in the three-objects condition. The same occurred when only two objects were presented to the model, $F = 39.26$, $p < .001$. The simulated data as well as data of Vogel et al., and Oberauer and Eichenberger, are presented in Fig. 3. What is apparent in Vogel et al., data is the huge ceiling effect, and only marginal drop in accuracy from set size two to four. In consequence, the null effect of the number of features in Vogel et al. for some unknown reason might have resulted from their ceiling effect. In contrast, simulated data more closely matched Oberauer and Eichenberger data, who observed much lower accuracy for set size three than did Vogel et al. for set size four (and even than their set size six). Thus, the prediction of Oberauer and Eichenberger that the number of features maintained loads on WM and affects accuracy, congruent with our simulation results, seems to be more realistic.

In the model, both the number of objects and features affected accuracy because increasing each number increased the total amount of lateral inhibition. However, the impact of additional objects was stronger than the impact of features (i.e., it was easier to maintain two two-feature objects than four one-feature objects). The likely cause of such an effect consists of the fact that increasing the number of features increased both inhibition (decreasing overall capacity) and – to some extent – coactivation (increasing capacity). Thus, the negative effect of additional features was partially attenuated by more robust representation of each object. Increasing the number of objects increased only inhibition, but not coactivation, so it impact was stronger.

Discussion

Using a novel oscillatory model, which aimed to describe the functioning of human VWM, in line with the capacity-as-objects account we showed that the major limitation of VWM in our simulations resulted from the number of to-be-maintained objects. However, we also demonstrated that, apart from the number of objects, also the number of visual features maintained in VWM may be limited. When the number of features became increased (especially, to four features), the system, being close to the maximum overall amount of lateral inhibition it could handle, could not maintain additional features with the same accuracy as one feature. This mechanism seems to explain well the data obtained within the capacity-as-features account (Oberauer & Eichenberger, 2013; Cowan et al., 2013).

Although the present model is limited by both the number of objects and the number of features it can effectively maintain, the model does not support the explanations of VWM capacity in terms of the total informational resource that is consumed by both the objects and the features (Bays, Catalao, & Husain, 2009; Bays & Husain, 2008). First, although the drop in accuracy did result from additional features, the performance of the model decreased much more slowly than did increase the total number of features. For example, accuracy for four features stored in the four-objects-one-feature condition was only 10% higher than accuracy in the four-objects-four-features condition, even though the total number of features maintained increased from 4 to 16 features. This is not in line with predictions of the sheer limited-resource account. Additional assumptions made to this approach that would explain that the more features, the less VWM resource on average is consumed by a feature, help only a little. As cogently noted by Oberauer and Eichenberger (2013), such add-ons supplementing the limited-resource account made it virtually unfalsifable.

However, even though the present model is more compatible with those accounts of capacity that postulate that its currency are available slots in VWM (one slot = one object), but with more features per object the probability that surplus features will not be encoded effectively, it seems to nicely extend these accounts. Most of existing slot models were formulated as pure mathematical formulae that yield the probability of the correct change detection as a function of the number of objects, the number of features, and sometimes some additional parameters (e.g., Cowan et al., 2013; Oberauer & Eichenberger, 2013). In contrast, the present model is a process (i.e., computational) model that not only describes the relation between the characteristics of the change detection task and the resulting detection accuracy, but also generates the very complex dynamics (interpreted at the neural level) underlying the performance on this task. For example, the model explains the nature of slots in terms of dynamic bindings that interact mutually, and are prone to stochastic as well as chaotic effects.

To conclude, the presented dynamic model of VWM provided the prediction that although the VWM capacity is strongly constrained by the number of maintained objects (which is a widely observed fact), it is also, though more weakly, limited by the number of features per object (the factor that so far has not been examined exhaustively, and still awaits the comprehensive empirical study). In general, the study implies that in cognitive science at least for some research problems more can be theoretically understood from the development of process models, generating a particular phenomenon, than from pure mathematical models that only describe such phenomenon.

Acknowledgments

This work was sponsored by The National Science Centre (NCN) of Poland (grant 2013/11/B/HS6/01234). The model's code can be downloaded from ecfi-group.eu

References

Alvarez, G. A., & Cavanagh, P. (2004). The capacity of visual short-term memory is set both by visual information load and by number of objects. *Psychological Science, 15,* 106–111.

Bays, P. M., Catalao, R. F. G., & Husain, M. (2009). The precision of visual working memory is set by allocation of a shared resource. *Journal of Vision, 9,* 7.

Bays, P. M., & Husain, M. (2008). Dynamic shifts of limited working memory resources in human vision. *Science, 321,* 851–854.

Brady, T. F., Konkle, T., & Alvarez, G. A. (2011). Review of visual memory capacity: Beyond individual items and toward structured representations. *Journal of Vision, 11,* 1-34.

Chuderski, A., Andrelczyk, K., & Smoleń, T. (2013). An oscillatory model of individual differences in working memory capacity and relational integration. *Cognitive Systems Research, 24,* 87-95.

Clevenger, P. E., & Hummel, J. E. (2014). Working memory for relations among objects. *Attention, Perception, & Psychophysics, 76,* 1933-1953.

Cowan, N. (2001). The magical number 4 in short-term memory: A reconsideration of mental storage capacity. *Behavioral and Brain Sciences, 24,* 87-114.

Cowan, N., Blume, C. L., Saults, J. S. (2013). Attention to attributes and objects in working memory. *Journal of Experimental Psychology: Learning, Memory, & Cognition, 39,* 731-747.

Cowan, N., Li, D., Moffitt, A., Becker, T. M., Martin, E. A., Saults, J. S., & Christ, S. E. (2011). A neural region of abstract working memory. *Journal of Cognitive Neuroscience, 23,* 2852-2863.

Fukuda, K., Awh, E., & Vogel, E. K. (2010). Discrete capacity limits in visual working memory. *Current Opinion in Neurobiology, 20,* 177-182.

Fukuda, K., Vogel, E. K., Mayr, U., & Awh, E., & (2011). Quantity, not quality: The relationship between fluid intelligence and working memory capacity. *Psychonomic Bulletin & Review, 17,* 673-679.

Horn, D., & Usher, M. (1992). Oscillatory model of short term memory. In J. E. Moody, S. J., Hanson, & R. P. Lippmann (Eds.), *Advances in Neural Processing and Information Systems Vol. 4* (pp. 125-132). Morgan and Kaufmann.

Johnson-Laird, P. N. (2006). *How we reason?* Oxford: Oxford University Press.

Kamiński, J., Brzezicka, A., & Wróbel, A. (2011). Short-term memory capacity (7±2) predicted by theta to gamma cycle length ratio. *Neurobiology of Learning and Memory, 95,* 19-23.

Luck, S. J., & Vogel, E. K. (1997). The capacity of visual working memory for features and conjunctions. *Nature*, *390,* 279-281.

McGrew, K. S. (2009). CHC theory and the human cognitive abilities project: Standing on the shoulders of the giants of psychometric intelligence research. *Intelligence, 37,* 1-10.

Oberauer, K., & Eichenberger, S. (2013). Visual working memory declines when more features must be remembered per object. *Memory & Cognition, 41,* 1212–1227.

Raffone, A., & Wolters, G. (2001). A cortical mechanism for binding in visual memory. *Journal of Cognitive Neuroscience, 13,* 766-785.

Siegel, M., Warden, M. R., & Miller, E. K. (2009). Phase-dependent neuronal coding of objects in short-term memory. *PNAS, 106,* 21341–21346.

Todd, J. J., & Marois, R. (2004). Capacity limit of visual short-term memory in human posterior parietal cortex. *Nature, 428,* 751-754.

Usher, M., Cohen, J. D, Haarmann, H., & Horn D. (2001). Neural mechanism for the magical number 4: Competitive interactions and nonlinear oscillation. *Behavioral and Brain Sciences, 24,* 151-152.

Vogel, E. K., Woodman, G. F., & Luck, S. J. (2001). Storage of features, conjunctions, and objects in visual working memory. *Journal of Experimental Psychology: Human Percepction and Performance, 27,* 92-114.

Xu, Y., & Chu, M. M. (2009). Selecting and perceiving multiple visual objects. *Trends in CogSci, 13,* 167-174.

How should we evaluate models of segmentation in artificial language learning?

Raquel G. Alhama (rgalhama@uva.nl)

Remko Scha (scha@uva.nl)

Willem Zuidema (w.h.zuidema@uva.nl)
Institute of Language, Logic and Computation, Science Park 107
Amsterdam, 1098XG, The Netherlands

Keywords: Statistical Learning; Cognitive Modelling

Introduction: Statistical Learning

One of the challenges that infants have to solve when learning their native language is to identify the words in a continuous speech stream. Some of the experiments in Artificial Grammar Learning (Saffran, Newport, and Aslin (1996); Saffran, Aslin, and Newport (1996); Aslin, Saffran, and Newport (1998) and many more) investigate this ability. In these experiments, subjects are exposed to an artificial speech stream that contains certain regularities. Infants are typically tested in a preferential looking paradigm; adults, in contrast, in a 2-alternative Forced Choice Tests (2AFC) in which they have to choose between a word and another sequence (typically a *partword*, a sequence resulting from misplacing boundaries).

One of the key findings of AGL is that both infants and adults are sensitive to transitional probabilities and other statistical cues, and can use them to segment the input stream. Several computational models have been proposed to explain such findings. We will review how these models are evaluated and argue that we need a different type of experimental data for model evaluation than is typically used and reported. We present some preliminary results and a model consistent with the data.

Models of Segmentation

Many different types models of segmentation have been proposed, that differ in the representational framework used (including symbolic, statistical, connectionist and exemplar-based representations, and combinations thereof) and in the level of description chosen. We focus here on three representative models: (i) Goldwater, Griffiths, and Johnson (2009) present a Bayesian rational model, which assumes an ideal learner and computes the most probable set of segments that could have produced the observed stream. (ii) PARSER (Perruchet & Vinter, 1998) is a symbolic, exemplar-based model that incrementally breaks the input stream into segments and memorizes them; when the weight of a segment in memory is strong enough, it influences how the subsequent part of the stream is segmented. The model also incorporates forgetting and interference. (iii) In the connectionist paradigm, TRACX (French et al., 2011) present a recognition-based neural network that learns to represent the input. The resemblance of the output representations with the input sequence indicates how well the sequence is recognized by the model.

Which of these models fits the experimental data best? That question turns out to be difficult to answer, as data from 2AFC experiments – the vast majority of experiments with adults in the AGL paradigm – make it difficult to choose between models, despite important differences in the cognitive processes they assume. This is because in 2AFC, only the relative preference of one stimulus over another one is measured, and typically only a single average accuracy is reported. All the models we considered have enough free parameters to reproduce any desired average accuracy.

In existing work on model evaluation, several authors have therefore proposed to focus on the analyses of the internal representations of the model (consisting on segments of the stream, as well as some score in the form of memory strength or probability), or on comparing the performance of the model in a 2AFC setting over a range of conditions. Perruchet and Vinter (1998) provide an example of the former. They define two criteria: the *loose criterion* states that the internal memory of the model contains the words with the highest weights, but also other sequences; to fulfill the *strict criterion*, the memory must contain the words with the highest weights, but other 'legal' sequences are possible (ex: two words concatenated). We will show that the assumptions that *all* the words should have the highest weights, and that the non-legal sequences should be forgotten, is not consistent with empirical data.

Frank, Goldwater, Griffiths, and Tenenbaum (2010), on the other hand, provide an example of the latter. They evaluate a number of different models by comparing the performances in a 2AFC task with those of humans for a range of conditions (e.g., for different numbers of words). Although this constitutes a major improvement over comparing only to a single datapoint, we still find that models which embody fundamentally different assumptions can easily provide similar performance. This is the case, for instance, with the Bayesian model of (Goldwater et al., 2009) and TRACX (French et al., 2011).

A call for a different type of experiments

We suggest a different experimental setup that we believe should complement the extensive body of research with 2AFC tests, and we advance some preliminary results.

In our experiment, we replicated the familiarization phase of experiment 1 in Pena, Bonatti, Nespor, and Mehler (2002).

In the test phase, each trial consisted of two questions about a single sequence (either a word or a partword). The first question was "Is this sequence a word of the language you have heard?", and it allows for a yes/no answer. The following question was a confidence rate about the previous answer, from 1 (not confident) to 7 (very confident).

Figure 1 shows the average responses for each test item. We do not claim that these responses provide us with direct access to the strength of the mental representations of the subjects, but we believe they are more revealing than the 2AFC responses.

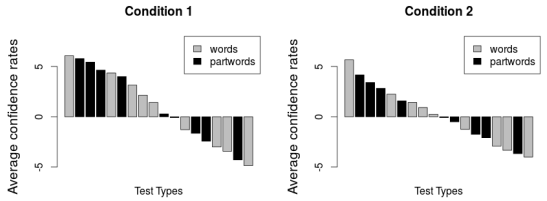


Figure 1: Average confidence rates for each test stimulus type, in decreasing order. Confidence rates for negative answers have negative values. Conditions only differ in the randomization of the syllables.

An important observation of these responses is that some partwords are rated higher than some of the words. This invalidates the criteria proposed by Perruchet and Vinter (1998), and justifies the need of models that store segments creating the skewed distributions observed. In the next section we present a model that can generate this kind of output.

The Retention & Recognition Model

Our model, the Retention&Recognition model (RnR) (Alhama, Scha, & Zuidema, 2014) can be considered a probabilistic chunking model. It incrementally breaks the stream into segments which may be stored into its internal memory, along with a weight that we call 'subjective frequency'. Given an initially empty memory, for any segment from the input stream, the model will attempt to recognize it with probability P_{rec} (eq. 1). If it succeeds, the subjective frequency of the segment is incremented with 1. If it fails, the model may still retain it in memory, with probability P_{ret} (eq. 2). In this case, it will either add it for the first time with initial subjective frequency one, or increment its subjective frequency with 1.

$$P_{rec}(segment) = (1 - B^{activation(segment)}) \cdot D^{\#types} \quad (1)$$

$$P_{ret}(segment) = A^{length(segment)} \cdot C^{\pi} \quad (2)$$

The model involves free parameters (A, B, C, D) that may be fitted to empirical data. The retention probability is inversely correlated with the length of the segment. The factor C^{π} attenuates the retention probability unless the segment appears right after a micropause. The recognition probability uses an activation function that depends on the accumulated subjective frequency of the subsequence. The number of word types adds difficulty to the task, resulting in a decreased recognition probability.

The interaction between retention and recognition can generate a range of results similar to those seen in the experiment. The recognition formula provides the dynamics of *rich get richer* (also present in some nonparametric bayesian models): once a sequence starts being recognized, it will be easier and easier to recognize, leading to a big subjective frequency. However, the retention and the probabilistic nature of the model are responsible for the fact that not all sequences are first incorporated into memory at the same time. This yields skewed distributions, similar to the distribution observed in our experiment.

Conclusions

In order to choose between models of segmentation we need to contrast them with data from experimental paradigms that complement the existing 2AFC results with data that shows other properties. We have proposed an alternative paradigm for experiments, with the hope that it will inspire many more experiments along this line. We have also illustrated how it provides empirical support for the misrepresentation of partwords in some models. Finally, we have described RnR, a probabilistic chunking model that can reproduce the patterns revealed by the data.

References

Alhama, R. G., Scha, R., & Zuidema, W. (2014). Rule learning in humans and animals. In *Proceedings of the international conference on the evolution of language* (p. 371-372).

Aslin, R. N., Saffran, J. R., & Newport, E. L. (1998). Computation of conditional probability statistics by 8-month-old infants. *Psychological science*, *9*(4).

Frank, M. C., Goldwater, S., Griffiths, T. L., & Tenenbaum, J. B. (2010). Modeling human performance in statistical word segmentation. *Cognition*.

French, R. M., Addyman, C., & Mareschal, D. (2011). Tracx: A recognition-based connectionist framework for sequence segmentation and chunk extraction. *Psychological Review*, *118*(4), 614.

Goldwater, S., Griffiths, T. L., & Johnson, M. (2009). A bayesian framework for word segmentation: Exploring the effects of context. *Cognition*, *112*, 21-54.

Pena, M., Bonatti, L., Nespor, M., & Mehler, J. (2002). Signal-driven computations in speech processing. *Science*, *298*(5593), 604–607.

Perruchet, P., & Vinter, A. (1998). Parser: A model for word segmentation. *Journal of Memory and Language*, *39*(2), 246–263.

Saffran, J. R., Aslin, R. N., & Newport, E. L. (1996). Statistical learning by 8-month-old infants. *Science*, *274*(5294), 1926–1928.

Saffran, J. R., Newport, E. L., & Aslin, R. N. (1996). Word segmentation: The role of distributional cues. *Journal of memory and language*, *35*(4), 606–621.

A constraint-based approach to pronoun interpretation in Italian

Margreet Vogelzang (margreet.vogelzang@rug.nl)
Center for Language and Cognition Groningen, University of Groningen
P.O. Box 716, 9700 AS Groningen, the Netherlands

Hedderik van Rijn (hedderik@van-rijn.org)
Departments of Experimental Psychology & Psychometrics and Statistics, University of Groningen
Grote Kruisstraat 2/1, 9712 TS Groningen, the Netherlands

Petra Hendriks (p.hendriks@rug.nl)
Center for Language and Cognition Groningen, University of Groningen
P.O. Box 716, 9700 AS Groningen, the Netherlands

Keywords: pronoun interpretation; cognitive modeling; pro-drop; constraint-based modeling

Introduction

Anaphoric pronouns such as the English 'he' are used in daily life to refer to entities that are mentioned in the previous discourse context. Such pronouns are ambiguous, as they can refer to any singular male entity in the discourse. Ambiguous pronouns have to be resolved in order to be (correctly) interpreted. Generally, third person pronouns are interpreted as referring to the grammatical subject of the previous sentence (subject bias; Gordon & Scearce, 1995), or as referring to the discourse topic (most accessible antecedent; Ariel, 1990; Grosz, Weinstein, & Joshi, 1995), resulting in topic continuation.

The resolution of ambiguous pronouns and taking into account the previous discourse context requires processing. With a constraint-based approach (derived from Optimality Theory; Prince & Smolensky, 2008), we try to identify rules (constraints) that guide this pronoun resolution process. We propose a number of constraints that, when implemented in a cognitive model, can simulate pronoun processing in Italian.

Pronoun interpretation in Italian

In pro-drop languages like Italian, contrary to non-pro-drop languages such as Dutch and English, a rich verb morphology allows for an additional subject form: the subject can be completely omitted, resulting in a null subject. Italian thus offers the possibility to either place a subject pronoun overtly or to use a null subject. Generally, a null pronoun refers to the previous discourse topic, whereas an overt pronoun refers to another, non-topical, referent (Carminati, 2002). However, this is merely a preference and interpretations of null as well as overt pronouns can vary.

So, a potential model of pronoun resolution in Italian can not simply take the topical character or the grammatical subject of the previous sentence as the referent of the pronoun, but will need more elaborate constraints. First, we ran an experiment to examine how different Italian subject forms are interpreted in discourse.

Experiment

In the experiment, Italian adults (n=40) heard short stories, the last clause of which contained one of three different subject anaphora: A full noun phrase (NP) such as *the dog* as an unambiguous baseline condition, a null subject (Ø), and the overt subject pronouns *lui* ('he') and *lei* ('she'). A sample story:

Il cane va a fare un viaggio in Germania.
The dog is going on a trip to Germany.
Ieri sera il cane ha invitato il gatto a viaggiare insieme,
Last night the dog has invited the cat to travel together,
mentre Ø/lui/il cane si lavava prima della partenza.
while Ø/he/the dog washed himself before the departure.

We recorded participants' responses to a referent selection question, in which they could choose between the discourse topic (*il cane*) and a non-topic antecedent (*il gatto*) as the referent of the subject anaphor.

The results of the experiment (Figure 1) show that, in line with Carminati (2002), null subjects are generally interpreted as referring to the discourse topic (86% of the time). Interpretations of overt subject pronouns vary somewhat, as they are interpreted as referring to the discourse topic 39% of the time and as referring to the non-topical referent 61% of the time.

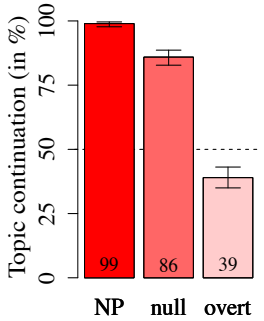


Figure 1: Experimental results for the interpretations of full NPs, null subjects, and overt subject pronouns in Italian.

Proposed model

We propose an adaption of the pronoun resolution model of Van Rij, Van Rijn, & Hendriks (2011) in order to simulate the processing and interpretation of Italian pronouns. The model is to be implemented in the cognitive architecture ACT-R (Anderson, 2007), which constrains models to ensure psychological plausibility. The proposed model uses the following, hierarchically ordered, constraints:

[1] There are no null subjects that refer to a non-topic
[2] There are no pronouns that refer to an entity that is not activated
[3] Avoid NPs
[4] Avoid overt pronouns

There three main steps in the model: determining the discourse topic, interpreting the pronoun, and perspective-taking. In this final step, the model takes the perspective of the speaker in order to determine if a speaker would indeed have used the encountered expression for the selected interpretation.

In the first step the character with the highest activation is taken as the current discourse topic. The activation of a referent is based on the previous occurrences in the discourse and its grammatical role during these occurrences. Mistakes can be made when determining the discourse topic because activation in ACT-R is subject to noise.

In the second step of the model, either a null subject or an overt subject pronoun needs to be interpreted. For Italian null subjects the discourse topic is taken as the referent of the pronoun based on constraint [1]. For overt subject pronouns however, the constraints do not restrict the interpretation to either the topic or to a non-topical, activated antecedent. Therefore, the referent of an overt subject pronoun can not be determined in this step.

In this case, the third step of taking into account the perspective of the speaker is essential. This final step can have three possible input states: either the speaker wants to refer to the discourse topic, to a non-topical, activated antecedent, or to a non-topical antecedent that is not activated. If the intended referent is the topic, constraints [1] and [2] do not restrict which form can be used. In this case, constraints [3] and [4] are applied: a null subject is easier (less effortful, more economic) to produce than an overt subject pronoun, which is easier to produce than a full NP (based on Burzio, 1998). Thus, if a speaker would want to refer to the discourse topic, she would use a null subject.

When the speaker wants to refer to a non-topic, constraint [1] does not apply. If the antecedent is not activated in the current discourse, the speaker will not use a pronoun (on the basis of constraint [2]), so she will use a full NP. If the non-topic antecedent is activated in the discourse however, constraint [1] still prevents the speaker from using a null subject, but constraint [2] does not apply. Therefore, constraint [3] determines that an overt subject pronoun is used instead of a full NP. So, because a speaker would use an overt pronoun to refer to a non-topical, activated antecedent, a listener would finally interpret an overt subject pronoun as referring to the non-topical, activated character.

So far, we have explained the hierarchical constraints and processing steps, which together lead to the correct interpretation preferences for null and overt subject pronouns. Additionally, activation in ACT-R is subject to noise and thus pronouns will not always be interpreted in the same way. However, the model should also account for the strong variation in the interpretations of overt subject pronouns. This can be simulated by the third step of the model, the perspective-taking step, which is necessary for the interpretation of overt subject pronouns but not for null subjects. Since the execution of an additional processing step takes time and effort, time constraints on language processing may prevent the third processing step from being completed. If the interpretation of the overt subject pronoun has not been determined yet, it will be guessed. This will result in more variability in the interpretation of overt subject pronouns than in the interpretation of null subjects.

Conclusions

In this paper, we propose a model that uses a constraint-based approach and perspective-taking. Combined with constraints from the cognitive architecture ACT-R and constraints on language processing, the model can plausibly simulate subject pronoun interpretation in Italian.

References

Anderson, J. R. (2007). *How Can the Human Mind Occur in the Physical Universe?* New York: Oxford University Press, USA.

Ariel, M. (1990). *Accessing noun-phrase antecedents*. London: Routledge.

Burzio, L. (1998). Anaphora and soft constraints. In P. Barbosa, D. Fox, P. Hagstrom, M. McGinnis, & Pesetsky D. (Eds.), *Is the Best Good Enough? Optimality and Competition in Syntax* (pp. 93–113). Cambridge, MA: MIT Press.

Carminati, M. N. (2002). *The processing of Italian subject pronouns* (Doctoral dissertation). University of Massachusetts, Amherst, USA.

Gordon, P. C., & Scearce, K. A. (1995). Pronominalization and discourse coherence, discourse structure and pronoun interpretation. *Memory & Cognition*, *23*(3), 313–323.

Grosz, B. J., Weinstein, S., & Joshi, A. K. (1995). Centering : A Framework for Modeling the Local Coherence of Discourse. *Computational Linguistics*, *21*(2), 203–225.

Prince, A., & Smolensky, P. (2004). *Optimality Theory: Constraint Interaction in Generative Grammar*. Oxford, UK: Blackwell.

Van Rij, J., Van Rijn, H., & Hendriks, P. (2011). WM load influences the interpretation of referring expressions. In *Proceedings of the 2nd workshop on Cognitive Modeling and Computational Linguistics* (pp. 67–75). Association for Computational Linguistics, Portland, OR.

Investigating the semantic representation of Chinese emotion words with co-occurrence data and self-organizing maps neural networks

Yueh-Lin Tsai
Department of Psychology, National Cheng Kung University, Taiwan

Hsueh-Chih Chen
Department of Educational Psychology and Counseling, National Taiwan Normal University, Taiwan

Jon-Fan Hu
Department of Psychology, National Cheng Kung University, Taiwan

Keywords: Chinese emotion words, word co-occurrence, Latent Semantic Analysis, semantic representation, self-organizing maps.

Introduction

Regarding the investigation of the word representations, previous researchers often asked participants to rate the similarities between emotion words (Barrett, 2004; Cheng, Cheng, Cho, & Chen, 2013; Romney, Moore, & Rusch, 1997), or to give the scores upon certain psychological dimensions (e.g. valence, arousal, et. al) (Bradley & Lang, 1999; Cho, Chen, & Cheng, 2013; Morgan & Heise, 1988). These direct similarity-based or anchor-based ratings indeed emerge categorical properties of emotion words according to existing theoretical postulations. However, these methods are just based on subjective and retrospective report data. To date we might well grasp what the meanings of general concepts are, but what emotional concepts refer to is still not fully clear. Hence adopting more objective way and robust theories about how people learn and represent the semantic concepts of emotion words is crucial for leading us to in-depth investigation.

Analyzing general products of word use might shed light to answer the question. Latent Semantic Analysis (LSA, Landauer, Foltz, & Laham, 1998) is used to study the concepts behind words from the perspective of how human build-up the word meanings. Accordingly, the semantic representations of words are gradually shaped and learned through the multiple constraints of the input data from the environment since childhood. By calculating the co-occurrence matrix between words and documents, we could study the semantic knowledge from large-scale corpus, and hence explore the possible relationships between individual words.

Besides, previous success of neural networks showed the competence to study the inner representations of human mind. Particularly, Self-Organizing Maps (SOM) model can vectorize the representation relationships of categories of words and display the topological properties across the maps (P. Li, 2009; Ping Li, Burgess, & Lund, 2000; P. Li, Farkas, & MacWhinney, 2004). Parameters of SOM models are sensitive tools to extract subtle variations of word meanings, therefore grasping the common properties under an unsupervised learning manner to express similarity- or anchor-free semantic representations of the complex emotion word meanings. Instead of comparing pairs of emotion words or rating the semantic properties based on predefined dimensions, the present modeling study combined both corpus-based analysis (LSA) and connectionist model to delineate the complexity of semantic representations of Chinese emotion words.

Methods

The Academia Sinica Balanced Corpus of Modern Chinese 3.0 (Sinica Corpus 3.0) was used as the corpora to provide Chinese word uses. This corpus has nearly ten thousand documents, fifteen million words, which were collected from magazines, speeches, internet, and other media in Taiwan. Target emotion words were selected from Taiwan Corpora of Chinese Emotions and Psychophysiological Data on EmotioNet (http://ssnre.psy.ntu.edu.tw/). This data collected 218 Chinese emotion-describing words, with 353 participants rated each emotion words for valence, arousal, dominance, continuance, frequency, and typicality (Cho et al., 2013). We chose 161 two-character words from the database. For validate the data, 36 nouns similar to the stimulus used in Lund and Burgess (1996) were also selected. The words included 12 animal names, 12 words of body parts, and 12 words of nations.

The present study established the semantic representations in Sinica Corpus 3.0 using CTM_PAK (Zhao, Li, & Kohonen, 2011). Window size and list of words were adjusted to the data of emotional words as the parameters of interested. SOM was used to represent the semantic representations of the emotion words but not the nouns. 5 nearest neighbors of each emotional word were also generated and being rated by researcher if the nearest neighbors belong to the same group or not.

Results

The SOM model of 36 nouns showed that three categories of words were projected onto different map regions, with only few words locating at the wrong regions. The results here were at large consistent with the findings of Lund and Burgess (1996), although they used multi-dimensional

scaling as a plausible approach to probe semantic category in the corpus. Hence we might ascertain that the present data could at least distinguish different types of semantic categories under a coarse framework as Lund and Burgess (1996) reported.

However, the words were not form any relevant clusters in the SOM models, and the semantic-similar words were not projected into adjacent region even in the different map size, training length, and initial radius. For excluding the possibility that some low-frequency words would affect the model, SOM models with emotional words appeared more than fifty times in the whole corpus were built. The results also showed that no meaningful clusters were formed.

The average percentages in which five-nearest neighbor words were in the same category as the target word revealed that across data with different parameters, the accuracy of the five-nearest neighbors was all below to thirty percent. Although there were some variations between data, the patterns were not clear so that we treat these variations as random and had nothing to do with the size of moving window and the content of word list.

Discussion

As the result showed, although the data could categorize different types of noun, the semantic representations of emotional words were far from perfect. One of the possible reasons is that emotional words can't be well anchored by other co-occurring words because of the complex and subjective component. As the theory of LSA mentioned, the vectors of each word could just represent the semantic concept across all contexts. Because of the subjective characteristic, highly different emotional words might be able to apply to the same context. So maybe it's not sufficient to separate the emotional words and concepts with only co-occurrence data.

Despite there seems to have above possibility, when taking a closer look in the SOM model of 36 nouns, the arrangement within category were also showed no finer and meaningful categories. Hence although we couldn't reject that emotional words might involve some complex components, the way we extract concepts behind words might have its limitation. It's possible that we have to include some dimensions about subjective feelings to establish the semantic structure of emotional words better. Across the theories about knowledge structure, embodiment cognition highlights the importance of motor, perceptual, and introspective states while forming and retrieving concepts. Further research might get more insight with the inclusion of this kind of approach, and the semantic structure could be well established.

Conclusion

The present study strived to investigate the semantic representation of Chinese emotional words with corpus-based analysis. The results showed that although the data could separate different types of nouns, it is not sufficient to separate and categorized different emotional concept. Further studies have to take other theory into consideration to construct the mental representation more properly.

Reference

Barrett, L. F. (2004). Feelings or words? Understanding the content in self-report ratings of experienced emotion. Journal of personality and social psychology, 87(2), 266.

Bradley, M. M., & Lang, P. J. (1999). Affective norms for English words (ANEW): Instruction manual and affective ratings: Citeseer.

Cheng, C.-M., Cheng, J., Cho, S.-L., & Chen, H.-C. (2013). A Structure Analysis of Chinese Emotions. Chinese Journal of Psychology, 55(4), 417-438.

Cho, S.-L., Chen, H.-C., & Cheng, C.-M. (2013). Taiwan Corpora of Chinese Emotions and Relevant Psychophysiological Data -- A Study on the Norm of Chinese Emotional Words. Chinese Journal of Psychology, 55(4), 493-523.

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. Biological cybernetics, 43(1), 59-69.

Landauer, T. K., Foltz, P. W., & Laham, D. (1998). An introduction to latent semantic analysis. Discourse processes, 25(2-3), 259-284.

Li, P. (2009). Lexical organization and competition in first and second languages: computational and neural mechanisms. Cogn Sci, 33(4), 629-664.

Li, P., Burgess, C., & Lund, K. (2000). The Acquisition of Word Meaning through Global Lexical Co-occurences. In E. V. Clark (Ed.), Proceedings of the thirtieth stanford child language research forum. Stanford: CA: Center for the Study of Language and Information.

Li, P., Farkas, I., & MacWhinney, B. (2004). Early lexical development in a self-organizing neural network. Neural Netw, 17(8-9), 1345-1362.

Lund, K., & Burgess, C. (1996). Producing high-dimensional semantic spaces from lexical co-occurrence. Behavior Research Methods, Instruments, & Computers, 28(2), 203-208.

Morgan, R. L., & Heise, D. (1988). Structure of emotions. Social Psychology Quarterly, 19-31.

Romney, A. K., Moore, C. C., & Rusch, C. D. (1997). Cultural universals: Measuring the semantic structure of emotion terms in English and Japanese. Proceedings of the National Academy of Sciences, 94(10), 5489-5494.

Zhao, X., Li, P., & Kohonen, T. (2011). Contextual self-organizing map: software for constructing semantic representations. Behav Res Methods, 43(1), 77-88.

Understanding the Misunderstood

David Tobinski (David.Tobinski@uni-due.de)
Institute of Psychology, Berliner Platz 6 - 8
45117 Essen, Germany

Oliver Kraft (Oliver.Kraft@stud.uni-due.de)
Department of Educational Sciences, Berliner Platz 6 - 8
45117 Essen, Germany

Keywords: LSA, Text Understanding, Automatic Tutoring

Introduction

The research on *Latent Semantic Analysis* (LSA) in the domain of natural language processing (NLP) shows the efficiency of this method (Landauer & Dumais, 1997). Nowadays the applicability in interactive semantic rich E-Learning contexts is interesting.

Future applications may use LSA techniques for automatic tutoring (Graesser et al., 1999) or automatic scoring of written essays in trainings, tests or MOOCs, which will be the way out of single-choice and multiple-choice tests in interactive learning environments.

Text understanding is central in interactive learning environments. For this a written essay is a much better indicator than single or multiple choice questions and answers. The question of rating *if a text has been understood* is followed by the more important question of what has not been understood within the text, so that intelligent feedback can be given.

LSA can see if knowledge has been decoded into a written essay or not. If textual information is not included, it is still unclear whether the information was not understood or just not activated enough (Anderson, 1976).

Methods

In our study, we try to use LSA to rate text understanding combined with a new method of classifying paragraphs by online-highlighting within the existing pdf-file.

Participants

The study was realized with 16 German participants (11 female; mean age 22,5 years).

Procedure

In a first part the participants had to read the scientific paper *"What Benefits do the Findings of Brain Science have for Pedagogics?"* (*9 pages, 51 paragraphs, 6495 words*) in a specially programmed online pdf-reader and their task was to highlight *important and difficult parts* of the text within the pdf-file. The paper was displayed by a PDF-Viewer, which was extended in two ways. The first extension added the ability to highlight the text in two colors whereas the second extension enabled the students to store the marked text on the server. Both marks were different in color, a blueish color marked parts of the paper students thought to be of central importance and a reddish color to mark difficult parts. Their time was limited to the end of the session, which meant about 60 minutes, to read and mark the text. There was no guideline as to whether the participants should first read and mark afterwards or do both simultaneously. The participants were not especially instructed to learn the paper and there was no information given about following tasks.

One week later all participants had to reconstruct the paper by writing an essay within a special online text-editor.

Results

The latent semantic space was reduced to 21 dimensions and no weight functions were applied. Cosine was used to calculate the similarity between students' summaries and the paragraphs of the paper. Regarding the text length, four groups are appropriate to analyze (*words<100; words=101-200; words=201-300; words>300*).

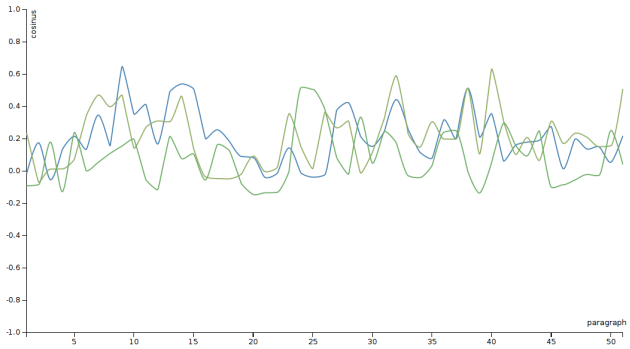


Figure 1: The cosine parameter for three participants with more than 300 written words.

Well decoded paragraphs are significantly differentiated from ill decoded paragraphs. The reason why the performance of text reconstruction differs this way lies in the text-structure itself, which can be more analyzed by autocorrelation, and in the previous knowledge of the participants, which can be analyzed within the highlighted text areas.

The autocorrelation of paragraphs presents the semantic relatedness of the paper very well.

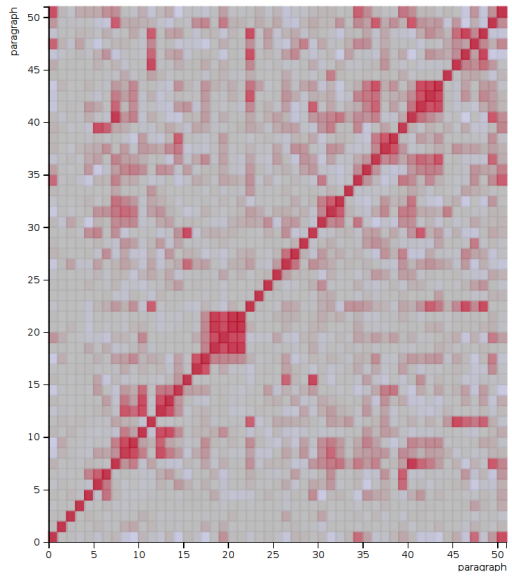


Figure 2: The 51 paragraphs' semantic content patterns.

The 51 paragraphs show perfect patterns of their semantic content. Very special paragraphs can be identified immediately (e.g. paragraph 12 about the *amygdala*).

The previous knowledge of the participants is incorporated in their highlighted text areas. The participants marked 18.87 % of the paper as of central importance and 1.4 % as opaque on average. Most (9 of 16) participants marked nothing as opaque. Exactly one student marked more opaque than as of central importance.

Table 1: Length (words in total) of the written essays and of the highlighted words

	Essay		**Important**		**Difficult**	
1	355	5.47%	627	9.65%	68	1,05%
2	221	3.40%	1138	17.52%	0	0%
3	204	3.14%	1778	27.37%	0	0%
4	104	1.60%	649	9.99%	840	12.93%
5	36	0.55%	1044	16.07%	0	0%
6	58	0.89%	1480	22.79%	0	0%
7	107	1.65%	1084	16.69%	33	0.51%
8	413	6.36%	327	5.03%	0	0%
9	321	4.94%	2170	33.41%	357	5.50%
10	170	2.62%	1420	21.86%	0	0%
11	55	0.85%	113	1.74%	0	0%
12	117	1.80%	2028	31.22%	33	0.51%
13	75	1.15%	1534	23.62%	0	0%
14	35	0.54%	1563	24.06%	91	1.40%
15	220	3.39%	1303	20.06%	30	0.46%
16	136	2.09%	1347	20.74%	0	0%

Discussion

LSA is a promising method for generating intelligent feedback in online courses and other E-Learning environments. For generating a more differentiated feedback in those systems, additional information is needed. Therefore, special highlighting methods could be introduced. Those techniques are still in development and special trainings for participants are needed to make them a standard tool in educational contexts.

References

Anderson, J. R. (1976). *Language, Memory and Thought.* New Jersey: Lawrence Erlbaum Associates.

Graesser, A. C., Wiemer-Hastings, K., Wiemer-Hastings, P., & Kreuz, R. (1999). AutoTutor: A simulation of a human tutor. *Cognitive Systems Research*, 1(1), 35–51.

Landauer, T. K., & Dumais, S. T. (1997). A Solution to Plato ' s Problem : The Latent Semantic Analysis Theory of Acquisition , Induction , and Representation of Knowledge. *Psychological Review*, *104*(2), 211–240.

Tobinski, D. A. (2014). Semantic Heat Map. *Unpublished paper.*

Towards a unified reasoning theory: An evaluation of the Human Reasoning Module in Spatial Reasoning

Matthias Frorath, Rebecca Albrecht, and Marco Ragni
{frorath, albrechr, ragni}@cs.uni-freiburg.de
Center for Cognitive Science
University of Freiburg, Germany

Abstract

Lately two modules which aim to improve the modeling of human reasoning have been introduced into the ACT-R cognitive architecture – the Human Reasoning Module (HRM) and the Pre-Attentive and Attentive Vision Module (PAAV). This is a first attempt to create a domain-specific infrastructure as an extension of a cognitive architecture. The HRM defines the basic functionality of two different theories in human reasoning – the mental model theory and the mental logic theory. Thus, it is a step towards a "unified theory of human reasoning". In this article we use a model of the continuity effect in spatial reasoning to evaluate this approach. The results show that the HRM is a clearly more convenient way to define ACT-R models for reasoning domains. However, as a unified theory of reasoning improvements are necessary before quantitative measures of all aspects in a reasoning task can be predicted.

Keywords: Spatial Reasoning; Human Reasoning Module; Cognitive modelling; ACT-R

Introduction

While navigating in a new environment, assembling furniture, or setting the table we always process spatial relational information. Consider the following abstract information:

Object A is to the left of Object B.

Object C is to the right of Object B.

The task of a reasoner is to infer a relation between the objects A and C (a *generation task*) or to test if a specific relation holds (a *verification task*). If only one arrangement (which we call in the following a *model*) can be built from the given information, we call it a *determinate problem*, otherwise an *indeterminate problem*. Indeterminate problems are empirically often more difficult than determinate problems (Johnson-Laird & Byrne, 1991). The way relational information is presented, e.g., if the information is presented *continuously*, e.g., A is to the left of B, B is to the left of C, C is to the left of D, *semi-continuously* e.g., B is to the left of C, A is to the left of B, C is to the left of D, or *discontinuously* e.g., A is to the left of B, C is to the left of D, B is to the left of C, has an impact on reasoning difficulty (e.g., Knauff, Rauh, Schlieder, & Strube, 1998; Ragni & Knauff, 2013). We call each sentence that contains relational information a *premise*. Please note that the relational information is identical, and only the order of the premises changes.

In order to explain human spatial reasoning processes various psychological theories have been proposed. Two major theories are the *mental model theory* (MMT, Johnson-Laird, 1983), that assumes that humans construct, inspect, and vary mental models; and the theory of mental logic, a *rule-based reasoning theory* (e.g., Rips, 1994). The mental model theory was recently extended by the *preferred mental model theory* and its computational model: PRISM (Ragni & Knauff, 2013). To evaluate the theories their predictions must be tested against empirical data. Consequently, we have implemented both theories (Brüssow, Ragni, Frorath, Konieczny, & Fangmeier, 2013) in the cognitive architecture ACT-R (Anderson, 2007). While predictions of the MMT can explain the data, this support is missing for the rule-based theory (Ragni, 2008).

ACT-R provides principally enough "structure", i.e., modules to model human spatial reasoning problems. Findings from psychometrics (e.g., the Block-Tapping task, Vandierendonck, Kemps, Fastame, & Szmalec, 2004) to neuroscience (Prado, Chadha, & Booth, 2011; Knauff, 2013; Ragni, Franzmeier, Wenczel, & Maier, 2014) support that humans use for spatial reasoning a specialized mental structure, e.g., an amodal or multimodal representation of objects as proposed by the MMT. Hence, researchers have proposed an extension, a specialized spatial module for ACT-R (e.g., Gunzelmann & Lyon, 2006; Lyon, Gunzelmann, & Gluck, 2008; Douglass, 2007).

Another recently introduced approach assumes two specialized ACT-R modules to model spatial relational reasoning, the *Pre-Attentive and Attentive Vision Module* (PAAV, Nyamsuren and Taatgen (2013)) and the *Human Reasoning Module* (HRM, Nyamsuren and Taatgen (2014)). The HRM was introduced with the goal "to create a unified theory of human reasoning". Its functionality includes to build models and to use inference rules (e.g., transitivity rules). The PAAV is an alternative for the default ACT-R vision module, and gives access to the newly introduced *Visual Short Term Memory* (VSTM), where analogical representations, i.e., mental models could be built.

The HRM has already been evaluated with regard to its ability to model the qualitative difference between determinate and indeterminate problems in spatial relational reasoning (e.g., the problems from Byrne & Johnson-Laird, 1989) in Nyamsuren and Taatgen (2014). Another aspect, as outlined above, is to test how the HRM deals with the way relational information is presented. Hence we created a model to analyze if performance differences induced by the continuity effect can be reproduced and which implications the continuity effect has on the processing of spatial reasoning.

The paper is structured as follows: Firstly, we summarize the core functionality provided by the PAAV and the HRM.

In a second step we introduce an empirical investigation of the continuity effect. Based on a model by Nyamsuren and Taatgen we created an extended cognitive model. This model was extended to solve problems from the empirical investigation of the continuity effect. The model's results are then compared with the empirical data. A discussion of additional properties or limitations of the modules for improving the predictions concludes the paper.

ACT-R Modules PAAV and HRM

The authors describe their intention for providing the *Human Reasoning Module* (*HRM*) as "a single system that can express different facets of reasoning", including deduction and induction, deterministic and probabilistic inference, using rules and mental models. As such, the HRM differs from most other ACT-R modules in two aspects. Firstly, the HRM includes a set of production rules instead of a mathematical function which models subsymbolic mechanisms. With these production rules the HRM is able to request other modules like the declarative module. Secondly, the HRM has direct access to the *Pre-Attentive and Attentive Vision Module's* (*PAAV*) *Visual Short Term Memory* (*VSTM*) for requesting task-specific information.

In the first step of the reasoning process the HRM relies on the VSTM's functionality to store objects. The VSTM is represented by a two-dimensional array and stores two types of information: firstly all information that was processed by the default visual buffer is placed automatically into the VSTM. Secondly, by an explicit request task objects are stored such that premise information are represented. The VSTM offers a limited capacity of objects (default: 4 objects) which are stored for a limited amount of time (default: 10 seconds). When the capacity is exceeded the oldest object is cleared from the VSTM. By requesting the visual-memory buffer a model can access an object in the VSTM. Accessing an object in the VSTM resets the associated time stamp.

The HRM uses a specific type of chunks, the *assertion* chunks, with three slots: a *property*, a *subject* and an *object*. For spatial reasoning, chunks like `(p1 ISA assertion property left-of subject A object B)` are used to represent the fact that 'object A is to the left of object B'.

During the reasoning process all premises are stored as assertion chunks in the declarative memory. As mentioned before, the HRM offers a set of production rules for two types of reasoning, a set for *forward reasoning* and a set for *backward reasoning*. Each reasoning process starts by sending a request to the *reasoner buffer*. Forward reasoning is used for generation tasks where a rather unspecified request is sent to the reasoner buffer. The task consists of infering some information that follows given the premises in the declarative memory. Backward reasoning is used for verification tasks where a mostly or completely specified assertion is sent to the reasoner buffer, e.g., specifying a subject and an object and the missing relation has to be inferred, or a completely specified assertion with property, subject and object is given that has to be verified or rejected. In our experiment we used a verification task, thus we concentrate on backward reasoning.

Backward reasoning is a three step process, the *backward reasoning pipeline*. In case one of the steps succeeds proving the assertion, the backward reasoning pipeline is stopped. In the first step (*bottom-up reasoning*) valid assertions are requested from the VSTM based on the request. One of the valid assertions is then randomly chosen. This step does not involve using production rule requests to the VSTM, but direct access to the VSTM. Thus accessing the VSTM does not cost time. In the second step (*declarative retrieval*) the assertion is requested from the declarative module. In the third step (*top-down reasoning*) inference rules (like transitivity, opposite rules) are used to prove the assertion. Inference rules and assertions are retrieved from the declarative memory and for checking the applicability of a inference rule the backward reasoning pipeline is requested recursively. Therefore, this step may include several requests to the VSTM and to the declarative memory.

Experiment

In the behavioral experiment semi-continuous (SC) and discontinuous (DC) spatial reasoning problems were tested (cp. Table 1). Semi-continuous problems consist of premises that do not have terms that appear in both, the second and the third premise. For integrating the third premise information from the first premise is necessary. Discontinuous problems, however, consist of premises that do not have terms that appear in both, the first and the second premise. As a result, at the time where the second premise is presented, no information is available how the new information can be integrated with the first premise. This leads to different complexities between SC and DC problems that are known as the premise order effect (Ragni & Knauff, 2013) or the continuity effect (Ehrlich & Johnson-Laird, 1982), respectively. The goal of this experiment is to allow for an in-depth evaluation of the presented ACT-R model and therefore the respective modules.

In the following sections we call terms or objects that appear in multiple premises *common terms*. Using the SC (right) condition from Table 1, the common term in the first and the second premise would be the term *C*.

Method

Participants. We tested fourty-five students from the University of Freiburg (27 female, mean age: 22.86 years) who received course credit or a nominal fee for their participation.

Design and Materials. In total each participant received 64 four-term series conclusion verification problems in randomized order. Each problem consisted of three premises and a conclusion. No indeterminate problems were tested, i.e., in all problems only one correct model could be constructed. Half of the problems were semi-continuous (SC), the other half were discontinuous (DC) problems. The 32 problems in each category included 16 problems with a cor-

Table 1: Empirically tested problem categories. Two problems have a semi-continuous premise order (SC), and the other two problems have a discontinuous premise order (DC). The first problem in each category demands to add terms in the model construction to the right of already included terms (right), the second problems to add terms to the left (left).

Category	Premise 1	Premise 2	Premise 3
SC (right)	B left of C	C left of D	A left of B
SC (left)	B left of C	A left of B	C left of D
DC (right)	A left of B	C left of D	B left of C
DC (left)	C left of D	A left of B	B left of C

rect and 16 problems with an incorrect conclusion. The 16 correct/incorrect problems in one problem category varied in the direction in which new terms need to be integrated in a partial model (to the left or to the right) (cf. Table 1).

Procedure. Participants were tested individually in a quite room using a computer that administered the experiment. Preceding the experiment participants received three practice trials with feedback. Each premise and conclusion was presented subsequently in a self-paced manner (indicated by a key-press). As premise terms fruit names were used, e.g., "The apple is to the left of the orange.". Participants were asked if a conclusion holds given the previously presented premises. The answer was given by pressing a key corresponding to a "yes" or "no" answer. Processing times for each premise and the conclusion were recorded as well as the given answer.

Results and Discussion

In the empirical analysis we used linear mixed-effect models. We examined a possible influence of the premise relations on the processing times, also known as the figural effect (Knauff, Rauh, & Schlieder, 1995). The investigated categories can be seen in Table 2.

Table 2: Analyzed problem categories for discontinuous problems. All four problems have a discontinuous premise order (DC). The first relation names the direction where terms of the second premise are inserted. The second relation (with -of) names the relation inside each premise.

Category	Premise 1	Premise 2	Premise 3
Right, Left-of	A left of B	C left of D	B left of C
Left, Left-of	C left of D	A left of B	B left of C
Right, Right-of	B right of A	D right of C	C right of B
Left, Right-of	D right of C	B right of A	C right of B

In each premise phase we did not find a significant difference between the four conditions. In the following data analysis we collapsed all four conditions to one single DC condition (premise 1: $F(3,38) = 1.09$, $p = 0.36$; premise 2: $F(3,38)$, $p = .11$; premise 3: $F(3,39) = 0.23$, $p = .88$). This was conducted analogously for the SC condition.

We analyzed differences in the processing time in each premise and the conclusion for semi-continuous (SC) and discontinuous (DC) problems. In the first premise ($F(1, 40) = .45$; $p = .51$), the second premise ($F(1, 42) = 2.81$; $p = .1$) and the conclusion ($F(1, 41) = .26$; $p = .61$) no significant differences in processing time between semi-continuous and discontinuous problems were found. In the third premise the processing time in semi-continuous problems was significantly shorter than in discontinuous problems ($F(1, 42) = 22.56$; $p < .0001$). Figure 1 shows a graphical overview of the results. These results support the assumption that in discontinuous problems additional processes, especially in the third premise, are necessary to process the respective premise and to integrate information into the mental model.

Additional to the differences in semi-continuous and discontinuous problems we investigated which strategy could be used in discontinuous problems. In these problems the first and second premises do not share a common term. As a result, no integrated representation can be constructed after the second premise was presented. Two different strategies have been proposed in the literature to handle this situation; (1) both premises are integrated into one mental model (e.g. adding the terms of the second premise to the right of the terms of the first premise), the information from the third premise then may result in a belief revision (Nejasmic, Krumnack, Bucher, & Knauff, 2011); (2) For both, the first and second, premises separate models are constructed, when the third premise is presented those models are integrated. In order to determine which strategy was used by the participants, we analyzed the response time in the third premise. In strategy (1) it was proposed that terms in the second premise are integrated into the mental model. If this information is inconsistent given the third premise a belief revision mechanism is used. As a result, the response time in the third premise should be significantly higher if belief revision is necessary (e.g., DC (right) vs. DC (left) in Table 1). This effect was not found ($F(1, 41) = 0$; $p = .98$).

ACT-R Model

Model Description

Original model by Nyamsuren and Taatgen (2014). They created a cognitive model to demonstrate the functionality of the HRM and the PAAV's VSTM in spatial reasoning tasks. For their demonstration Nyamsuren and Taatgen chose determinate and indeterminate five-term problems in a generation task setting with two-dimensional relations as introduced in Byrne and Johnson-Laird (1989). The model clearly focused on mental model construction in the VSTM and the reasoning processes in the HRM. Therefore, premises were placed directly in ACT-R's imaginal buffer. In order to represent all terms in one problem in the VSTM the capacity was increased to five objects.

For evaluating the quality of their model and, therefore, their modules they compared correctness in determinate and indeterminate problems with existing experiments (e.g., Byrne and Johnson-Laird (1989)). The model predicts 100% correctness in determinate problems. In indeterminate problems, due to several possible valid mental models, errors only occur during mental model modification.

An important aspect of Nyamsuren and Taatgen's model is the integration of new information into the VSTM. When the second premise is presented the terms of the first premise have already been included into the VSTM. In order to integrate the terms of the second premise the model checks whether the object of the corresponding assertion is already in the VSTM. If this is the case, the subject is integrated into the VSTM with regard to the relation represented by the property. If the object is NOT yet in the VSTM the *opposite* relation is requested from the HRM. As a result, the model makes some interesting predictions about the time to integrate premises into the VSTM.

Extension for the Continuity Effect. In order to model the continuity effect with the HRM and PAAV, we extended the cognitive model by adding the functionality necessary to solve discontinuous problems.

Instead of placing premises into the imaginal buffer directly, we used the ACT-R vision module to read premises and conclusions from the screen. Premises and conclusions are presented in the form of "A L B" representing the sentence "A is to the left of B". Similarly, in order to include the self-paced character of our experiment the model pressed keys (using the manual buffer) to view the next premise or the conclusion. As in the experiment processing times and answers were recorded.

We presented all premise information visually, so all chunks from the visual buffer were inserted into the VSTM automatically. This had an impact on the model's predictions. Including the manually added term information a maximal capacity of 13 objects was necessary. In order to counter this effect we extended the model by a rehearsal mechanism. Each time premise information is integrated in the VSTM and before the model views the next premise all manually added objects into the VSTM are rehearsed. As a result, only the visually presented premise information is removed from the VSTM and terms from the mental model are held in the VSTM. With this mechanism still a maximum capacity of 7 objects is necessary to hold the complete mental model in the VSTM. Note that the rehearsal mechanism has no significant impact on differences in the evaluated problem categories.

In order to also allow for the processing of discontinuous problems additional changes to the model were necessary. As explained in the Experiment Section there are two possible strategies how discontinuous information can be integrated. Empirical data suggests that premise information are only integrated after they can be linked by a common term, i.e., when the third premise has been presented. However, different implementations of this strategy are possible. We chose the strategy to keep the first premise inside the VSTM and to store the information from the second premise in an assertion chunk in the declarative memory. Once the third premise is presented and integrated into the VSTM the assertion chunk is recalled from the declarative memory. The model checks if there are now common terms in the assertion and the VSTM and when a common term is found, the information from this assertion is integrated into the VSTM.

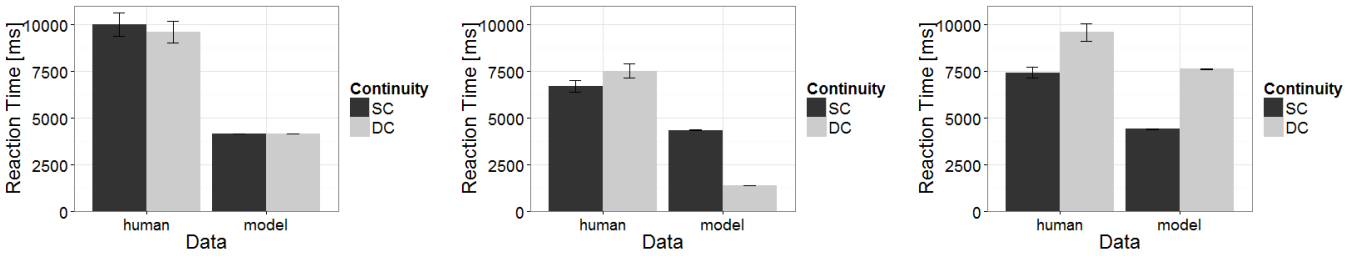
Model Evaluation

Premise processing times. Figure 1 shows a comparison between the model predictions and the empirical data for the processing time of the three premises in the semi- and discontinuous problems.

For evaluating our model for the continuity effect we compared model predictions with the overall response time and the processing time for all premises. In the premise processing phases the overall response times could not be predicted. The integration of all the premises is too fast compared to the human reaction times. For this reason we concentrated on comparing the qualitative reaction time differences. We found that for the first and the third premise the reaction time trends could be modeled. While the first premise has no significant difference for both continuities no differences could be found in the model predictions. For the third premise a significant difference between semi-continuous and discontinuous problems could be found. The model predicts higher processing times for discontinuous problems due to the integration of the postponed integration of the second premise.

For the second premise we did not found a significant difference between semi-continuous and discontinuous problems. The model, however, does predict a lower processing time for discontinuous problems than for semi-continuous problems. After noticing that no common term can be found in the VSTM the model stores the second premise in the declarative memory without integrating terms into the VSTM.

Correctness and Conclusion Answer Time. In order to evaluate our choice concerning the maximum capacity of the VSTM we used correctness as a measure. Table 3 shows the correctness for correct and incorrect conclusions of the empirical data (H) and for capacities of 4 (default) to 7. For incorrect conclusions the model predicts a correctness of 100% for each capacity. For correct conclusions the correctness drops from 100% to under 50% when the capacity is not sufficient to store all terms in a problem. There is no capacity which predicts the human correctness. The reason for this rapid drop in correctness is the switch between the mental model approach and the rule-based reasoning approach in the top-down reasoning process. Inference rules (e.g., transitivity) are used to validate a conclusion. This process includes several requests to the declarative module and, thus, is highly error prone. It can also be noted that with the current module implementation the correctness for incorrect conclusions cannot be lower than 100%. The reason is that there are no mecha-



(a) First premise. (b) Second premise. (c) Third premise

Figure 1: Processing time for the first, second, and third premise comparing human data and model predictions for semi-continuous problems (SC) and discontinuous problems (DC).

nisms to accept an incorrect conclusion in the top-down reasoning mechanism.

A comparison between the empirical data and the model predictions for a VSTM capacity of 7 shows that the time to reject an incorrect conclusion is significantly higher than to accept a correct one (cp. Fig. 2). In the latter case only the VSTM needs to be checked. In contrast for an incorrect conclusion all three steps including the top-down reasoning mechanism is necessary before a rejection.

Table 3: Proportions of correct answered problems in empirical data (H) and model predictions for VSTM capacities of 4 to 7 comparing correct and incorrect conclusions.

	H	4	5	6	7
Correct	0.74	0.24	0.27	0.47	1.0
Incorrect	0.84	1.0	1.0	1.0	1.0

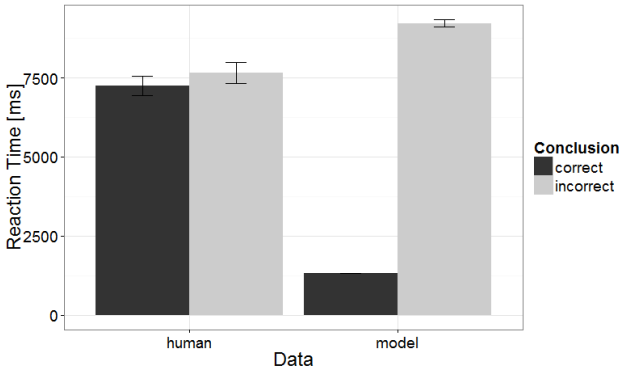


Figure 2: Time to process a conclusion and give an answer for correct and incorrect conclusions.

Conclusion

The Human-Reasoning Module (HRM) aims "to create a unified theory of human reasoning" (Nyamsuren & Taatgen, 2014). The module achieves this mainly by introducing two major changes to the ACT-R philosophy. (1) The HRM includes a set of production rules instead of a mathematical function which models subsymbolic mechanisms; (2) the HRM has direct access to other modules, especially the PAAV. (1) introduces a certain flow of information and control into the ACT-R system which cannot be influenced by the modeler. This introduction of limitations to the ACT-R architecture restricts what can be explained and adds effects that cannot be explained. Of course, introduced restrictions to a cognitive architecture must be theoretically sound and empirically validated. The aim of this paper is to evaluate these introducted restrictions by analyzing a model for the continuity effect.

The presented model is able to predict empirical data for the processing of the first and third premise, but not for the second premise. In discontinuous problems, the second premise cannot be integrated into the existing mental model. Our model stores this premise in declarative memory to recall it later. Other approaches, like the spatial buffer of Douglass (2007) or in the Spatial and Visual System of the cognitive architecture SOAR (Wintermute, 2009) use hierarchical spatial objects. An extension by such a mechanism should be evaluated.

Especially in the construction phase of the mental model overall response times could not be predicted. The model is too fast in the integration of all the premises. The HRM does not include any assumptions on this phase in spatial reasoning. The switch from a mental model-based approach to a rule-based approach only occurs when a conclusion must be validated. Thus, additional process assumptions should be considered, e.g., focus operations defined in the PRISM model (Ragni & Knauff, 2013).

The correctness could not be predicted as well for several evaluated capacities associated with the mental model representation. For incorrect conclusions the correctness is in each

case 100%. In case of correct conclusions the correctness drops from 100% to under 50% as soon as the capacity is not sufficient to hold all terms in a problem. Orthogonally, the response time for a correct conclusion is significantly lower than in the empirical data if all information is accessible in the VSTM. In order to address these issues a decay mechanism instead of a fixed capacity should be considered to limit the mental representation.

To conclude, the HRM and PAAV are interesting approaches to allow for a more convenient model definition and to introduce restrictions to ACT-R. Additional improvements are possible to better predict empirical effects in higher-level cognition using ACT-R.

References

Anderson, J. R. (2007). *How can the human mind occur in the physical universe?* New York: Oxford University Press.

Brüssow, S., Ragni, M., Frorath, M., Konieczny, L., & Fangmeier, T. (2013). Premise annotation in mental model construction: An ACT-R approach to processing indeterminacy in spatial relational reasoning. *Cognitive Systems Research*, *24*(1), 52–61.

Byrne, R. M., & Johnson-Laird, P. (1989). Spatial reasoning. *Journal of memory and language*, *28*(5), 564–575.

Douglass, S. A. (2007). *A computational model of situated action*. dissertation, Carnegie Mellon University.

Ehrlich, K., & Johnson-Laird, P. N. (1982). Spatial descriptions and referential continuity. *Journal of verbal learning and verbal behavior*, *21*(3), 296–306.

Gunzelmann, G., & Lyon, D. R. (2006). Qualitative and quantitative reasoning and instance-based learning in spatial orientation. In *Proceedings of the 28th annual conference of the cognitive science society* (pp. 303–308). Vancouver, British Columbia, Canada.

Johnson-Laird, P. N. (1983). *Mental models: Towards a cognitive science of language, inference, and consciousness*. Cambridge, MA, USA: Harvard University Press.

Johnson-Laird, P. N., & Byrne, R. M. J. (1991). *Deduction*. Hillsdale, NJ: Erlbaum.

Knauff, M. (2013). *Space to Reason: A Spatial Theory of Human Thought*. MIT Press.

Knauff, M., Rauh, R., & Schlieder, C. (1995). Preferred mental models in qualitative spatial reasoning: A cognitive assessment of allen's calculus. In *Proceedings of the seventeenth annual conference of the cognitive science society* (pp. 200–205).

Knauff, M., Rauh, R., Schlieder, C., & Strube, G. (1998). Continuity effect and figural bias in spatial relational inference. In *Proceedings of the twentieth annual conference of the cognitive science society* (pp. 573–578). Mahwah, NJ: Erlbaum.

Lyon, D. R., Gunzelmann, G., & Gluck, K. A. (2008). A computational model of spatial visualization capacity. *Cognitive Psychology*, *57*, 122–152.

Nejasmic, J., Krumnack, A., Bucher, L., & Knauff, M. (2011). Cognitive processes underlying the continuity effect in spatial reasoning. In *Proceedings of the 33rd annual conference of the cognitive science society* (pp. 1127–1132).

Nyamsuren, E., & Taatgen, N. A. (2013). Pre-attentive and attentive vision module. *Cognitive Systems Research*, *24*, 62–71.

Nyamsuren, E., & Taatgen, N. A. (2014). Human reasoning module. *Biologically Inspired Cognitive Architectures*, *8*, 1–18.

Prado, J., Chadha, A., & Booth, J. R. (2011). The brain network for deductive reasoning: a quantitative meta-analysis of 28 neuroimaging studies. *Journal of Cognitive Neuroscience*, *23*(11), 3483–3497.

Ragni, M. (2008). Human logic in spatial reasoning. In B. C. Love, K. McRae, & V. M. Sloutsky (Eds.), *Proceedings of the 30th Annual Conference of the Cognitive Science Society* (pp. 933–939). Austin, TX: Cognitive Science Society.

Ragni, M., Franzmeier, I., Wenczel, F., & Maier, S. (2014). The role of the posterior parietal cortex in relational reasoning. *Cognitive Processing*.

Ragni, M., & Knauff, M. (2013). A theory and a computational model of spatial reasoning with preferred mental models. *Psychological Review*, *120*(3), 561–588.

Rips, L. J. (1994). *The psychology of proof: Deductive reasoning in human thinking*. Cambridge, MA: The MIT Press.

Vandierendonck, A., Kemps, E., Fastame, M. C., & Szmalec, A. (2004). Working memory components of the corsi blocks task. *British Journal of Psychology*, *95*(1), 57–79.

Wintermute, S. (2009). *An overview of spatial processing in soar/svs* (Tech. Rep.). Technical Report CCA-TR-2009–01, Center for Cognitive Architecture, University of Michigan.

The Ship of Theseus: Using Mathematical and Computational Models for Predicting Identity Judgments

Tuna Cakar (e170996@metu.edu.tr)
Cognitive Science Department, Informatics Institute
Middle East Technical University, Ankara, TURKEY

Annette Hohenberger (hohenber@metu.edu.tr)
Cognitive Science Department, Informatics Institute
Middle East Technical University, Ankara, TURKEY

Keywords: identity judgments; paradoxical reasoning; Ship of Theseus; modeling of decisions.

Introduction

Reasoning processes have been one of the central targets for cognitive modeling. Modeling of reasoning processes appears as an even harder challenge during paradoxical conditions such as the Ship of Theseus paradox. This work attempts to model empirical data from a behavioral study on paradox resolution with different modeling techniques: discriminant analysis (DA), decision tree analysis and neural networks. While each method has its own advantages and disadvantages, this paper attempts to compare and to contrast these methods trying to select the best model for future work.

Identity judgments have long been at the center of philosophical debates, e.g., is a car still the same after being fixed after a serious accident? Beyond the philosophical debates on the nature of objects and the concept of identity, it has also been a matter of interest how laymen respond to the identity question under different circumstances. The present study focuses on a famous paradox from ancient Greece, the Ship of Theseus (Hall, 1998). Answers to this paradox have been predicted by a Conceptual Tendency Test (CTT) tapping the concept of "sameness". The main aim of this paper is to present and compare the results of three predictive models in terms of their accuracy (predictive success) and to discuss the theoretical basis of the findings.

The initial, empirical part of this work aims to understand how participants reason during resolution of a given paradox, namely, the Ship of Theseus (Clark, 2002). In a nutshell, a ship owned by Theseus has been renewed part by part over time. At the end, all of the parts of the old ship have been renewed (Ship A) and the removed parts were reassembled to build another ship (Ship B). Thus, there are two ships finally. The classical paradox is: Which ship is the ship of Theseus, the renewed one (Ship A) or the one that has been reassembled with the old parts (Ship B)? Ship A responses seem to reflect a "functionalist" position, i.e., the function of the ship has been preserved; whereas ship B responses seem to reflect an essentialist position, i.e., the physical essence of the ship has been preserved. The problem has been discussed by several philosophers and related to the concept of "sameness" or "identity" (Wiggins, 2001). It is plausible to assume that participants' decisions are determined by several dimensions involved in the critical concept at stake, among them spatiotemporal considerations: how long did the renewal and reassembly process take (short or long) and where did it take place (at a proximal or distal place)? (Rips *et al*., 2006; Scholl, 2007). Functionalist and essentialist positions could be affected by these parameters differently. Participants initially performed a Conceptual Tendency Test (CTT) in which they were asked to rate a set of propositions which are directly related to the core concept of "sameness/identity" involved in the paradox before answering the paradox (see method).

In this current work, we focus on two main research questions: (1) Do the identity judgments in the CTT contain conceptual cores that are influential during the reasoning process on the paradox? (2) Can the final decision of a participant be predicted by the CTT? Our hypotheses on paradox resolution are as follows:

H_1: Participants take spatiotemporal features into account while making decisions about judgments on identity of an object over time.

H_2: The final decisions of the participants to the paradox can be predicted by their response to the CTT.

Experimental Design & Data Collection

50 undergraduate and graduate students (25 female; age-range 19-28 years) were allocated to the two experimental conditions – high vs low spatiotemporal proximity (STP) – randomly. Fourty-eight propositions were prepared as image files and were randomly presented for 15 seconds on the computer screen by E-Prime 1.0. Participants were initially asked to rate a set of propositions (the CTT) which are directly related to the core concept involved in the paradox. Participants responded on a scale from 1 to 5 (where 1 corresponds to total agreement, 3 to neutral, and 5 to total disagreement). There were 24 proposition pairs half of which were phrased in terms of "same" ("A" for Turkish "ayni" ("same")), and half in terms of "different" ("F" for Turkish "farkli" ("different")), e.g., A: "A bicycle that has its pedals removed is the same"; F: "A piece of paper bent over 3 times is different". Each proposition was presented for 15 seconds. After the presentation of all propositions, participants were then presented with the Ship of Theseus

paradox in one of the STP conditions: in the high STP condition, the ship was renewed/reassembled over a short period of time (5 years) and at a neighboring port; in the high STP condition it was renewed/reassembled over a long period of time (50 years) and at a distant port. Participants were given unlimited time to respond on a 5-point Likert scale, where ratings of 1 and 2 were considered as strong and weak "ship A" ratings, ratings of 3 as "undecided" and ratings of 4 and 5 as weak and strong "Ship B" ratings.

Results & Discussion

Behavioral Results

The obtained responses to the Ship of Theseus question revealed a bimodal, M-like distribution (20 cases for ship A and 24 cases for ship B) for the paradox. The M-shape indicates that few subjects would take a strong stance and respond with the value 1 for A or 5 for B, respectively, but rather take a weak stance (2 or 4). In the middle of the distribution were 6 undecided participants. This result shows that participants avoided strict positions but rather stayed in a flexible zone while reasoning about the paradox. Responses for the two conditions (high STP vs low STP) were almost equally distributed. In other words, there was no effect of condition, contrary to our hypothesis. The initial statistical analysis on the Conceptual Tendency Test (one-way ANOVA) revealed that 4 different propositions (P17F, P21A, P6F, and P2F) reached significance and 6 further propositions (P9F, P10A, P10F, P14F, P17A, P6A) reached marginal significance ($p<.08$). Among the significant propositions was P17F, stating that a robot that had been disassembled and reassembled was different now; P2F: that two birds with identical genetic and behavioral features were different; P6F: that a piece of paper bent over 3 times was different; and P21A: that a robot with memory problems after a memory chip transplantation was the same.

Modeling Results

As a first mathematical model, Discriminant analysis (DA) was used to classify the responses to the paradox relying on the responses to the CTT. In DA groups of participants are discriminated based on linear combinations of variables. The initial discriminant analysis was run with 2 variables for the response to the paradox (Ship A or Ship B). Strong and weak positions for Ship A (1,2) and for Ship B (4,5) were therefore collapsed and intermediate positions (3) were eliminated in order to meet the statistical assumptions (Box's M-Test). Wilks' lambda was significant for the single function that the DA had computed (V=0.571, $\chi^2(9)$=21.020, p=.013). The canonical correlation that is a function of the eigenvalue was .655 for this function whereas the eigenvalue had the value of .751. 79.5% of the originally grouped cases were correctly classified. 6 out of 9 (66.6%) misclassified responses stemmed from weak positions (2 or 4) that were obviously harder to classify than strong positions (1 and 5).

As a second model, a decision tree analysis was performed for the same data based on two core propositions, namely P10F and P17F (both at significance level p=.002, Bonferroni-adjusted), resulting in 77.3% predictive success This decision tree consists of 5 nodes (3 of which are terminal nodes) and has the depth of 2. It is important to note that both of the propositions are 'different' (F) statements. This finding indicates that participants responded differently to the 'same' (A) versus 'different' (F) propositions. Interestingly enough, propositions with 'different' status were found to be more critical in predicting identity judgments.

As a third and last model the identity judgments were modeled with the neural network modeling technique (multilayer perceptron) relying on the same critical propositions of the CTT. 70% of the cases were used as training items and 30% as test items. The model was run with two units in a single hidden layer and the activation function was hyperbolic. The obtained Neural Network Model classified test cases with 88.9% predictive success.

Conclusion

Our modeling results revealed that the use of mathematical models like DA is beneficial in order to understand and explain the reasoning processes during paradox resolution like in the Ship of Theseus paradox – despite the fact that a computational model like the neural network model could predict the same data better. The present work demonstrates that the final judgments of the participants to the paradox could be predicted with a relatively high predictive success (>77%) solely relying on some critical propositions of a previously designed Conceptual Tendency Test (CTT). Participants tend to rely on a conceptual core about the target concept "sameness" which guides them through their reasoning process. Moreover, 'different' statements seem to play a more critical role in identity judgments when compared to 'same' statements. This finding suggests that these are two distinct cognitive processes even though they appear to be similar and participants were not aware of the fact that they responded differently to 'same' and 'different' statements. In conclusion, modeling is a worth-while methodology in order to better understand higher cognitive processes such as reasoning about paradoxes.

References

Clark, M. (2002). *Paradoxes from A to Z*. TJ International Ltd: New York.

Hall, D. G. (1998). Continuity and the persistence of objects: When the whole is greater than the sum of the parts. *Cognitive Psychology*, *37*, 28-59.

Rips, L. J., Blok, S., & Newman, G. (2006). Tracing the identity of objects. *Psychological Review*, *113*, 1-30.

Scholl, B. J. (2007). Object persistence in philosophy and psychology. *Mind & Language, 22*, 563-591.

Wiggins, D. (2001). *Sameness and Substance Renewed*. Cambridge: Cambridge University Press.

Modelling insight: The case of the nine-dot problem

Thomas C. Ormerod (t.ormerod@sussex.ac.uk)
School of Psychology, University of Sussex, Falmer, BN1 9QH UK

Patrice Rusconi (p.rusconi@surrey.ac.uk) & Adrian Banks (a.banks@surrey.ac.uk)
School of Psychology, University of Surrey, Guildford, GU2 7XH UK

James N. MacGregor (jmacgreg@uvic.ca)
School of Public Administration, University of Victoria, Victoria BC, Canada

Abstract

A number of frameworks for capturing insight phenomena have been proposed, but there are no executable models of knowledge-lean insight problem-solving. Here, an ACT-R model is presented for the nine-dot problem, which implements the Criterion for Satisfactory Progress theory for this problem. The model has two main components: a mechanism for searching for possible moves in the problem representation, and a mechanism for expanding the search to discover new moves not immediately available in the initial problem representation. The model accounts for key phenomena including impasse, fixation and the 'aha' moment, as well as predicting the relative difficulty of different problem variants.

Keywords: Insight; ACT-R; Problem-solving; 9-dot problem.

Introduction

Recent theories of insight are of two kinds: knowledge-based accounts such as Representational Change Theory (RCT: Knoblich et al., 1999), in which problem difficulty is mediated by inappropriate knowledge; and strategic accounts such as Criterion for Satisfactory Progress theory (CSP: MacGregor, Ormerod, & Chronicle, 2001), in which problem difficulty is mediated by search for moves that maximize progress towards a goal. Most researchers agree both knowledge and strategy are essential for explaining insight (e.g., Kershaw & Ohlsson, 2004), and integrated frameworks have been proposed (e.g., Hélie & Sun, 2010). However, progress is hampered by a lack of executable models of knowledge or strategy mechanisms.

An ACT-R model of 9-dot problem-solving

Here we present an ACT-R implementation of CSP for the 9-dot problem ("Draw four connected straight lines to cancel 9 dots arranged in a 3x3 grid"). The problem is notoriously difficult, with solution rates < 5%. Although knowledge-based accounts predict that a given first line extending beyond the array should serve as a solution cue, a first line remaining within the square leads to higher solution rates. An internal first line leads to earlier criterion failure, which motivates change of search strategy (MacGregor et al, 2001, Expts. 4-5). Our ACT-R model implements two heuristics: *maximisation* and *minimisation* (Chronicle, MacGregor, & Ormerod, 2004; MacGregor et al., 2001; Ormerod et al, 2013) to solve the problem.

Search through maximisation

Under maximisation, individuals select moves that appear most promising to achieve a hypothesised goal. Progress is monitored against a criterion derived from the problem statement. With the nine-dot problem, an initial line connecting three dots represents an implementation of a maximising heuristic because individuals cancel the most dots in a single move. Progress made with this move is evaluated against a criterion equal to the number of remaining dots divided by the number of remaining lines.

To implement maximization, the model searches for previously unattended and uncancelled dots at random and tests how many are cancelled by each move between dots. If it cancels more than the current best move, this move is stored in the imaginal buffer (where problem representations are stored). Then the cycle repeats until all unattended dots are inspected, when search for another one fails. This triggers a reset of all uncancelled dots to 'unattended'.

The line stored in the imaginal buffer represents the move that maximises progress. This line is checked against the progress-monitoring criterion. This criterion derives from two main sources of information in the initial representation: the number of dots and number of lines to be drawn. The criterion is equal to the number of remaining dots divided by the number of remaining lines. In the production (P PROMISING), if the number of cancelled dots is greater than the criterion, then the move is labelled as 'promising' (status slot of the imaginal chunk). Otherwise, in the production (P EXHAUSTED), there is criterion failure and the move is categorized as 'exhausted' in the 'status' slot of the imaginal chunk, and another best move is looked for. If the move has a promising status, the model draws a line. This is the first move. After a line is drawn, the model begins again the first cycle selecting previously unattended and uncancelled dots at random. The first stage stops either when there are no more dots to be cancelled or when the move count has reached the value of four and thus four moves are completed: except that it never does, without stage 2, the relaxation of the minimisation heuristic.

Discovery through minimisation

According to the minimisation heuristic, people limit a problem representation to the minimum required to achieve

satisfactory progress toward a goal. In the 9-dot problem, minimisation constrains the initial representation to the dot array presented in the initial problem (a grid of 3 x 3 dots). Relaxation of minimisation is triggered by criterion failure. Once relaxed, parsing the properties of previously explored moves, to identify invariants (cf. Kaplan & Simon, 1990) or unique move properties, derives new knowledge that can be used to infer possibilities for the discovery of new moves.

The model described in the first stage fails to find a criterion-satisfying fourth move, because the only places in the initial problem array correspond to dot coordinates. To 'learn' new places to look for moves, the minimisation heuristic needs to be relaxed. Relaxing the minimisation heuristic invokes a move parser that analyses the best moves produced to date and extracts properties that may enable discovery of new move types. In the four-line 9-dot problem, properties include space between known points, line lengths, and angles between lines. The model then uses these properties, in an order ranked according to principles of commonality, to discover new options to the current problem space based on inferences drawn from this knowledge (e.g., "if the most maximizing move currently has an average unit distance between cancelled dots of 1 unit, extend the line by 1 unit as a putative new move").

In this second stage, the model compares the properties of the lines drawn at the first stage. In the production (P COMPARE) it notices differences and invariants in terms of X and Y coordinates among the 'best moves' drawn. Based on these detected units of invariance among moves, the model, through the production (P EXTEND), uses the extracted units of invariance to extend the length of the first drawn line. In this way, the knowledge about properties extracted by comparing lines allows the problem space to be expanded to include (non-dot) spaces.

Phenomena captured by the model

Runs of the model provide ordinal differences between problem variants that are consistent with the published empirical literature on the problem. Like human solvers, it struggles to solve the problem (demonstrating impasse): in trials invoking 50 runs of the two-stage model, solution rates are less than 5%. It also returns, after attempts that extend beyond the 3x3 dot array, to exploring moves within the array (demonstrating fixation). However, it does solve on occasion (demonstrating the 'Aha' experience).

Also like human solvers, it easily solves (within 2 runs) the 13-dot variant in which the complete problem space is available in the initial representation, and finds solutions to 12- and 11-dot variants, where non-dot gaps within the dot array must be discovered, with increasing complexity but in runs < 10 (McGregor et al, 2001, Expt. 2). Finally, the implementation captures the difference between variants in which the first line is given, extending outside or within the initial dot array (McGregor et al, 2001, Expts. 4 and 5), with significantly fewer runs required for the latter than the former to discover solution, $p < .01$.

Discussion

The ACT-R implementation of CSP theory for the 9-dot problem demonstrates basic phenomena of insight captured by two simple heuristics governing search and expansion of an initial problem representation. Maximisation is a hill-climbing heuristic, while minimisation is a forcing function for discovering new problem knowledge based on recent discoveries of solution attempt properties. No additional knowledge is required, suggesting knowledge-rich accounts of insight (e.g., Knoblich et al., 1999; Kershaw & Ohlsson, 2004) may be overly elaborate for this particular problem.

Much remains to be done to provide a full implementation of knowledge-lean insight problem solving. Critically, the properties of the initial problem representation are hard-wired. Our hope is that the mechanism for minimisation can also be applied to parse the problem statement to build an initial representation. Building the ACT-R implementation raised new questions, such as whether maximization should be optimal (finding the very best move each run) or satisficing (finding the first criterion-satisficing move). These questions remain to be answered, but the growing ACT-R implementation provides a vehicle for doing so. In future work, we aim to extend the same principles to modeling other knowledge rich problems, such as the six-coin problem (Chronicle et al, 2004).

References

Chronicle, E. P., MacGregor, J.N., & Ormerod, T.C. (2004). What makes an insight problem? The roles of heuristics, goal conception, and solution recoding in knowledge-lean problems. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 30*, 14-27.

Hélie, S., & Sun, R. (2010). Incubation, insight, and creative problem solving: A unified theory and a connectionist model. *Psychological Review, 117*, 994-1024.

Kaplan, C. A., and Simon H.A. (1990). In search of insight. *Cognitive Psychology,* 22, 374-419.

Kershaw, T. C., & Ohlsson, S. (2004). Multiple causes of difficulty in insight: The case of the nine-dot problem. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 30, 3-13.

Knoblich, G., Ohlsson, S., Haider, H., & Rhenius,D. (1999). Constraint relaxation and chunk decomposition in insight problem solving. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 25*, 1534-1555.

MacGregor, J. N., Ormerod, T. C., & Chronicle, E. P. (2001). Information processing and insight: A process model of performance on the nine-dot and related problems. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 27*, 176-201.

Ormerod, T. C., MacGregor, J. N., Chronicle, E. P., Dewald, A. D., & Chu, Y. (2013). Act first, think later: The presence and absence of inferential planning in problem solving. *Memory & Cognition, 41,* 1096-1108.

Cognitive Models Predicting Surprise in Robot Operators

David Reitter (reitter@psu.edu), **Yang Xu (yang.xu@psu.edu)**
College of Information Sciences and Technology, Penn State, University Park, PA, USA

Patrick Craven (plc35@drexel.edu)
Computing & Informatics, Drexel University, Philadelphia, PA, USA

Anikó Sándor, Chris Garrett, E. Vince Cross, & Jerry L. Franke
Lockheed Martin Lockheed Martin Advanced Technology Laboratories, USA

Keywords: Spatial Path Planning, Robotics, Autonomous Vehicle Operators, Trust

Problem Statement

Robots and other autonomous vehicles have great utility, and even more potential: they go where human drivers can't breathe and where the lives of human pilots would be too costly to risk. Unfortunately, robots and their remote human operators do not always form a cohesive team. When robots make autonomous decisions, operators can be surprised and will, consequently, lose trust in the automation and end up micro-managing the robot. The benefits of partial autonomy are lost in the process.

The project discussed here evaluates ways to communicate robot reasoning to operators when needed (c.f., Kennedy et al., 2007). Its goal is to restore appropriate trust in automation without overloading the operator's attentional resources (c.f., Merritt et al., 2008).

Our approach assumes that misunderstandings between robot and operator are often due to differences in available information about the environment, different decision-making processes, and different levels of experience. Some sensory information may be withheld: the robot might know more than it visualizes, or the human is able to interpret a video feed more accurately than computer vision can. Further, decision-making algorithms and expertise are not synchronized; a robot may have un-inspectable machine-learning models, and an operator might have years of field experience.

A Robot that Explains Itself

Our objective is to enable the robot to convey pertinent information to improve monitoring performance and appropriate trust in the system, via the right modality and at the right time. The experiment discussed asks participants to interact with a system that can explain itself verbally. For instance, it may say *"I see a table and some glass shards on the ground. I planned a path around those obstacles"*. This explanation would allow the operator to accept the reasoning as is, verify it by referencing the simulated video feed, or reject it outright.

The system is designed to preempt operator surprisal by providing explanations at the best moment. It allows the operator to adopt a management-by-exception strategy: monitoring the autonomous vehicles rather than actively controlling them (Franke et al., 2005).

Experiment

The experiment has four conditions: *no explanations*, explanations only by *operator request*, *ongoing detailed* ones, and *selective* ones given when a cognitive model (see next section) detects operator surprisal.

Participants are asked to divide their time between the robot monitoring task (Figure 1) and a secondary, sensory analysis task, which draws away their visual attention. For the primary task, a standard exploration scenario is used with different rooms containing office furniture. It is implemented using a realistic robot simulation and operator interface (Gerkey et al., 2003).

Robots are evaluated by participants using a trust questionnaire (Merritt et al., 2008) and through neglect tolerance and preference ranking. We hypothesize that operators develop more trust in the three explanation conditions, and that they develop trust congruent with robot performance. We expect them to maintain the highest performance at both tasks only in the *selective* (model-driven) condition.

Increased trust is not necessarily a desirable as autonomous systems do make mistakes. In a second experiment, we will concentrate on *appropriate trust*. Here, participants are exposed to a high and a low-performing simulated robot per condition. The low-performing robot makes mistakes in identifying obstacles: it circumnavigates glass shards, while attempting to go through water, while its actual capabilities are the opposite (water only is to be avoided). The ensuing errors have to be corrected by the operator manually.

Application of a Path-Planning Model

How does the cognitive model predict operator surprisal? We have equipped our experiment system with a cognitive model formulated in ACT-R that predicts the operator's cognitive process in planning paths for the robot. When the robot's path deviates from the path that the model predicts, we detect potential for a surprise and issue an explanation. The experiment (see is designed to create situations in which the robot will misunderstand sensory information and plan an inappropriate path.

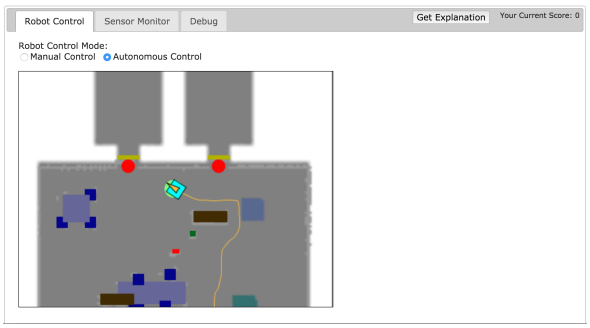


Figure 1: Operator interface, showing pre-defined goals at the top and a path taken by the robot around desks, chairs and some glass debris. In the condition shown, the subject may request a voice explanation.

The cognitive model (Reitter et al., 2010) has originally been developed to fit comparable data: waypoints set by robot operators to carry out an urban search&rescue task. In this setting, robots scout a contaminated office building, circumnavigate walls, cover all rooms, and discover all victims of a fictitious disaster. The insert in Figure 2 shows an itinerary defined by an operator, along with the corresponding model path.

The model predicts the plans an operator would develop for a robot to move from its given location to another given location. As a theoretical rational solution, one may think of a search process that guarantees the shortest workable path. (This standard robotics problem can be addressed via a standard A* algorithm or the more commonly used D*Lite (Koenig et al., 2005).) In contrast, the cognitive model predicts that human operators use a heuristic that selects the straight-line segment available from a given position that reduces the geometric distance to the goal; the initial choice is made at the starting position, and then the algorithm is applied recursively until the destination is reached or backtracking becomes necessary (for models of spatial navigation, compare Fum et al., 2000; Zhao et al., 2013).

The model explains scalability of the task with size of the environment as well as with cognitive load, such as when paths are to be planned for multiple robots (see Figure 2). It was evaluated with automatically generated mazes and on a dataset gained from robot operators that controlled 4, 8 or 12 simulated urban search&rescue robots at a time (Lewis et al., 2007).

Conclusions

As valuable as explanations may be, they can have a downside: cognitive overload and distraction. Therefore, our goal is to provide information when we believe it is necessary during the monitoring task. The experiment is designed to evaluate this approach.

The poster will present our analysis of the empirical results with 40 participants (the experiment has not been concluded at the time of writing).

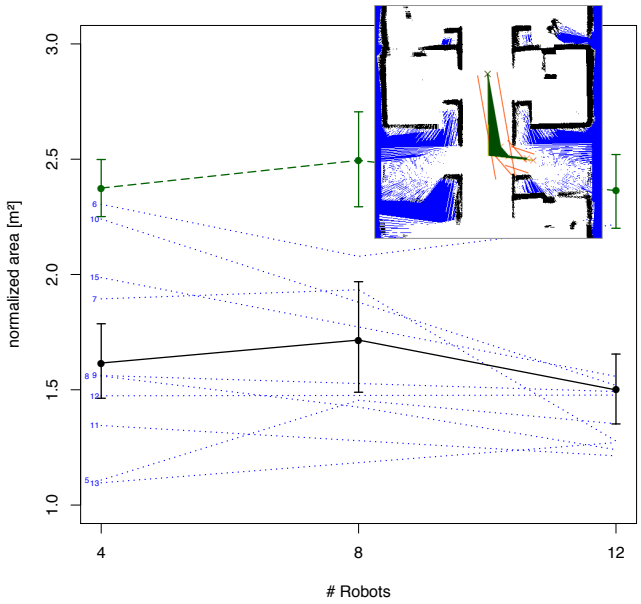


Figure 2: Average model error (solid), model vs. individual subject error (dotted), and avg. baseline model error (dashed) at different operator workload conditions. Insert: Example operator path for a robot (red crosses) and model's prediction for that path; difference area in solid green (from: Reitter et al., 2010).

With our approach, we do not design a cognitive model to fit new experimental results. Instead, we use a model that has been evaluated before as a means to predict the expectations of human operators in a realistic task relevant to national defense, safety and security. The experiment helps us analyze explanations as a means to affect trust in autonomous system. It also allows us to evaluate an ACT-R model in an extrinsic setting.

Acknowledgements

This work was funded by the Air Force Research Laboratory. We would like to thank Joe Lyons for his comments.

References

Franke, Jerry L et al. (2005). "Inverting the operator/vehicle ratio: Approaches to next generation UAV command and control". In: *Proc. of AUVSI Unmanned Systems North America* 2005.

Fum, D. and F. Del Missier (2000). "Climbing the mazes: A cognitive model of spatial planning". In: *In Proc. of the Third Int'l Conf on Cognitive Modeling*. Universal Press.

Gerkey, Brian, Richard T Vaughan, and Andrew Howard (2003). "The player/stage project: Tools for multi-robot and distributed sensor systems". In: *Proc. 11th Int'l Conference on Advanced Robotics*. Vol. 1, pp. 317–323.

Kennedy, William G et al. (2007). "Spatial representation and reasoning for human-robot collaboration". In: *AAAI*. Vol. 7, pp. 1554–1559.

Koenig, Sven and Maxim Likhachev (2005). "Fast replanning for navigation in unknown terrain". In: *Robotics, IEEE Transactions on* 21.3, pp. 354–363.

Lewis, Michael, Jijun Wang, and Stephen Hughes (2007). "USARSim: Simulation for the study of human-robot interaction". In: *Journal of Cognitive Engineering and Decision Making* 1.1, pp. 98–120.

Merritt, Stephanie M and Daniel R Ilgen (2008). "Not all trust is created equal: Dispositional and history-based trust in human-automation interactions". In: *Human Factors: The Journal of the Human Factors and Ergonomics Society* 50.2, pp. 194–210.

Reitter, David and Christian Lebiere (2010). "A Cognitive Model of Spatial Path Planning". In: *Computational and Mathematical Organization Theory* 16.3, pp. 220–245.

Zhao, Changkun, Jonathan H Morgan, and Frank E Ritter (2013). "Understanding human high-level spatial memory: An ACT-R model to integrate multi-level spatial cues and strategies". In: *Biologically Inspired Cognitive Architectures* 3, pp. 1–5.

Cue confusion and distractor prominence explain inconsistent effects of retrieval interference in human sentence processing

Felix Engelmann (felix.engelmann@uni-potsdam.de)
Department of Linguistics, University of Potsdam, Karl-Liebknecht-Str. 24/25
14476 Potsdam, Germany

Lena Jäger (lena.jaeger@uni-potsdam.de)
Department of Linguistics, University of Potsdam, Karl-Liebknecht-Str. 24/25
14476 Potsdam, Germany

Shravan Vasishth (vasishth@uni-potsdam.de)
Department of Linguistics, University of Potsdam, Karl-Liebknecht-Str. 24/25
14476 Potsdam, Germany

Keywords: sentence processing; retrieval; interference; ACT-R

The cognitive mechanisms underlying the processing of non-adjacent syntactic dependencies are critical for the understanding of human language processing. For instance, a verb needs to be syntactically and semantically integrated with its subject, or a reflexive like *himself* needs to be syntactically bound by its antecedent before it can be assigned any meaning. Thus, when processing the second part of a syntactic dependency, the parser needs to retrieve the corresponding first part of this dependency. The mechanisms underlying these syntactically triggered retrieval processes have drawn considerable attention in psycholinguistic research. Lewis and Vasishth (2005) (LV05) developed a model of sentence processing which is based on the general cognitive architecture ACT-R (Anderson et al., 2004). This model assumes a content-addressable memory in which cue-based retrieval processes are subject to similarity-based interference from (partially) cue-matching distractors. The LV05 model has widely been used to explain interference effects observed in the processing of syntactic dependencies such as reflexive-antecedent or subject-verb dependencies. Although the model is able to capture some of the empirically observed effects, there is a range of data the model is unable to explain. We propose to extend the LV05 model by two independently motivated assumptions, namely *cue confusion* and *activation-sensitive interference*. We demonstrate that this extended model explains a wide range of empirically observed effects the original LV05 model does not account for.

The LV05 model predicts that when retrieving the left part of a dependency (the target), a syntactically inaccessible noun phrase (distractor) that overlaps in features with the target noun phrase causes similarity-based interference, which leads to slowed processing (i.e., inhibitory interference). This is predicted, e.g., in the retrieval of a reflexive's antecedent as in (1) in Table 1, where the stereotypical gender on the target *surgeon* and on the distractor *Jonathan* both match the gender cue on the reflexive. By contrast, in (2), the stereotypical gender of the target *surgeon* mismatches the gender cue at the reflexive; here, a matching distractor is predicted to speed up processing by luring the parser into erroneous retrievals (facilitatory interference). Both effects are attested (e.g., Pearlmutter, Garnsey, & Bock, 1999; Badecker & Straub, 2002).

However, some studies have found facilitatory interference where inhibition was expected, and vice versa; other studies have failed to find interference effects. We developed a computational model extending LV05 by two independently motivated principles that can account for these apparently contradictory results. We show this in simulations that reproduce the patterns that were seen in a large-scale literature review.

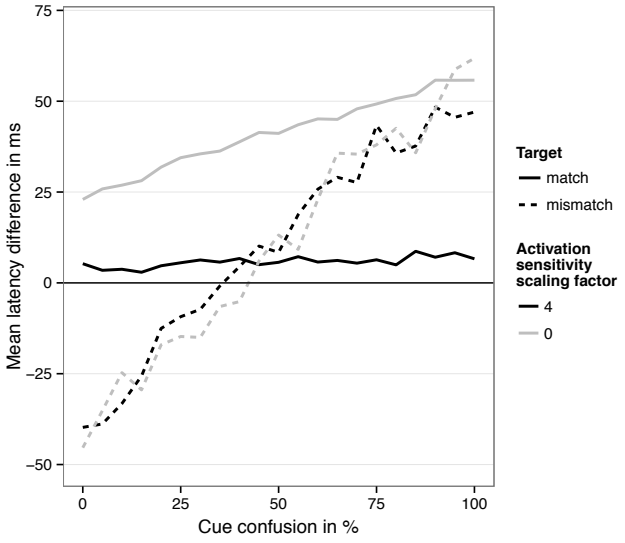


Figure 1: Predicted interference effects (interference condition - no-interference condition) by cue confusion for models without activation-sensitive interference (gray lines) and with activation-sensitivity scaling factor 4 (black lines). Solid lines represent the conditions where the target matches the semantic cue, mismatch conditions are represented by dashed lines. The LV05 model's prediction is shown by the gray lines at cue confusion of 0%.

Table 1: Gender-match/mismatch design commonly used in psycholinguistic experiments investigating interference effects in reflexives; example from Sturt (2003).

Match type	Example	Prediction (LV05)
(1) Target-Match	The **SURGEON**$^{+masc}_{+c\text{-}com}$ who treated [Jennifer$^{-masc}_{-c\text{-}com}$/**Jonathan**$^{+masc}_{-c\text{-}com}$] had pricked **HIMSELF**$^{masc}_{c\text{-}com}$	inhibition
(2) Target-Mismatch	The SURGEON$^{-fem}_{+c\text{-}com}$ who treated [Jonathan$^{-fem}_{-c\text{-}com}$/**Jennifer**$^{+fem}_{-c\text{-}com}$] had pricked **HERSELF**$^{fem}_{c\text{-}com}$	facilitation

Principle 1: Cue confusion

We assume that a retrieval cue can be associated with more than one feature. The strength of this association is represented on a continuous scale and is shaped by experience. If two retrieval cues co-occur frequently in a certain retrieval environment, each of the two cues becomes associated also with the feature matched by the other cue. E.g., the Mandarin reflexive *ziji* invariantly cues for the feature pair $\{^{anim}_{c\text{-}com}\}$. This co-occurrence leads to a certain *crossed association* between *c-com* and *anim*. The same would hold for the *c-com* and *plur* cues in reciprocals. By contrast, English reflexives vary in number and gender: $\{^{fem/masc,\ plur/sing}_{c\text{-}com}\}$, resulting in a stronger one-to-one association rather than a crossed association between *c-com*, number, and gender. With crossed cue-feature associations, similarity-based interference can arise between memory items that do not share the same features. This explains the inhibitory interference effects observed in Target-Mismatch in Mandarin reflexives (Jäger, Engelmann, & Vasishth, subm.) and Hindi reciprocals (Kush & Phillips, 2014).

Independently of cue co-occurrence, we suggest that the associative strength between cues and features is modulated by working memory capacity: A strong one-to-one association is assumed to involve cognitive effort, hence readers with lower working memory capacity experience more crossed associations, leading to inhibitory interference in Target-Mismatch, even in English reflexives, as has been observed by Cunnings and Felser (2013).

Principle 2: Activation-sensitive interference

The strength of similarity-based interference is assumed to be scaled by the activation difference between target and distractor. E.g., in Target-Match, the target activation is much higher than the distractor activation because the target is a perfect match to the retrieval cues, which reduces the interference effect induced by the distractor. Thus, the following three patterns can be explained by distractor activation (prominence): (i) the well-known "grammatical asymmetry" (Wagers, Lau, & Phillips, 2009): interference effects are found more reliably in Target-Mismatch than in Target-Match; (ii) inhibitory interference increases in Target-Match when the distractor is more active, e.g., when it is in a more prominent subject position (Badecker & Straub, 2002); and (iii) facilitatory interference in Target-Match (e.g., Cunnings & Felser, 2013) due to fast misretrievals masking the similarity-based interference when the distractor has an even higher activation than the target.

Conclusion

In summary, we show in a computational model how two independently motivated principles that extend LV05's cue-based retrieval theory provide a principled explanation of hitherto unexplained patterns in the literature on interference in dependency processing: *Cue confusion* accounts for unexplained inhibitory interference, and *activation-sensitive interference* explains the conditions under which interference effects disappear

References

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004, October). An integrated theory of the mind. *Psychological Review*, *111*(4), 1036–60.

Badecker, W., & Straub, K. (2002). The processing role of structural constraints on the interpretation of pronouns and anaphors. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *28*(4), 748–769.

Cunnings, I., & Felser, C. (2013). The role of working memory in the processing of reflexives. *Language and Cognitive Processes*, *28*(1-2), 188–219.

Kush, D., & Phillips, C. (2014). Local anaphor licensing in an SOV language: Implications for retrieval strategies. *Frontiers in Psychology*, *5*(1252).

Lewis, R. L., & Vasishth, S. (2005). An activation-based model of sentence processing as skilled memory retrieval. *Cognitive Science*, *29*(3), 375–419.

Pearlmutter, N. J., Garnsey, S. M., & Bock, K. (1999). Agreement processes in sentence comprehension. *Journal of Memory and Language*, *41*, 427–456.

Sturt, P. (2003). The time-course of the application of binding constraints in reference resolution. *Journal of Memory and Language*, *48*, 542–562.

Wagers, M. W., Lau, E. F., & Phillips, C. (2009). Agreement attraction in comprehension: Representations and processes. *Journal of Memory and Language*, *61*, 206–237.

A spreading activation model of a discrete free association task

Vencislav Popov (vencislav.popov@gmail.com)
Department of Cognitive Science and Psychology,
New Bulgarian University, 21 Montevideo Street
Sofia 1618, Bulgaria

Introduction

Researchers in psycholinguistics often assume that the frequency with which an associate is given to a cue in a discrete free association task ("What is the first word that comes to your mind in response to the word ROBIN"?) by a group of participants, reflects the association strength in the mental lexicon of each individual (Nelson, McEvoy, & Schreiber, 2004). Based on this assumption they use these group-level production frequencies to control experimental stimuli or to define experimental condition.

It is also assumed that associates are produced by spreading activation from the cue to its targets as a function of their association strength and that the pattern of association strength is roughly the same for all speakers of a language. Two questions are warranted: (1) How does the cognitive system choose a response among the activated associates, so that it produces a different response each time, while maintaining the overall frequency pattern? (2) Why do people produce different associates if they share the same associative network?

I present a simple model that explains how the same associative network might give rise to different responses, whose frequencies approximate the underlying individual association strengths. The model serves mainly as a proof-of-concept that frequencies obtained by group experiments can be used to infer individual association strength. It does not, however, aim to be a general model of semantic memory, nor does it aim to model any other experimental effects at this time.

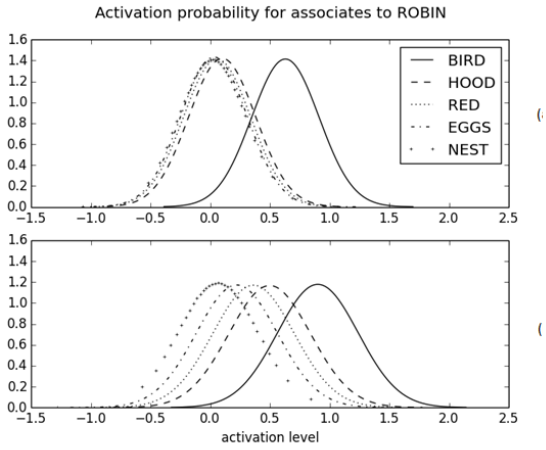


Figure 1. The probability of activation of each associate of the cue "ROBIN" in simulation 1 (a) with SD of the noise input 0.28 and in simulation 2 (b) with SD of the noise input 0.34

The free association model

In the model concepts are represented as single units, and the connection weights between them represent their association strength. If a cue is activated it spreads activation multiplied by the corresponding connection weight to all of its associates. Gaussian random error $\sim N(0, \sigma^2)$, which represents random input from the rest of the system, is added to the input of each associate. The most active node is selected as a response to the task. Predicted production frequencies are obtained by running the model N times and dividing the count of all unique responses by N and all runs of the model are independent of each other.

Evaluation of the model

The model was evaluated with the root mean square error of the prediction frequencies,

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}\left(freq_{observed,i} - freq_{predicted,i}\right)^2}{N}}$$

Goals of the simulations

Since the observed frequencies are interpreted to reflect connection strengths, then the connection strengths must be a function of those observed frequencies. The simulations had two goals: 1) to estimate the connection weight parameters from observed production frequencies ($w_{i,c} = f(FSG)$) and 2) to estimate the noise distribution in a way that would minimize the residuals of the predicted and the observed production frequencies.

FSG corpus

Data for the observed FSG was obtained from the **University of South Florida Free Association Norms**, a database of association norms for 5019 cue words (Nelson et al. 2004). Each cue was presented to a mean of 149 (SD = 15) people, who gave a single response to each of about 100-120 cues. The database is freely accessible at http://w3.usf.edu/FreeAssociation/

Simulations

Simulations 1 and 2

Simulations 1 and 2 tested which of two functions of the observed frequencies when used as connection weights and what dispersion of the noise would lead to a better

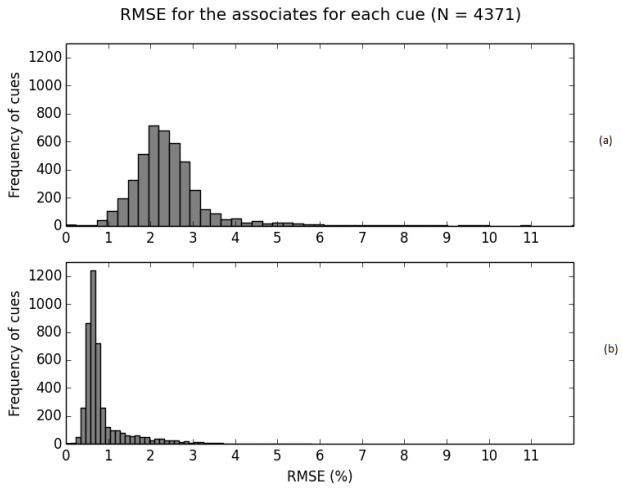


Figure 2. Frequencies of different RMSE levels for 4371 different cues for simulation 3 (a) and simulation 4 (b)

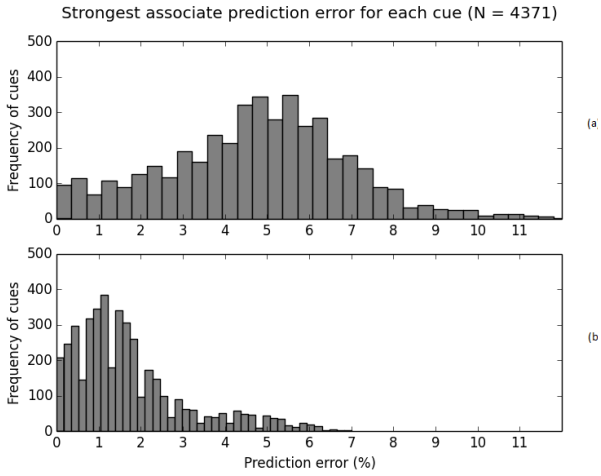


Figure 3. Frequencies of different prediction error of the strongest associate levels for 4371 different cues for simulation 3 (a) and simulation 4 (b)

approximation of the data. Simulation 1 tested the hypothesis that the individual connection weights are equal to the group-level production frequencies. Simulation 2 tests a model in which the connection strength in the mental lexicon is a logarithmic function of the observed production frequencies:

$$\frac{\log_{10}(freq_{observed,i} * 100)}{2}$$

Both simulations were run sequentially for 10000 times for each standard deviations of the noise input for all values from 0 to 1 with a step of 0.01. Both simulations fitted this parameter on the same subset of 10 randomly chosen cue words: 'TOMBSTONE', 'DOZEN', 'FEDERAL', 'REQUEST', 'BODY', 'LIFE', 'ROBBER', 'READ', 'WHISTLE', 'UNIVERSE'.

Results. Simulation 2 provided a much better fit of the data. Overall, in simulation 1, the RMSE was lowest when the noise standard deviation was equal to 0.28. In that case the predicted value of the model differed from the observed production frequencies by 2.4%, and the predicted value of the strongest associate – by 3.78%.

In simulation 2, when the weight of the connection between the cue and its associates is set to be equal to a logarithm of the observable production frequencies, the production frequencies of the model are closest to the observable production frequencies when the standard deviation of the noise input is equal to 0.34. The predicted frequencies of the model differ from the observed production frequencies by 0.72%, and the predicted value of the strongest associate – by 1.29%

Figure 1 presents a possible explanation for why the log transformation is more effective – it spreads the activation distributions of each associate further apart, which makes them more distinct.

Simulations 3 and 4

Both model 1 and model 2 were run on all standardized 4371 cues with SD of the noise input equal 0.28 and 0.34 respectively.

Results. Model 1 predicted the observed frequencies for the all 4371 cues with a 2.35% error rate. The model predicted the frequency of the strongest associate with a 5.24% prediction error. However, model 2 was again an even better predictor of the data for all 4371 cues – 99.15% overall successful prediction and 97.29% prediction success for the strongest associate. Also, the variance of the prediction error for all words (figure 2), and for the first associate (figure 3) was much smaller for model 2, compared to model 1, which makes its prediction much more reliable.

Discussion

This model provides a mechanism that can simulate the observable production frequencies of associates in a free association experiment with 0.85% prediction error, when activation is modeled as the spreading activation through a network in which the association strength is a logarithmic function of the observed production frequencies plus a Gaussian noise with a SD = 0.34. Importantly, this possibly validates the use of group-level production frequencies to estimate association strength between words in the individual lexicon. In this way it validates FSG's use in creating experimental conditions and its use as a control variable in psycholinguistics.

References

Nelson, D. L., McEvoy, C. L., & Schreiber, T. A. (2004). The University of South Florida free association, rhyme, and word fragment norms. *Behavior Research Methods, Instruments, & Computers*, *36*(3), 402–407.

Fail fast or succeed slowly: Good-enough processing can mask interference effects

Bruno Nicenboim (bruno.nicenboim@uni-potsdam.de) and **Felix Engelmann (felix.engelmann@uni-potsdam.de)**
Department of Linguistics, University of Potsdam
24-25 Karl-Liebknecht-Straße
Potsdam, 14476 Germany

Katja Suckow (katja.suckow@phil.uni-goe.de)
Department of German Studies,
University of Göttingen, 3 Käte-Hamburger-Weg 3
Göttingen, 37073 Germany

Shravan Vasishth (vasishth@uni-potsdam.de)
Department of Linguistics, University of Potsdam
24-25 Karl-Liebknecht-Straße
Potsdam, 14476 Germany

Keywords: cue-based retrieval; sentence processing; interference; multinomial processing trees; ACT-R

On a cue-based retrieval account of sentence processing (Van Dyke & Lewis, 2003; Vasishth & Lewis, 2006), grammatical heads such as verbs provide retrieval cues that are used to distinguish between the target item and competitors in memory. Similarity based interference occurs when items share retrieval cues, which makes it harder to distinguish between them, causing both longer reading times (RTs) and lower question-response accuracy. Since lower accuracy could be the result from either incorrectly retrieving a competitor or simply failing to complete a retrieval (an unstarted or aborted process), it is unclear how RTs are related to question-response accuracy. We investigated this question with two approaches: (i) by using the outcome of multinomial processing trees modeling accuracy in a linear mixed model with RTs as a dependent variable, and (ii) by fitting RTs and accuracy with ACT-R.

Experiment

In a self-paced reading experiment (N=84), we investigated interference effects in subject-verb dependencies in German by manipulating the number feature of two intervening competitor NPs (*the student/s of the teacher/s*). In the high interference (HI) condition, the two competitors share the feature singular (*sg*) with the target (*The driver*), while in the low interference (LI) condition the competitor NPs have, in contrast, the feature plural (*pl*). In order to investigate accuracy, we had yes-no questions targeting either the dependency between the subject and the embedded verb (*had transported*), or the dependency between the subject and the matrix verb (*sat*).

(1) a. HIGH INTERFERENCE

Der **Fahrer**, der den
The.sg.nom driver, who.sg.nom the.sg.acc
Schüler des Lehrers **transportiert**
student the.sg.gen teacher **transported**
hatte, saß angeschnallt im Bus.
had.sg, sat.sg using a belt in the bus.

'The driver, who had transported the student of the teacher, sat using a belt in the bus'

b. LOW INTERFERENCE

Der **Fahrer**, der die
The.sg.nom driver, who.sg.nom the.pl.acc
Schüler der Lehrer **transportiert**
student the.pl.gen teacher **transported**
hatte, saß angeschnallt im Bus.
had.sg, sat.sg using a belt in the bus.

'The driver, who had transported the students of the teachers, sat using a belt in the bus'

We found the expected retrieval interference effect: longer RTs in HI vs. LI at the embedded verb (Posterior Mean= 0.02; 95% Credible Interval $= [0.00, 0.04]$)[1] , as well as lower accuracy across question types in HI vs LI (PM= -0.40; 95% CI= $[-0.65, -0.16]$).

Multinomial Processing Trees

In order to investigate the relationship between latencies and question-response accuracy, we estimated the probability of successfully completing *any* retrieval at the embedded verb (R), the probability of the retrieval of the target conditional on R (C), and the bias to guess "Yes" (G). These estimations were carried out by fitting multinomial processing trees (MPT: Batchelder and Riefer, 1999) using Bayesian hierarchical modeling (Matzke, Dolan, Batchelder, & Wagenmakers, 2013). The model in Figure 1 postulates four processing trees depending on the correct answer and on the targeted verb. We estimated the parameters for the HI and LI conditions, assuming that G was independent of the manipulation. In order to reduce the number of parameters in the MPT, we further assumed no (or negligible) interference at the retrieval triggered by the main verb. If the parser completed a retrieval (even an incorrect one) at the embedded verb, a complete sentence representation will allow to give a correct answer for questions targeting the main verb. Even though it has been shown that already integrated nouns can interfere with subject retrieval of later verbs (Van Dyke, 2007), retrieval at the main verb may be easier here because only a subject-verb dependency has to be completed (in contrast to both subject-

[1] All the statistical analysis were done in the Stan probabilistic programming language. We report Bayesian linear mixed-effects models on $-1000/\text{RTs}$.

and object-verb dependencies for the embedded verb) and because the retrieval is facilitated by the use of cues such as attachment status and clause, which were unavailable for the embedded verb. Furthermore, we found no effect in RTs at the main verb, while the response question accuracy was significantly higher for questions targeting the main verb.

The MPT model revealed that both R and C were higher for LI conditions compared to HI conditions (see table 1). The difference in retrieval probability entails that more often in HI than in LI conditions, readers did not complete the dependency at the verb, and resorted to guessing at the stage of the comprehension question (in line with one possible conception of good-enough parsing; Ferreira, Bailey, & Ferraro, 2002). The model also yielded estimates of subject-level retrieval probabilities, which we regressed against RTs for each condition. The regressions showed that an increase in retrieval probability is associated with an increase in RTs (HI: PM= 0.05; 95% CI= $[0.00, 0.10]$; LI: PM= 0.08; 95% CI= $[0.02, 0.14]$). This suggests that a failed retrieval process is faster than a complete one. Taken together, these findings support the idea that at the locus of interference, the RT of each observation (for each subject) is generated by either fast good-enough parsing associated with a failed retrieval, or relatively slow, thorough parsing associated with retrieval completion. While HI produces latencies in retrieval completion in comparison with LI, it is also more likely that observations belonging to the HI condition will be generated by fast good-enough parsing. This suggests that in other experiments the selective good-enough parsing strategy associated with retrieval failure has the potential to mask interference if individual-level retrieval probability is ignored. Crucially, a linear mixed model including the estimates of retrieval as a covariate supports our hypothesis: We found a stronger effect of interference in our data when the individual-level measure of retrieval completion was included (PM= 0.04; 95% CI= $[0.01, 0.06]$).

Table 1: Parameters of the MPT model.

	Posterior Mean Probability	95% Credible Interval 2.5%	97.5%
R_{HI}	0.61	0.38	0.77
R_{LI}	0.81	0.68	0.90
C_{HI}	0.89	0.68	0.99
C_{LI}	0.95	0.85	1.00
G	0.69	0.54	0.82
R_{LI}-R_{HI}	0.20	0.02	0.42
C_{LI}-C_{HI}	0.06	−0.09	0.27

ACT-R

We implemented the assumption that readers follow a good-enough parsing strategy when a retrieval process fails using ACT-R (Anderson et al., 2004). The model implements the good-enough parsing by including an integration process of 150 ms only if the retrieval is completed. For each participant, we fitted a model to RTs and question-response accuracies by varying the retrieval threshold while keeping all other parameters fixed. The ACT-R model replicates the findings from the MPTs: higher retrieval probability and correct retrievals in LI in comparison with HI (0.78 vs. 0.67; 0.99 vs. 0.97), while it accounts for the observed RTs and accuracies.

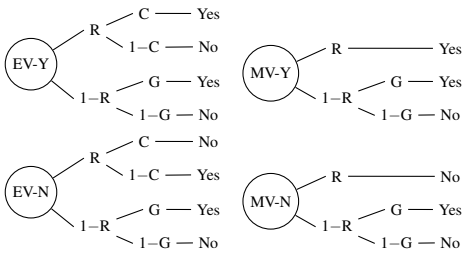


Figure 1: Multinomial processing trees. EV and MV indicate questions targeting the embedded verb and the main verb respectively; Y and N indicate whether the correct answer for the question was "Yes" or "No" respectively.

Conclusion

In sum, the results show that good-enough parsing, as construed above, may mask slowdowns due to interference, if both RTs and accuracy are not taken into account.

Acknowledgments

The work was supported by Minerva Foundation, Potsdam Graduate School, and the University of Potsdam.

References

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological Review*, *111*(4), 1036–1060.

Batchelder, W. H. & Riefer, D. M. (1999). Theoretical and empirical review of multinomial process tree modeling. *Psychonomic Bulletin & Review*, *6*(1), 57–86.

Ferreira, F., Bailey, K. G. D., & Ferraro, V. (2002). Good-enough representations in language comprehension. *Current Directions in Psychological Science*, *11*(1), 11–15.

Matzke, D., Dolan, C. V., Batchelder, W. H., & Wagenmakers, E.-J. (2013). Bayesian estimation of multinomial processing tree models with heterogeneity in participants and items. *Psychometrika*, 1–31.

Van Dyke, J. A. (2007). Interference effects from grammatically unavailable constituents during sentence processing. *Journal of Experimental Psychology. Learning, Memory, and Cognition*, *33*(2), 407.

Van Dyke, J. A. & Lewis, R. L. (2003). Distinguishing effects of structure and decay on attachment and repair: A cue-based parsing account of recovery from misanalyzed ambiguities. *Journal of Memory and Language*, *49*(3), 285–316.

Vasishth, S. & Lewis, R. L. (2006). Argument-head distance and processing complexity: Explaining both locality and antilocality effects. *Language*, *82*(4), 767–794.

Evaluating Instance-based Learning in Multi-cue Diagnosis

Christopher W. Myers[1], Kevin A. Gluck[1], Jack Harris[1], Vladislav Veksler[2], Thomas Mielke[3], & Rachel Boyd[4]
christopher.myers.29@us.af.mil, kevin.gluck@us.af.mil, jack.harris@us.af.mil, vdv718@gmail.com,
thomas.r.mielke@gmail.com, rachel.boyd.rb1@gmail.com

[1]*Air Force Research Laboratory* [2]*Army Research Laboratory* [3]*L3 Link Simulation at AFRL*
[4] *Oak Ridge Institute for Science & Education at AFRL*

Keywords: instance-based learning; fast and frugal trees

Introduction

Decision heuristics are often described as fast and frugal, meaning that they take little time and require relatively few computations to make a decision when compared to optimal decision systems (Gigerenzer & Todd, 1999). Fast & Frugal Trees are one heuristic that are a special case of decision trees in which there is a possible exit out of the decision process at every cue considered in the tree (Luan, Schooler, & Gigerenzer, 2011).

There is currently no computational account of how humans learn heuristics like F&FT-based decision processes. This is a significant gap in our scientific understanding, and we aim to begin addressing that gap in this effort. In this abstract we report results from a pilot study assessing Instance-based Learning Theory (IBLT) as an account of human learning from experience in domains where F&FTs may be good decision heuristics, such as diagnostic tasks.

Instance-based Learning Theory

Instance-based Learning Theory (IBLT) is a theory of how humans acquire and apply new knowledge given performance feedback and a particular context. It was developed to explain and understand human decision processes in dynamic task environments (Gonzalez, Lerch, & Lebiere, 2003) The four components of any IBLT model are (1) episodic memory elements (i.e., *instances*), (2) retrieving the instance from memory, (3) contextual similarity, and (4) integrating feedback across multiple, contextually similar events. In essence, an instance provides the utility of a particular action given a specified context in a way similar to expected utility theory

As far as we can tell, IBLT has not been applied to decision tasks where a set of different cues can be discriminately sampled for improving decision making. In the following section we introduce a multi-cue diagnosis task.

Multi-cue Diagnosis Task

The multi-cue diagnosis task is an extension of 2AFC tasks, where a decision-maker is provided with two alternative responses and a set of cues with which to inform the decision. Cues are binary (i.e., present or absent) and may be related to particular responses; part of decision-makers' task is to learn which cue(s) is (are) important.

In the task, there are three cues that a decision-maker can choose to use for determining a response. Cue information is not immediately visually available and requires clicking on a cue button to reveal its presence or absence. The decision maker is free to use any number of the cues in any order for informing their decision, and the only cost with accessing cue information was behavioral (i.e., moving to, clicking, etc.). Further, decision-makers were not speeded in their response and no penalty was issued based on trial response time. Given this basic task, we derived two environments: an easy environment (EZ) and data recreated from real-world CCU diagnoses (GnM; Green & Mehr, 1997). These environments were selected to provide approximate ceiling and floor performance in not only response accuracy, but also the adoption of prescribed F&FTs.

There were two payoff regimes: balanced (BAL) and heavy-miss (HM). In balanced, hits and correct-rejections received 10 points whereas misses and false alarms were penalized -10. The HM regime was the same as BAL except misses received -50.

In the pilot study reported here we ran a 2 (environment difficulty) x 2(payoff regime) between subjects design. We ran five participants through each of the four conditions. Each participant performed nine blocks of 30 trials. For each subject, on each trial, we captured their accuracy, the symptoms they revealed, and the order in which they were revealed. For each block the proportion of correct responses (i.e., accuracy), response time (RT), the proportion of selected responses (i.e., response selection), and the adherence to the prescribed F&FT (i.e., rule adherence) was calculated. Subjects' performance improved with experience in each condition and the EZ environment was easier than the GnM environment. Further subjects' RTs decreased as their acquisition and adherence to the prescribed rule increased (see Figure 1).

ACT-R Instance-based Learning Model

We developed an IBLT model in the ACT-R architecture (Anderson, 2007). We used ACT-R's declarative memory system to instantiate IBLT components one and two. We did not vary the degree of similarity between instances, instead opting for identity. This is justified as there was no hypothetical relationship between the binary cues. Finally, we used the ACT-R *blending* mechanism to instantiate IBLT component four.

The model used IBLT to determine the order of cues to check, when to stop checking, and which response to make. We believe this to be a novel use of IBLT, and the model represents complete adherence to the theory for execution.

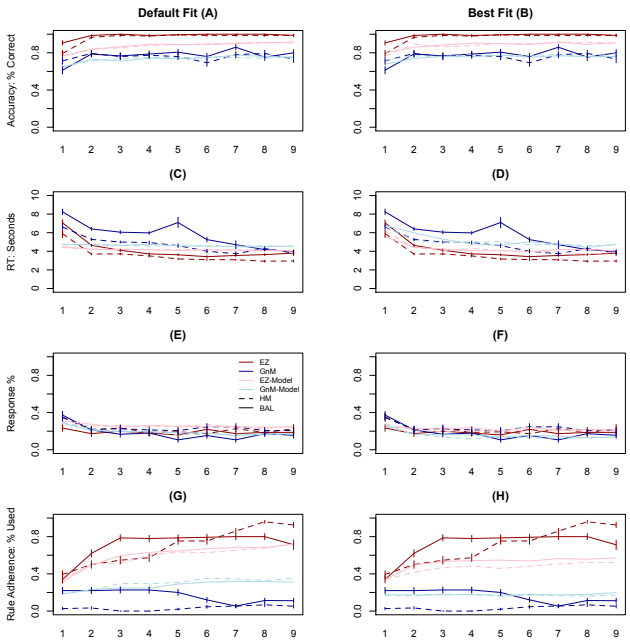


Figure 1: Human data and model results.

The model did not use production compilation nor production utility learning for acquiring any skill in the task environment.

Model Evaluation

We performed three evaluations of the model. For each evaluation, the model was run five times across 19 blocks of 30 trials, resetting after each run. The first evaluation (i.e., *default fit*) used parameters that were either default, taken from the central tendency of parameters in the Max Planck ACT-R parameter database (Wong, Cokely, & Schooler, 2010), or hypothesized where no guidance was available (see Table 1). The second and third evaluations (*best fit-all but RT* and *best fit-RT*, respectively) varied the retrieval threshold, blending temperature, activation noise, and decay parameters in a full combinatorial design producing 37,632 combinations of values. We only modified declarative memory and blending parameters as the investigation was on the adequacy of IBLT to account for multi-cue diagnosis tasks. Further, we report two different RMSEs for each evaluation: one for *RT* and another for the rest of the dependent variables (i.e., *Other*). We did this because the RT and the other dependent variables are on quite different scales.

The *best fit-all but RT* and *best fit-RT* surprisingly resulted in the same model parameters, and thus is referred to as *best-fit* (see Table 1, Best-fit column). The model performed quite well in the *default fit* evaluation, with an RT $RMSE = 1.162; R^2 = 0.552$ and an Other $RMSE = 0.682; R^2 = 0.914$. The model also performed well in the *best-fit* evaluation, with an RT $RMSE = 0.786; R^2 = 0.747$ and an Other $RMSE = 0.466; R^2 = 0.906$. Interestingly, with an improved fit in RT with best-fitting parameters (panel D, Figure 1) over the default (panel C), there is a reduction in rule adherence fitness (see panel H & G). Further, the :rt and :ans parameters are quite different from the central tendency of those reported by the community (see Table 1).

Parameter Name	Best-fit	Default	Source
decay (:bll)	0.1	0.4	MPIB-DB
base level constant (:blc)	0	1	Free
retrieval threshold (:rt)	-50	-0.4	MPIB-DB
activation noise (:ans)	0.75	0.4	MPIB-DB
blending temp	1	1	Free
imaginal-activation	1	1	Free

Table 1: Parameter values for the *default* and *best-fit* models. *MPIB-DB* refers to the Max Planck ACT-R Database and *Free* refers to hypothesized values due to no guidance on its setting. The source refers only to the *default* model parameters as the *best-fit* were derived by iterating over the large parameter space and minimizing RMSE. All other parameters were default values.

Conclusions

Generally, IBLT seems well suited for multi-cue diagnosis tasks. However, there appears to be a tradeoff between accounting for rule adherence and response times. Specifically, when fitting the model RTs, rule adherence decreased relative to the *default-fit* parameters. Consequently, we conclude that IBLT may not be sufficient to account for both RTs and rule adherence in this environment. Finally, the default fit model performed quite well, highlighting the value of making model parameter databases available to the community.

Acknowledgments

This work was supported by the Air Force Office of Scientific Research, grant 13RH06COR.

References

Anderson, J. R. (2007). *How can the human mind exist in the physical universe?* Oxford University Press.

Gigerenzer, G., & Todd, P. (1999). *Simple Heuristics That Make Us Smart*. Oxford, England: Oxford University Press.

Gonzalez, C., Lerch, F. J., & Lebiere, C. (2003). Instance-based learning in dynamic decision making. *Cognitive Science*(27), 591–635.

Green, L., & Mehr, D. R. (1997). What alters physicians' decisions to admit to the coronary care unit? *Journal of Family Practice*, *45*, 219–226.

Luan, S., Schooler, L. J., & Gigerenzer, G. (2011). A signal-detection analysis of fast-and-frugal trees. *Psychological Review*, *118*(2), 316–338.

Wong, T. J., Cokely, E. T., & Schooler, L. J. (2010). An Online Database of ACT-R Parameters : Towards a Transparent Community-based Approach to Model Development Managing Parameters for ACT-R Models. In *International conference of cognitive modeling* (Vol. 1, pp. 282–286).

The Influence of Cognitive Strategies on Performance in Working Memory Tasks

Menno Nijboer (m.nijboer@rug.nl)
Department of Artificial Intelligence, Nijenborgh 9
9747 AG Groningen, The Netherlands

Jelmer P. Borst (jelmerborst@gmail.com)
Department of Artificial Intelligence, Nijenborgh 9
9747 AG Groningen, The Netherlands

Hedderik van Rijn (hedderik@van-rijn.org)
Department of Experimental Psychology, Grote Kruisstraat 2/1
9712 TS Groningen, The Netherlands

Niels A. Taatgen (n.a.taatgen@rug.nl)
Department of Artificial Intelligence, Nijenborgh 9
9747 AG Groningen, The Netherlands

Keywords: cognition, multitasking, ACT-R, threaded cognition, computational modeling.

Introduction

Working memory is normally considered a capacity-limited system. This suggests that working memory performance is purely determined by the structure of the underlying architecture and the storage requirements of the task. Here, we argue instead that working memory performance is much more flexible and dependent on task-specific strategies.

Paradigm

In a concurrent dual-task, three working memory tasks were performed in pairs simultaneously. These tasks were an n-back task, a tone-counting task, and a spelling task that required the concatenation of individual letters to form a word. The tone-counting and spelling tasks were designed to be comparable on all aspects, and especially capacity. The difference between the two tasks is that compared to tone-counting, the spelling task was expected to require an additional memory resource to update the information in working memory when a new stimulus is presented. Crucially, this resource is also required by the n-back task.

Model

We used the ACT-R cognitive architecture (Anderson, 2007) to build a threaded cognition model (Salvucci & Taatgen, 2008) of the three tasks discussed previously. The crucial aspect of the model is the difference between spelling and tone-counting in terms of the cognitive working memory resources these tasks require. The spelling task model relies more on the problem state resource (Borst, Stocco, Van Rijn & Taatgen, 2010; Borst, Taatgen & Van Rijn, 2010), while the tone-counting task relies more on the declarative memory resource. Finally, the 2-back task uses both of these resources extensively. Previous research has shown that overlap in resource use leads to task interference (Nijboer, Borst, Van Rijn & Taatgen, 2014). Thus, our paradigm should produce a distinct interference pattern for the dual-task conditions, where spelling and tone-counting interfere strongly with the 2-back due to contention for resources. However, these tasks should interfere less with each other, due to the reliance on different resources.

Results

As presented in Figure 1, the amount of observed interference during the spelling task depended on the second task: The model replicates these results, which indicates that a difference in working memory strategy – without a difference in capacity requirements – can result in greater interference between tasks, as different resources are recruited. This suggests that there is a strategic and task dependent factor determining performance in working memory constrained tasks.

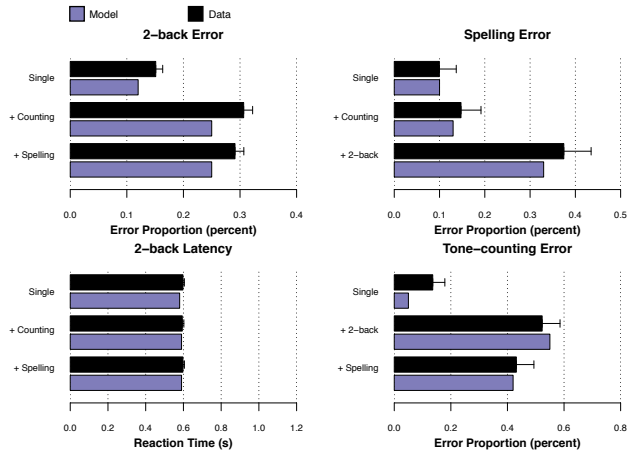


Figure 1: Behavioral results compared against model results.

Acknowledgments

This research was funded by ERC-StG grant 283597 awarded to Niels Taatgen.

References

Anderson, J. R. (2007) How Can the Human Mind Occur in the Physical Universe? New York: Oxford University Press.

Borst, J. P., Taatgen, N. A., Stocco, A., & Van Rijn, H. (2010). The Neural Correlates of Problem States: Testing fMRI Predictions of a Computational Model of Multitasking. *PLoS ONE*, 5(9), e12966.

Borst, J. P., Taatgen, N. A., & Van Rijn, H. (2010). The problem state: a cognitive bottleneck in multitasking. *Journal of Experimental Psychology. Learning, Memory, and Cognition*, 36(2), 363–382. doi:10.1037/a0018106

Nijboer, M., Borst, J.P., Van Rijn, H., & Taatgen, N.A. (2014). Single-task fMRI Overlap Predicts Concurrent Multitasking Interference. *NeuroImage* 100, 60-74.

Salvucci, D. D., & Taatgen, N. A. (2008). Threaded cognition: an integrated theory of concurrent multitasking. *Psychological Review*, 115(1), 101–130. doi:10.1037/0033-295X.115.1.101 Salvucci, D. D., & Taatgen, N. A. (2008). Threaded cognition: an integrated theory of concurrent multitasking. Psychological Review, 115(1), 101–130. doi:10.1037/0033-295X.115.1.101

Numerical Induction beyond Calculation: An fMRI Study in Combination with a Cognitive Model

Xiuqin Jia (xiuqin.jia@gmail.com)
Department of Radiology, Xuanwu Hospital, Capital Medical University, 45 Changchun Street, Xicheng District, Beijing 100053, China

Peipeng Liang (ppliang1979@gmail.com)
Department of Radiology, Xuanwu Hospital, Capital Medical University, 45 Changchun Street, Xicheng District, Beijing 100053, China

Xiaolan Fu (fuxl@psych.ac.cn)
Institute of Psychology, Chinese Academy of Sciences, 16 Lincui Road, Chaoyang District, Beijing 100101, China 100101

Kuncheng Li (cjr.likuncheng@vip.163.com)
Department of Radiology, Xuanwu Hospital, Capital Medical University, 45 Changchun Street, Xicheng District, Beijing 100053, China

Keywords: numerical reasoning; calculation; fMRI; ACT-R

Introduction

The ability to detect environmental regularities is a cognitive skill essential for survival. Human beings have a capacity, called numerical reasoning, to identify and extrapolate number serial patterns in such diverse areas as scientific discovery, economics, and the weather. Numerical reasoning and calculation have long been intimately associated, leading to the suggestion that they share a common system of the manipulation of numbers. The question of interest is whether numerical inductive reasoning is fully embedded in number calculation, or operates beyond calculation? To directly address these issues, we run an fMRI experiment to compare the number series completion task with the addition calculation task, and to understand the results in a cognitive architecture.

On the basis of previous researches, we hypothesized that the numerical reasoning compared to calculation was more activation in parietal areas for visual representation of relationship between numbers and prefrontal areas for relational integration.

In addition, computational cognitive modeling was employed to make specific predictions about the different processes of numerical reasoning and calculation. We will test our understanding of these processes by modeling the data within an information-processing theory called the adaptive control of thought-rational (ACT-R) (Anderson J, 2004; 2007).

Material and Method

Subjects and Stimuli

Fifteen paid healthy graduate students (8 females) with the mean age of 22.1 ± 2.3 years participated in the experiment. Three types of problems were organized into a block design, the numerical reasoning task (Rea), calculation (Cal), and judgment baseline (Jud) (see Fig. 1).

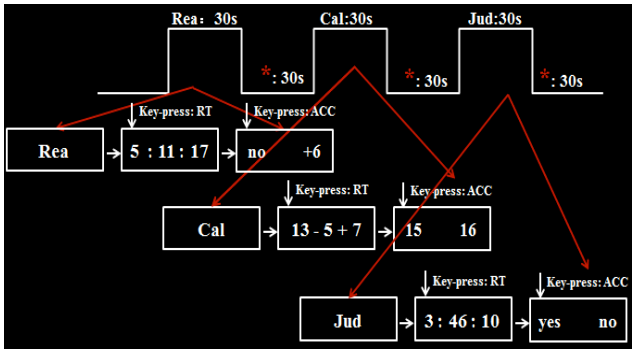


Fig. 1 Presentation paradigm of the stimuli.

FMRI Analysis

Data were analyzed using SPM5 software (http://www.fil.ion.ucl.ac.uk). Condition effects at each voxel were estimated according to the general linear model. The contrast of Rea vs. Jud would reveal regions for reasoning, the contrast of Rea vs. Cal would reveal regions specific to reasoning, and the contrast of Cal vs. Rea would reveal regions more involved in calculation. An uncorrected voxel-level intensity threshold of $p < 0.01$ with a minimum

cluster size of 27 contiguous voxels was used to correct for multiple comparisons using the AlphaSim method (http://afni.nimh.nih.gov/pub/dist/doc/manual/AlphaSim.pdf). This procedure yielded a corrected threshold of $p < 0.01$.

The Modeling

To understand the results in the frame work of the ACT-R theory, the model we constructed primarily depends on the visual module to perceive the stimuli, the manual module to respond, the retrieval module to retrieve a fact from memory, and the imaginal module to encode and update its stored representation.

Results and the Model

Behavioral Performance

We carried out analyses of variance for Rea, Cal and Jud on both RT and accuracy (Fig. 2, solid lines). Response to Rea task was significantly longer [$F(1,14) = 115.20$, $p < 0.001$; $F(1,14) = 12.37$, $p = 0.003$] and less accurate [$F(1,14) = 12.50$, $p = 0.003$; and $F(1,14] = 35.72$, $p < 0.001$] than that of Jud and Cal respectively. As shown in Fig. 2 (dotted lines), the model predictions of the behavioral results fit the data reasonably well.

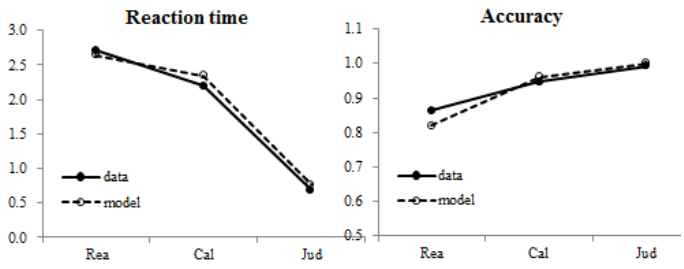


Fig. 2. Behavioral performance. Data (solid lines) and model fits (dotted lines) for Rea, Cal, and Jud problems.

FMRI Results

The Rea > Jud contrast revealed activation in the left dorsolateral prefrontal cortex (DLPFC), anterior cingulate cortex (ACC), bilateral intraparietal sulcus (IPS), and left occipital area (Fig. 3A). The Rea > Cal contrast revealed activation in the left DLPFC, precentral Gyrus, right superior parietal lobule (SPL), and left occipital Gyrus (Fig.3B), while the Cal > Rea contrast revealed activation in the bilateral thalamus, caudate, and posterior cingulate cortex (PCC) extending into cuneus (Fig.3C).

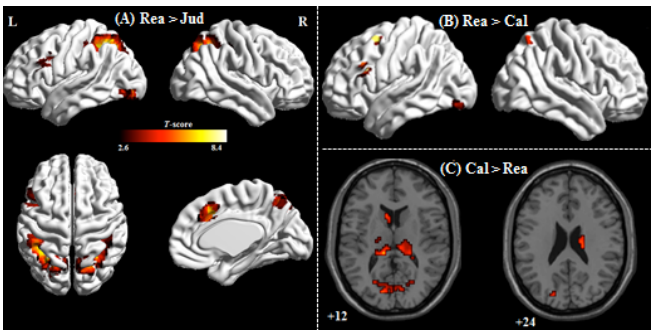


Fig. 3 Activation for the three contrasts

BOLD Responses

As shown in Fig. 4, four regions were of particular interest in this study: two regions of DLPFC and SPL specific to (Rea > Cal) (Fig. 4A); two regions of thalamus and caudate specific to (Cal >Rea) (Fig.4B).

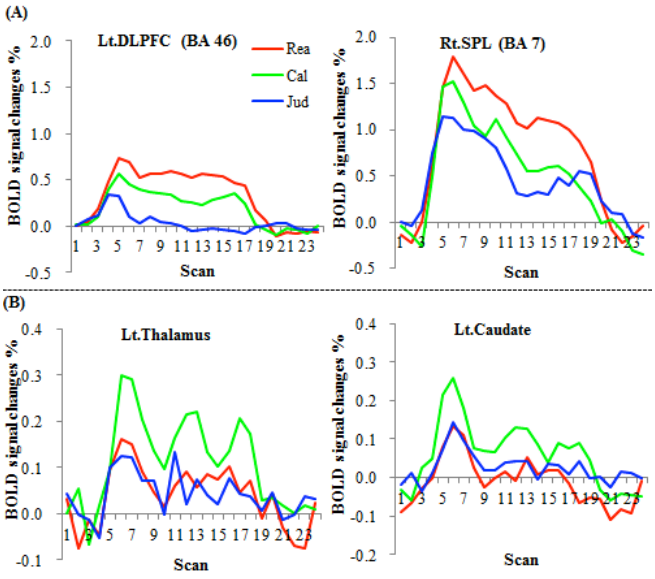


Fig. 4 BOLD responses for the ROIs.

Acknowledgments

This work was supported by the Natural Science Foundation of China (Grant Nos. 31400958, 61473196), and Key Projects in the National Science & Technology Pillar Program during the Twelfth Five-year Plan Period (2012BAI10B04).

References

Anderson JR, Bothell D, Byrne MD, Douglass S, Lebiere C, Qin Y (2004) An integrated theory of mind. Psychol Rev, 111: 1036-1060.

Anderson JR (2007) How can the human mind occur in the physical universe? New York: Oxford University Press.

Is it lie aversion, risk-aversion or tax audit aversion? Modeling deception under risk and no risk

Tei Laine (tlaine@msh-alpes.fr)
Maison des Sciences de l'Homme-Alpes, Université Pierre Mendès-France
1221 avenue centrale, Domaine universitaire, St Martin d'Hères, France

Tomi Silander (tomi.silander@xrce.xerox.com)
Xerox Research Centre Europe, 6 chemin de Maupertuis
38240 Meylan, France

Kayo Sakamoto (sakamotok@ihpc.a-star.edu.sg) **Ilya Farber (farberi@ihpc.a-star.edu.sg)**
Institute of High Performance Computing, A*STAR, 1 Fusionopolis Way, #16-16 Connexis North, Singapore 138632

Abstract

We studied deceptive decision making in hypothetical scenarios that involved risk of being caught of deceiving, or a penalty after being caught of deceiving, or both. We found that the deception rate was the lowest in the scenarios involving both the risk and the penalty. Our hierarchical model for deception suggests that in balancing the possible benefits from deception, the personal discomfort of getting caught is as large or larger than the inherent aversion to deception.

Keywords: Decision making; risk attitudes; deception; incentives; MTurk.

Introduction

In our recent study that asked participants' choices between risky and certain options (in the context of tax return), which either involved deception (deception condition) or did not involve deception (gamble condition), we observed a high rate of deception aversion[1] in the condition in which deception resulted in the better outcome than being honest, but involved both risk, i.e., non-zero probability of detection, and a potential loss in the form of a tax penalty, if the deception was detected (Laine, Sakamoto, & Silander, 2013). The participants who were particularly deception averse in the deception condition were also more risk averse than others in the gamble condition, which had equivalent risky and certain outcomes, but involved no deception.[2] In other words, the participants who took more risk in gamble condition, also deceived more in the deception condition.

We speculated if the reason for such a high level of tax compliance in the risky deception condition was exceptionally high level of risk aversion or exceptionally high level of deception aversion, or alternatively the task domain combined with the participant pool characteristics. We used Amazon MTurk workers from the US. This is a group of individuals who are willing to do simple tasks for little monetary compensation. Alternatively, based on their own prior experiences or knowledge of others' encounters with the Internal Revenue Service (IRS), our participants (most of them US tax payers) may have wanted to avoid any friction (even hypothetical) with the tax authorities, and indicated their willingness to pay due taxes, even in the presence of substantial financial incentives for evasion.

To rule out the explanation pertaining to the task domain and the participant pool, we conducted another study with MTurk participants, and added conditions from which we excluded either the risk or the tax penalty. Again we observed an exceptionally high rate of deception aversion in the condition that involved both a non-zero probability of detection and and a penalty after the deception was detected. Thus, it seems that the "IRS aversion", in other words the aversion to a potential audit by the tax authorities, is not alone enough to explain the high level of tax compliance, since the participants demonstrated some willingness to evade taxes in conditions from which either the risk or the penalty for detected deception was absent. In this study we wanted to find out what distinguishes those participants who refused to deceive in their taxes no matter what from those who properly incentivized switched from complete tax compliance to some degree of tax evasion.

Who are those who do X, where X $\in$ {take risk, deceive, evade taxes}

In general, people tend to be risk averse when facing gains and risk seeking when facing losses (Holt & Laury, 2002; Kahneman & Tversky, 1979). Many economic models of choice behavior are based on the concept of individual risk attitude, which can be measured experimentally and modelled with the shape and parameters of a utility function (Holt & Laury, 2002; Isaac & James, 2000; Weber, 1998). It has been considered a stable construct similar to personality traits, which drives behavioral patterns across situations (Blais & Weber, 2006).

However, this interpretation is problematic, since several studies have found that the risk attitude varies across task types (e.g., hypothetical vs. real outcomes) (Holt & Laury, 2002; Taylor, 2013), domains (e.g., financial or health related decision making), elicitation methods (e.g., choice between

[1] In this experiment 279 (42%) out of 672 participants did not choose the risky deceptive option a single time in the deception condition.

[2] Only 25 out 672 participants never chose the risky option in the gamble condition.

gambles, a questionnaire, an auction, or a multiple price list method (Berg, Dickhaut, & McCabe, 2005; Charness, Gneezy, & Imas, 2013; Crosetto & Filippin, 2013)), and even ages (Harbaugh, Krause, & Vesterlund, 2002). For instance, Isaac and James (2000) demonstrated in a within participant study using two different risk elicitation tasks that some participants could in an instant turn from being risk averse to risk seeking, whereas some others remained risk neutral in both tasks. Blais and Weber (2006) suggest that even if an individual's risk attitude towards a perceived risk does not differ from one domain to another or from one situation to another, her risk-taking behavior might, if she perceives the risk and the benefits to be different in those situations.

There are differences between genders, too. Harris, Jenkins, and Glaser (2006) found that women's lower engagement rate in risky activities correlates with their tendency to judge negative events more likely and expected enjoyment of risky activities less highly than men in gambling, recreation, and health domains. However, there was no gender differences in risk-taking and risk perception in the social domain. Mixed findings have been reported on age differences in risk-taking propensity. Many studies have found more risk aversion in older people, but also the opposite in certain circumstances, or even no differences between age groups (see for instance: Deakin, Aitken, Robbins, & Sahakian, 2004; Dror, Katona, & Mungkur, 1998; Harbaugh et al., 2002; Huang, Wood, Berger, & Hanoch, 2013; Mather, 2006).

How do these studies relate to age and gender differences in deceptive behavior? Studies with adults have shown that while women are more lie averse than men in general, they are more likely to lie if it benefits others, and less likely if it hurts others, whereas men lie more for self-serving purposes, particularly monetary gains (dePaulo, Kashy, Kirkendol, & Wyer, 1996; Dreber & Johannesson, 2008; Erat & Gneezy, 2012). Finally, even if there is considerable heterogeneity in tax evasion within any group defined by a demographic category such as income or age, studies have shown that men evade taxes more than women, high income people evade taxes less than low income people, and married and under 65-year-old tax payers evade more than others (Slemrod, 2007).

These findings at least partially suggest that there is a link between risk-attitude and propensity to deceive, even to certain extent in taxes. However, most studies have focused on individuals or groups who partake in behaviors or activities in question (acts of commission), whereas in the following we are interested in those who do not (acts of omission).

Experimental design

According to the standard economic model of rational and selfish human behavior (i.e., the "homo economicus"), one should deceive if it is beneficial compared to being honest, and the decision should be solely determined by the trade-off between the gain from lying and the cost incurred if detected, given the probability of detection (Abeler, Becker, & Falk, 2014; Gneezy, Rockenbach, & Serra-Garcia, 2013). Therefore, the policies to curb deception, for instance in tax returns, have almost solely focused on increasing the detection probability and the penalty. However, these measures are not necessarily effective if people's deceptive behavior is driven by internal rewards instead of cost-benefit analysis of external rewards (Mazar, Amir, & Ariely, 2008); sometimes otherwise inconceivable indisposition to deceive have been attributed to factors like pure lie aversion (Fischbacher & Heusi, 2008; Gneezy, 2005; Gneezy et al., 2013; Erat & Gneezy, 2012; López-Pérez & Spiegelman, 2012; Lundquist, Ellingsen, Gribbe, & Johannesson, 2009), altruism (Abeler et al., 2014), maintenance of positive self-image, e.g., avoiding to appear greedy (Mazar et al., 2008; Fischbacher & Heusi, 2008), and moral considerations based on the norms and values of the society (Mazar et al., 2008; Sip et al., 2012).

Since our primary goal was to study the role of risk and monetary incentives in deception, we wanted to rule out the above factors. Instead, to more efficiently isolate the effect of risk from the effect of the outcomes, we made the expected value of the risky deceptive option higher than the expected value of the non-deceptive option, and added two conditions in which—still maintaining the expected value difference—either (1) both deception and being honest resulted in a certain outcome, i.e., there was no risk, or (2) failing the deception resulted in the same outcome as being honest, i.e., there was no penalty for detected deception.

Method

Participants We recruited 372 participants in Amazon MTurk to complete an online questionnaire in Qualtrics. After discarding data from participants who either did not complete the whole questionnaire or failed the attention check question we had 301 participants (130 women, 170 men, one unknown; median age 30 years, age range: 18-71 years). All participants were native English speakers, aged 18 or above, residing in the US. Each participant received USD 1.00 for their participation.

Material Using filling in a tax return as the deceptive domain, we prepared six gain-facing and six loss-facing scenarios in three conditions:

VarRisk Variable risk and constant outcome from successful deception (gain of $1000 and loss of $0), and from being truthful or unsuccessful deception (gain of $0 and loss of -$1000). We used detection probabilities of 0.01, 0.1, 0.25, 0.5, 0.75, and 0.90.

VarOut Variable outcomes and constant risk at 0% detection probability.

VarBoth Both variable outcomes and variable risk.

The **VarBoth** condition matched the **VarOut** condition in the expected values of options (see Table 1), and the **VarRisk** condition in the detection probabilities. This design allowed us to address the effects of risk and expected utility separately.

Example scenarios in **VarBoth** and **VarOut** in the gain condition are shown in Figure 1.

Table 1: Expected values in **VarBoth** and **VarOut** conditions for gains and losses.

Gains	Honest option $	891	810	675	450	225	90
	Deceptive option $	990	900	750	500	250	100
Losses	Honest option $	11	110	275	550	825	990
	Deceptive option $	10	100	250	500	750	900

VarBoth scenario Jenny is doing her yearly taxes. In addition to her regular job, she has done some part-time work on the side. Jenny is aware that she needs to report all of her income in her tax return.

If she omits the extra part-time earnings from her report, she will get back $1,000. If she includes these extra earnings in her report, she will get back $675. The part-time work was done for a small business, which might or might not report this payment to the IRS. If Jenny fails to declare the income, but the business reported it, she will lose her entire tax refund as a penalty, meaning that she will get $0. But since there's no way to prove that it wasn't just an accounting mistake, there is no other fine or long-term penalty for getting caught.

From talking to other people who've worked at the same place, Jenny estimates that there is a 25% chance that the business reported the payment. If you were Jenny, would you

[] Report the extra income and get a refund of $675.

[] Not report the extra income, in which case you have a 75% chance of getting back $1,000, but also a 25% chance of getting back $0.

VarOut scenario John is doing his yearly taxes. In addition to his regular job, he has done some part-time work on the side. John is aware that he needs to report all of his income in his tax return. If he omits the extra part-time earnings from his report, he will get back $750. If he includes these extra earnings in his report, he will get back $675.

The part-time work was paid in cash and the employer didn't record his name. Therefore, John knows that there is no chance that the IRS knows about this income. If he leaves it out of the report, there is no chance that he will be caught. If you were John, would you

[] Report the extra income and get a refund of $675.

[] Not report the extra income and get a refund of $750.

Figure 1: Example questions in **VarOut** gain and **VarBoth** gain conditions.

Procedure After giving their informed consent the participants were asked to make their choices in six sets of six questions (the order of the sets was randomized for each participant.[3] All participants answered all 36 questions, so the experimental condition manipulation (**VarBoth** vs. **VarOut** vs. **VarRisk**) was within participant. After finishing the choice questionnaire they answer a set of 30 risk- and deception attitude questions and completed a brief numeracy test, results of which are not reported here. They finished by filling in optional background information, including age, gender, and education. The questionnaire ended with a debriefing. It took them 15 minutes on average to finish the whole experiment.

[3]We prepared two versions of each scenario, one with a female and one with a male tax payer, and picked one randomly for each participant.

Results

In both **VarOut** and **VarRisk** conditions we observed much more deception than in the **VarBoth** condition, see Figures 2 and 3.

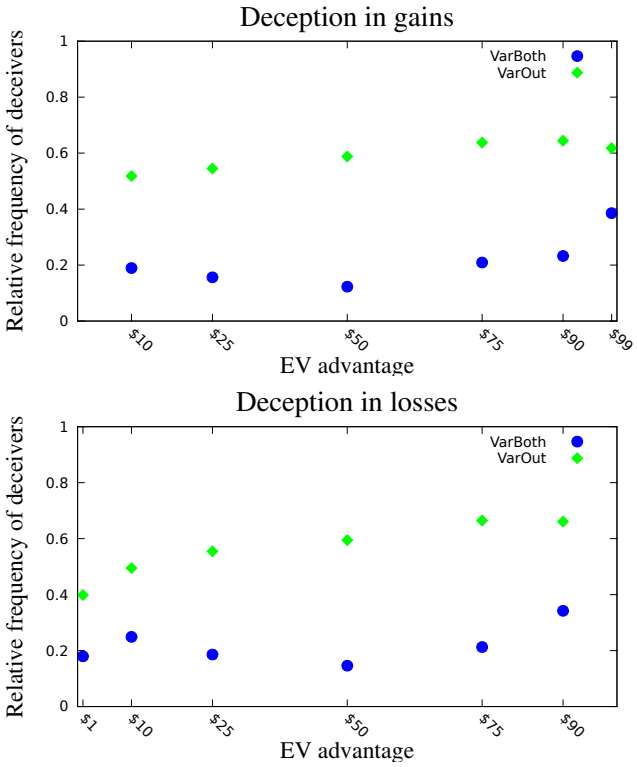


Figure 2: Relative frequencies of deceivers in the conditions varying only outcome, or both risk and outcome.

Risk aversion vs. penalty aversion The percentage of participants who never deceived was much higher in the **VarBoth** condition than in the other two conditions (see Table 2), and roughly corresponded to the percentage observed in Laine et al. (2013) (37% compared to 42%). The number of non-deceivers in the **VarBoth** condition significantly differed from the other two conditions both overall, $\chi^2(2, N = 301) = 20.0076$, $p = 4.523e{-}05$, and also separately for gains, $\chi^2(2,301) = 38.0499$, $p = 5.465e{-}09$, and losses, $\chi^2(2, N = 301) = 31.259$, $p = 1.63e{-}07$. All other differences were insignificant, i.e., between **VarRisk** and **VarOut** conditions overall ($p = 0.77$), for gains ($p = 0.65$), and for losses ($p = 0.64$).

The low deception rate in **VarBoth** condition suggests that

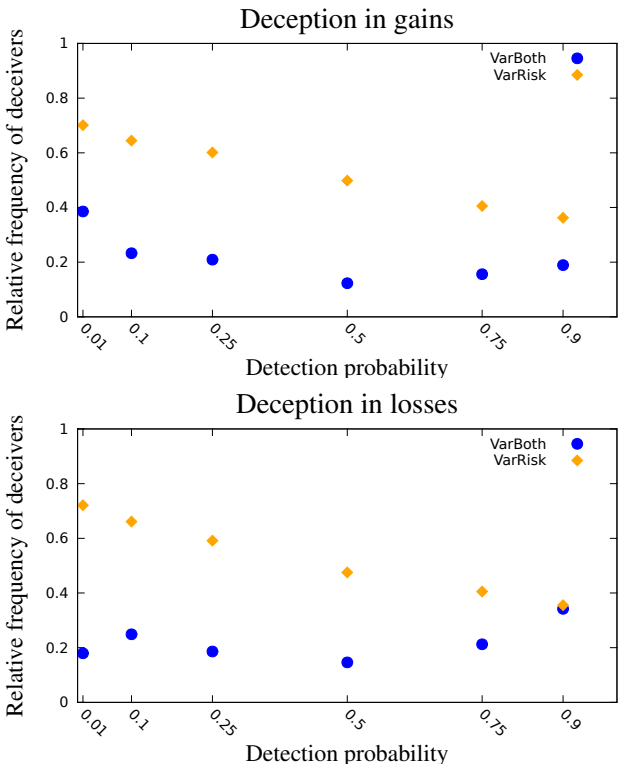


Figure 3: Relative frequencies of deceivers in the conditions varying only risk, or both risk and outcome.

Table 2: Number of deceivers and non-deceivers in each condition

Condition	Non-deceivers	Deceivers
VarOut	68	233
VarRisk	71	230
VarBoth	112	189

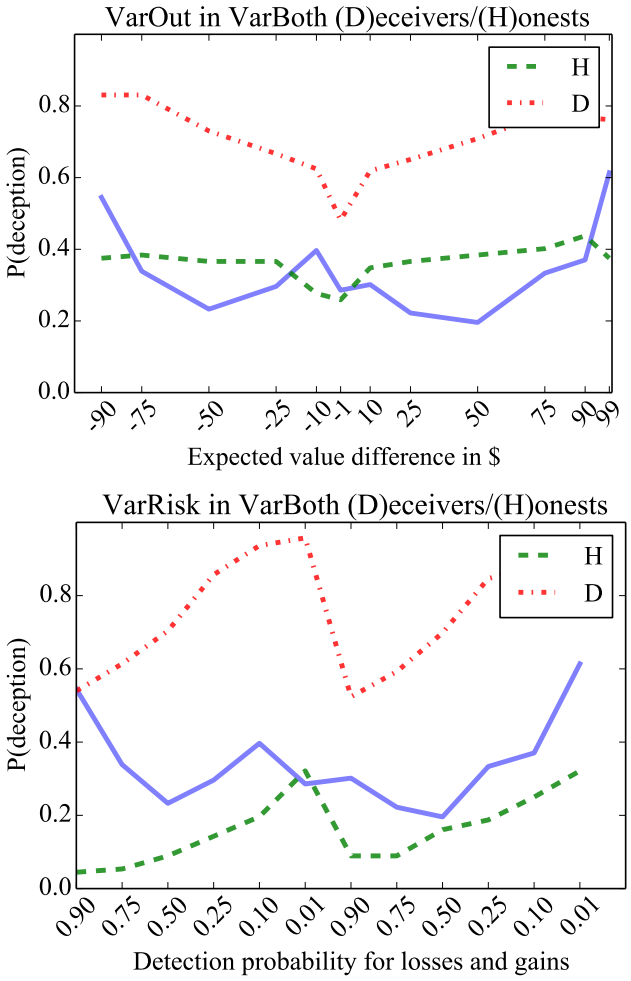


Figure 4: Probability of Deception in **VarOut** and **VarRisk** conditions for those who did and did not deceive in **VarBoth** condition.

our participants were extremely risk averse. This is partially supported by the higher deception rate observed in the riskless **VarOut** condition. On the other hand, the expected value differences between deceptive and non-deceptive options in these two conditions were equivalent, so one would expect the same (deceptive) choices in both of them, if the participants were basing their decisions on the expected values. It is less clear, though, why there is no significant difference between **VarOut** and **VarRisk** conditions, since the latter did involve risk.

Looking at the behavior in the other two conditions, we try to gain insight on what distinguishes those participants who never deceived in **VarBoth** condition (non-deceivers) from those who deceived at least once (deceivers). First, we did not find any meaningful patterns or relationships between the choices and the background variables (e.g., age or gender) in either group. Second, in both **VarOut** and **VarRisk** conditions deception rate was higher in **VarBoth** condition in both of these groups. However, there were some qualitative differences both between the groups and between the two conditions.

While removing risk (**VarOut** condition) increased the probability of deception equally in deceivers and non-deceivers, removing the penalty (**VarRisk** condition) increased it much more in deceivers (see Figure 4: the blue solid line represents the deceivers in **VarBoth** condition). In other words, incurring no potential penalty after the deception is detected seemed to be more effective incentive to deceive than having no risk of getting detected in the first place, but only for those who had a higher propensity to deceive to start with.

In both of these conditions the deceivers responded better to incentives; their probability of deceiving appears to be a function of detection probability in **VarRisk** condition and the expected value difference in **VarOut** condition, whereas such a pattern was not as apparent in non-deceivers. One

can hypothesize that this is because the non-deceivers have a higher cost of deception which overrides the effect of incentives no matter how attractive they are.

A hierarchical model

These findings motivated us to entertain a Bayesian hierarchical model (Lee, 2011) where in addition to the expected monetary gain from deception each individual i has two factors that influence her choices. The first one is her own "prize" $cdec_i$, which measures the monetary equivalent of inherent cost of lying (i.e., pure deception aversion). The other one is the cost of getting caught, $cdet_i$, which measures the shame or regret of getting caught. The model assumes that these two factors vary in the population.

The data we use in this model consists of the responses of the 301 participants in six gain questions in **VarOut** condition and six gain questions in **VarBoth** condition. We denote the choice of the participant i in the question k, $(k \in 1,2,\ldots,12)$ as D_{ik}, and the expected monetary value of the deception in question k as V_k. The probability of getting caught in question k is denoted as $pdet_k$. This probability is zero for the questions in **VarOut** condition. Finally, to turn the utilities (costs and expected gains) measured in terms of money into probabilities for choosing deception, we need a "temperature" parameter w that controls the mapping.

We can now express the model more formally,

$$
\begin{aligned}
\mu_{cdec} &\sim N(\mu_0, \sigma_0^2),\\
\sigma_{cdec} &\sim Uni(0,1000),\\
cdec_i &\sim N(\mu_{cdec}, \sigma_{cdec}^2),\\
\\
\mu_{cdet} &\sim N(\mu_0, \sigma_0^2),\\
\sigma_{cdet} &\sim Uni(0,1000),\\
cdet_i &\sim N(\mu_{cdet}, \sigma_{cdet}^2),\\
\\
w &\sim Uni(0,10),\\
p_{ik} &= logit^{-1}(w \times (V_k + cdec_i + pdet_k \times cdet_i)),\\
D_{ik} &\sim Bernoulli(p_{ik}).
\end{aligned}
$$

The hyperparameters were set to non-informative values, and the estimation was conducted using PyMC python library that implements adaptive Metropolis sampling (Patil, Huard, & Fonnesbeck, 2010).

The posterior mean values for participants' detection and deception costs appear to indicate that on average they both play an equal role in deception (see Figure 5). The pure deception aversion seems to vary between \$30 and \$150, and the detection cost between \$40 and \$110. The joint density plot reveals that the individual's detection and deception costs tend to correlate, but there are people whose detection cost is twice their deception cost (say \$100 vs. \$50).

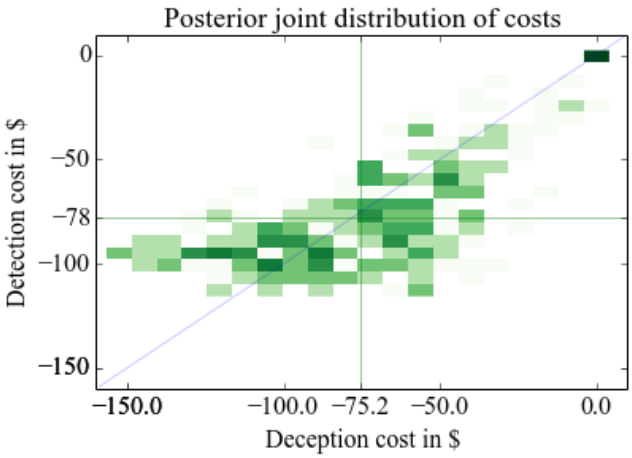


Figure 5: Joint posterior density of estimated deception and detection costs.

Discussion

It might be suggested that the MTurk workers, who are willing to spend time completing simple tasks to earn just few cents, must be unusual, and unusual in ways that can skew the effects of the experimental manipulations. They have been found to be more risk averse than other participant pools, for instance general public or student samples, but show the same pattern of risk attitudes by being risk seeking when facing losses and risk averse when facing gains (Horton, Rand, & Zeckhauser, 2011; Paolacci, Chandler, & Ipeirotis, 2010; Paolacci & Chandler, 2014; Rand, 2011).

Our participants also exhibited a seemingly high level of risk aversion. However, it was not the presence of risk per se that made the participants avoid deception, but also what may happen after their deception gets detected. If there was no difference in the outcomes when getting detected and telling the truth, the participants were willing to take risk and deceive, but if there was also a chance of incurring a penalty after getting detected, they were not. In other words, it was loss aversion rather than pure deception aversion that determined the deceptive behavior.

There are two possible interpretations of the findings: First, the participants who chose to deceive really were responding to monetary incentives, either to potentially higher gain when no risk was involved, or to the absence of loss (if getting detected) when risk was involved, rather than the presence of risk itself. However, the incentives seemed to influence more those participants who were already willing to deceive to some extent in the condition that involved both the risk and the penalty.

An alternative interpretation suggests that in those participants who never deceived both the pure lying cost and the cost of getting caught actually reflected the violation of social norms or individuals' own moral standards. In this case, the reluctance to deceive can be interpreted as maintenance of self-concept as suggested by Mazar et al. (2008).

References

Abeler, J., Becker, A., & Falk, A. (2014). Representative evidence on lying costs. *Journal of Public Economics*, *113*, 96–104.

Berg, J., Dickhaut, J., & McCabe, K. (2005). Risk preference instability across institutions: A dilemma. *Proceedings of the National Academy of Sciences*, *102*(11), 4209–4214.

Blais, A.-R., & Weber, E. U. (2006). A domain-specific risk-taking (dospert) scale for adult populations. *Judgment and Decision Making*, *1*(1), 33–47.

Charness, G., Gneezy, U., & Imas, A. (2013). Experimental methods: Eliciting risk preferences. *Journal of Economic Behavior & Organization*, *87*, 43–51.

Crosetto, P., & Filippin, A. (2013). The "bomb" risk elicitation task. *Journal of Risk and Uncertainty*, *47*, 31–65.

Deakin, J., Aitken, M., Robbins, T., & Sahakian, B. J. (2004). Risk taking during decision-making in normal volunteers: Changes with age. *Journal of the International Neuropsychological Society*, *10*(4), 590–598.

dePaulo, B. M., Kashy, D. A., Kirkendol, S. E., & Wyer, M. M. (1996). Lying in everyday life. *Journal of Personality and Social psychology*, *70*(5), 979–995.

Dreber, A., & Johannesson, M. (2008). Gender differences in deception. *Economics Letters*, *99*, 197–199.

Dror, I., Katona, M., & Mungkur, K. (1998). Age differences in decision making: to take a risk or not? *Gerontology*, *44*(2), 67-71.

Erat, S., & Gneezy, U. (2012). White lies. *Management Science*, *58*, 723–733.

Fischbacher, U., & Heusi, F. (2008). Lies in disguise: An experimental study on cheating. *Journal of the European Economic Association*, *11*(3), 525–547.

Gneezy, U. (2005). Deception: The role of consequences. *The American Economic Review*, *95*(1), 384–394.

Gneezy, U., Rockenbach, B., & Serra-Garcia, M. (2013). Measuring lying aversion. *Journal of Economic Behavior & Organization*, *93*, 293–300.

Harbaugh, W. T., Krause, K., & Vesterlund, L. (2002). Risk attitudes of children and adults: Choices over small and large probability gains and losses. *Experimental Economics*, *5*, 53–84.

Harris, C. R., Jenkins, M., & Glaser, D. (2006). Gender differences in risk assessment: Why do women take fewer risks than men? *Judgment and Decision Making*, *1*(1), 48–63.

Holt, C. A., & Laury, S. K. (2002). Risk aversion and incentive effects. *American Economic Review*, *92*(5), 1644–55.

Horton, J. J., Rand, D. G., & Zeckhauser, R. J. (2011). The online laboratory: conducting experiments in a real labor market. *Experimental Economics*, *14*, 399–425.

Huang, Y., Wood, S., Berger, D., & Hanoch, Y. (2013). Risky choice in younger versus older adults: Affective context matters. *Judgment and Decision Making*, *8*(2), 179–187.

Isaac, R. M., & James, D. (2000). Just who are you calling risk averse? *Journal of Risk and Uncertainty*, *20*(2), 177–187.

Kahneman, D., & Tversky, A. (1979). Prospect theory: An analysis of decision under risk. *Econometrica*, *47*, 263–291.

Laine, T., Sakamoto, K., & Silander, T. (2013). Do risk-averse people lie less? A comparison of risk-taking behavior in deceptive and non-deceptive scenarios. In M. Knauff, N. Sebanz, M. Pauen, & I. Wachsmuth (Eds.), *Proceedings of the 34th annual meeting of the cognitive science society* (pp. 2808–2813). Austin, TX.

Lee, M. D. (2011). How cognitive modeling can benefit from hierarchical Bayesian models. *Journal of Mathematical Psychology*, *55*(1), 1–7.

López-Pérez, R., & Spiegelman, E. (2012). Why do people tell the truth? Experimental evidence for pure lie aversion. *Experimental Economics*, *16*(3), 233–247.

Lundquist, T., Ellingsen, T., Gribbe, E., & Johannesson, M. (2009). The aversion to lying. *Journal of Economic Behavior & Organization*, *70*, 81–92.

Mather, M. (2006). A review of decision making processes: Weighing the risks and benefits of aging. In L. C. . C. Hartel (Ed.), *When I'm 64* (pp. 145–173). Washington, DC: National Academies Press.

Mazar, N., Amir, O., & Ariely, D. (2008). The dishonesty of honest people: A theory of self-concept maintenance. *Journal of Marketing Research*, *45*(6), 633–644.

Paolacci, G., & Chandler, J. (2014). Inside the Turk: Understanding Mechanical Turk as a participant pool. *Current Directions in Psychological Science*, *23*, 184–188.

Paolacci, G., Chandler, J., & Ipeirotis, P. G. (2010). Running experiments on Amazon Mechanical Turk. *Judgment and Decision Making*, *5*(5), 411–419.

Patil, A., Huard, D., & Fonnesbeck, C. J. (2010). Pymc: Bayesian stochastic modelling in python. *Journal of Statistical Software*, *35*(4), 1–81.

Rand, D. G. (2011). The promise of Mechanical Turk: How online labor markets can help theorists run behavioral experiments. *Journal of Theoretical Biology*, *299*, 172–179.

Sip, K. E., Skewes, J. C., Marchant, J. L., McGregor, W. B., Roepstorff, A., & Frith, C. D. (2012). What if I get busted? Deception, choice, and decision-making in social interaction. *Frontiers in Neuroscience*, *6*, 1–10.

Slemrod, J. (2007). Cheating ourselves: The economics of tax evasion. *Journal of Economic Perspectives*, *21*(1), 25–48.

Taylor, M. P. (2013). Bias and brains: Risk aversion and cognitive ability across real and hypothetical settings. *Journal of Risk and Uncertainty*, *46*, 299–320.

Weber, E. U. (1998). Who's afraid of a little risk? New evidence for general risk aversion. In *Decision research from Bayesian approaches to normative systems: Reflections on the contributions of Ward Edwards.* Norwell, MA: Kluwer Academic Publisher.

Should Androids Dream of Electric Sheep? Mechanisms for Sleep-dependent Memory Consolidation

George Kachergis[1], **Roy de Kleijn**[2], **and Bernhard Hommel**[2]
[1]george.kachergis@nyu.edu
[1]Psychology Department, New York University, NY, USA
[2]Institute for Psychology / LIBC, Leiden University, the Netherlands

Keywords: memory consolidation; sleep; dreaming

Humans spend almost a third of our lives asleep, with the most convincing explanation being that we otherwise suffer degradation of many cognitive and motoric skills. However, there is now also substantial empirical evidence that both declarative (i.e., facts and events–'what', 'where', 'when') and procedural (i.e., skills–'how') memory benefit from even short periods of sleep. Memory is typically described as three processes: 1) encoding: forming new traces from experience, 2) consolidation: integrating memories with prior knowledge and strengthening/crystallizing the trace, and 3) retrieval: task-dependent extraction of overall familiarity or recall of particular traces. Sleep is generally accepted to aid in consolidation, but under what circumstances it helps and by what mechanisms is not well understood.

Storage versus Processing

Machine learning algorithms can be classified as either incremental–allowing data to be added to the model instance by instance–or batch, requiring a (sometimes large) set of training instances before before the model produces useful predictions. Incremental or online algorithms (e.g., naïve Bayes) clearly offer the advantage of being able to work (however poorly) with very little data, and can learn immediately when new data are acquired. Moreover, since instances are processed immediately, they do not need to be stored for later updating. One disadvantage is that online updating may require significant computational resources, perhaps at an inconvenient time. In contrast, batch (i.e., offline; e.g., support vector machines, decision trees) learning algorithms may need a large store of data and quite some time to build an initial useful model, and adding a single training instance may require iterating over the entire (and increasing) data store to update the model. A survey of learning algorithms will reveal the classic algorithmic tradeoff: one can store more, and process less upfront (but retrieval can be costly), or process more upfront and store less.

Another problem with many incremental algorithms is the potential to arrive at different learning outcomes based on the order the instances are encountered in. In many cases, such order effects are undesirable, but humans and animals show a variety of order effects (e.g., in associative learning: Kachergis (2012). Could sleep be a chance to mitigate the order effects brought on during online learning? A few batch-update models have been found to have roughly-equivalent incremental versions. For example, latent semantic analysis (Deerwester, Dumais, Furnas, Landauer, & Harshman, 1990, LSA) learns semantic similarities of words via the singular value decomposition (SVD)–an expensive matrix operation–of a large word $\times$ document co-occurrence matrix. This large matrix–adults know over 70,000 unique words, and have read thousands of documents–must be kept in memory to be updated when a new document is read. Updating the model requires performing the SVD again, so it would be quite expensive to update knowledge every time a new document is read. It is more sensible to read a batch of documents–although, of course, this means that any new knowledge is not available in the model until the latest batch is incorporated.

Models that use batch updating require storing all of the instances in long-term memory, allowing the model to iterate over all episodes–even multiple times–to extract higher-level features (e.g., correlations of multiple features). On the other hand, incremental updating can reduce the need to store so much information, much of which may be redundant or already over-learned. We conclude that sleep might be a way to get the best of both worlds: incremental learning based on salient features for immediate use, in addition to storage of daily episodes–especially exciting or confusing memories–that can be replayed during sleep to make more thorough, careful updates to knowledge representations before further compressing the memories.

Sleep Characteristics, Effects, and Theory

Sleep in mammals and birds consists of cycles of four stages, proceeding from non-rapid eye movement (NREM) stages 1, 2, and 3 (also called slow-wave sleep), to rapid eye movement (REM) sleep. Human adults typically go through four or five cycles each night, reaching REM sleep every 90 minutes or so. More slow-wave sleep (SWS; NREM3) occurs early in the night, whereas more REM sleep occurs in the last few hours of a night's sleep. Loss of NREM3 and REM sleep results in drastically increases in these stages the following night, suggesting they are of critical importance. From neural recordings of rats, it appears that memory replay during non-REM sleep occurs at a 10x speedup, whereas REM replay is roughly at the speed of the behavioral episode (Bendor & Wilson, 2012). Waking levels of acetylcholine (ACh) during REM sleep may support encoding of new declarative memories, whereas low ACh during SWS is thought to allow replay and transfer of hippocampal memories to the neocortex (Hasselmo, 1999).

Although implicit memory effects have also been found in sleep studies, we focus on declarative memory (i.e., semantic and episodic memory; facts and knowledge). Declarative memory is thought to be largely dependent on the hippocampus enabling sleep-based consolidation of memory. During

SWS, episodic information stored in the hippocampus is replayed and projected to brain regions in the neocortex, storing stable, permanent memories. This information flow reverses during later REM sleep, conceivably allowing the hippocampus to remove the unstable, short-term memories in order to make room for new memories to be stored there (Wamsley & Stickgold, 2011). Sleep has been shown to improve recall for nonsense syllables (Jenkins & Dallenbach, 1924) and for paired-associate word stimuli (Gais & Born, 2004).

For declarative memory, there are two basic theories of how memory consolidation is improved during sleep: the active hypothesis states that consolidation depends on sleep, whereas the permissive hypothesis views consolidation as a time-dependent, interference-sensitive process that uses periods of low hippocampus input to process prior information (Mednick & Alaynick, 2010). Procedural memory is just generally thought to be 'enhanced' by sleep, but this idea is not universally accepted (Mednick & Alaynick, 2010). We will focus on proposing specific computational mechanisms for improving declarative memory, since the current models are more readily adapted to this task, and the empirical evidence indicating the necessity for this is strong.

Proposed Mechanisms

Our proposed modifications will be specified in terms of the REM (Retrieving Effectively from Memory) model from Shiffrin and Steyvers (1997), which is a multitrace memory model representing both episodic traces as well as lexical-semantic traces. Our first proposed modification is that the updating of the lexical-semantic (LS) features–which is typically not even simulated in REM–could take place during a sleep period, when episodic traces since the last sleep period are (randomly, or perhaps surprising or emotionally-charged ones) reactivated. That is, we assume that updating LS traces is tantamount to modifying the neocortical representations, which is best left for an offline period. Meanwhile, the hippocampal episodic traces may still be retrieved and used in various ways throughout the day. REM assumes that when the same stimuli appear multiple times in similar contexts, the old trace may be updated by filling in missing features from LS traces, instead of making a new trace (this differentiation process is how it accounts for the word frequency mirror effect and null list strength effect).

Retrieval in REM uses context features–reinstated by the probe, whatever its source (internal or external)–to activate a subset of long-term memory (e.g., to the studied list of items). For recognition, REM computes a likelihood ratio indicating how well a test cue (from the LS traces) match each episodic trace in the activated subset being considered. This likelihood ratio incorporates the base rate in the long-term, and the number of both the non-zero mismatching and matching features. Thus, the decision depends on not only the number of matching features, but also on how diagnostic the features are. Since small feature values will tend to be quite common and thus undiagnostic, whereas the more useful large feature values are rarely encountered, a potential mechanism for improving memory would be to redistribute feature values. By choosing at least one (unique) high-valued, diagnostic feature for each trace (or group of highly-related traces), memory will be improved. This is clearly quite computationally expensive (which is why it should be offline), but a simple, greedy version might choose one of the common stimuli from the day's traces, select one of its' LS trace's common features, and increment that feature value by one.

The SARKAE (Storing And Retrieving Knowledge And Events) model (Nelson & Shiffrin, 2013) develops REM further to explain how knowledge co-evolves along with episodic memory. Unlike REM's traces, SARKAE's traces represent not only feature values but counts of each feature value (e.g., "blue") organized by feature types (e.g., "color"). Event traces contain a single feature value (with a count of 1) if the value is copied from the stimulus, but a feature count vector may instead be copied from memory. Knowledge traces are simply those event traces that have been reactivated and updated many times, and thus contain distributions of feature values. In the SARKAE framework, a straightforward role for sleep is to act as the cleanup period: event traces from the day are considered in turn, and their feature counts are either added to an existing knowledge trace (in cortex) if a similar one is found, or copied as a new knowledge trace.

The proposed mechanisms involve many comparisons and updates to long-term lexical-semantic traces stored in neocortex, making them more suitable for conducting during sleep. Note that while the complex version of redistributing diagnostic feature values would have to be done in batch, the simple greedy version (choosing a single feature to increment) is more batch-incremental. Although we specified these mechanisms in terms of the REM model, the same mechanisms could be used in related multitrace modeling frameworks such as SARKAE or MINERVA2.

References

Bendor, D., & Wilson, M. A. (2012). Biasing the content of hippocampal replay during sleep. *Nat. Neuro.*, *15*(10), 1439–1444.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, *41*(6).

Gais, S., & Born, J. (2004). Declarative memory consolidation: Mechanisms acting during human sleep. *Learning & Memory*, *11*(6), 679–685.

Hasselmo, M. E. (1999). Neuromodulation: Acetylcholine and memory consolidation. *Trends in Cognitive Sciences*, *3*, 351–359.

Jenkins, J. G., & Dallenbach, K. M. (1924). Obliviscence during sleep and waking. *American Jour. of Psych.*, *35*(4), 605–612.

Kachergis, G. (2012). Learning nouns with domain-general associative learning mechanisms. In N. Miyake, D. Peebles, & R. P. Cooper (Eds.), *Proc. of cogsci 34* (p. 533-538). Austin, TX: Cognitive Science Society.

Mednick, S. C., & Alaynick, W. A. (2010). Comparing models of sleep-dependent memory consolidation. *J. Exp. Clin. Med.*, *2*(4), 156–164.

Nelson, A. B., & Shiffrin, R. M. (2013). The co-evolution of knowledge and event memory. *Psychological Review*, *120*(2), 356–394.

Shiffrin, R. M., & Steyvers, M. (1997). A model for recognition memory: REM–retrieving effectively from memory. *Psychonomic Bulletin and Review*, *4*(2), 145–166.

Wamsley, E. J., & Stickgold, R. (2011). Memory, sleep, and dreaming: Experiencing consolidation. *Sleep Med. Clin.*, *6*(1), 97–108.

Social Categorization Through the Lens of Connectionist Modeling

André Klapper (a.klapper@psych.ru.nl)
Radboud University Nijmegen, Behavioural Science Institute
Postbus 9104, 6500 HE Nijmegen, Netherlands

Iris van Rooij (i.vanrooij@donders.ru.nl)
Radboud University Nijmegen, Donders Institute for Brain, Cognition, and Behaviour
Postbus 9104, 6500 HE Nijmegen, Netherlands

Ron Dotsch (r.dotsch@uu.nl)
Utrecht University, Social and Organizational Psychology
P.O. Box 80.140, 3508 TC Utrecht, Netherlands

Daniël Wigboldus (d.wigboldus@psych.ru.nl)
Radboud University Nijmegen, Behavioural Science Institute
P.O. Box 9104, 6500 HE Nijmegen, Netherlands

Keywords: Categorization; Connectionism; Person Perception; Stereotyping; Cognitive Modeling; Computational Explanation

Social psychology does not yet have a strong cognitive modeling tradition. This is not for lack of cognitive modeling tools that are relevant and useful for modeling social psychological phenomena. For instance, several researchers have successfully demonstrated how connectionist modeling techniques can be used to build computational explanations of key phenomena of interest to social psychologists, such as stereotyping, prejudice and priming (Kunda & Thagard, 1996; Schröder & Thagard, 2014). In this project we contribute to this important development by addressing a major obstacle to the progression of connectionist modeling in social psychology: That is, how can we reconcile the intuitive concepts that figure in the verbal explanations that pervade social psychological theories with formal properties and processes in connectionist models? We illustrate a systematic way of addressing this question by considering the theoretical concept of 'social categories', which plays a central role in social psychological theories. Using computer simulation, we show that if social categories are defined as 'excluders' in connectionist models then key social psychological phenomena can be replicated, while maintaining a clear link with the intuitive concept of social categories. We discuss the broader implications of our simulation results for both social psychology and cognitive modeling.

Existing Person Perception Models

Social categorization theories belong to the most prominent verbal theories in the area of person perception (Fiske & Neuberg, 1990; Macrae & Bodenhausen, 2000). A general claim in these theories is that stereotyping and prejudice are the result of the natural tendency of people to categorize perceived people. A central assumption in these theories is that people construed other people based on two types of mental representations: social categories (e.g. gender, nationality, or occupation) and attributes (e.g. personality traits or physical features). It is further assumed that if a person categorizes another person then that triggers a set of (implicit) beliefs about the categorized person in the perceiver. This set of (implicit) beliefs is referred to as the *stereotype* of the category. In contrast, if attributes are assigned to the person, no such (or much fewer) beliefs are triggered. Hence, in social categorization theories, social categories are the main cause of stereotyping.

More recently, (localist) connectionist models of person perception have been proposed, which explain stereotyping and prejudice by the spread of activation between mental representations via associative links (Freeman & Ambady, 2011; Kunda & Thagard, 1996). Every mental representation in these models is associated with other mental representations, which means that every mental representation (and thus not only a particular subset) can trigger associated beliefs, in principle. Perhaps for this reason, Kunda and Thagard (1996) have presented their connectionist model as an (competing) alternative to social categorization theories. In contrast, Freeman and Ambady (2011) proposed that the general process of social categorization may be implemented by a connectionist process. These conflicting perspectives illustrate that the relationship between connectionism and social categorization has remained relatively unclear. If, and how, the different perspectives can be reconciled is thus an important open problem.

What is a Social Category?

A major obstacle to unifying social categorization and connectionist models is that the verbal term 'social category' leaves too much room for interpretation. As a first step towards an integrative model, we disentangle the most

prominent interpretations. We argue that most interpretations are either in conflict with major assumptions of social categorization theories or with empirical evidence. Based on this, we argue for an interpretation in which social categories can roughly be described as '*excluders*': that is, mental representations that strongly exclude some other mental representation. This interpretation can be implemented in a connectionist model by giving those mental representations that are conceived of as categories strong inhibitory connections that prevent their co-activation (see Fig. 1).

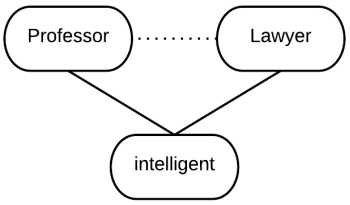


Figure 1: Illustration of a connectionist network of 'social categories' and 'attributes'. Excitatory connections are denoted by solid lines and inhibitory connections by doted lines. Within this network, *Professor* and *Lawyer* are social categories because they are connected by a (strong) inhibitory connection. In contrast, *intelligent* is an attribute because it does not have a (strong) inhibitory connection.

An Integrative Model

In our poster, we will demonstrate how social categories under our connectionist interpretation give rise to key phenomena that have been attributed to (social) categorization. Specifically, we will present simulation results that show how our connectionist interpretation of social categories gives rise to stereotyping in a way that is consistent with the general assumptions of social categorization theories.

Andersen and Klatzky (1987) provided empirical evidence that people can infer more varied characteristics about a person when provided with a category label (e.g. *professor*) compared to a trait label (e.g. *intelligent*). We replicate these results in our connectionist simulation in which activating a category (under our interpretation of categories) by external input leads to the activation of more other mental representations compared to a situation in which an attribute is activated by external input. In other words, category activation triggers more (stereotypical) beliefs than attribute activation, which does not only explain the results by Andersen and Klatzky but also conceptually replicates the category-attribute distinction in social categorization theories.

Furthermore, we show how inhibitory associations generate the general phenomenon attributed to categorization that the subjective similarities of people within categories and is decreased and the subjective similarities of people between categories is increased. This gives rise to the well-replicated phenomenon that discrimination performance is highest for stimuli that are separated by a category boundary (Goldstone & Hendrickson, 2009).

Conclusions

We replicate key social psychological phenomena that have been attributed to social categorization processes in a formal connectionist model. In addition, we provide a clear mapping of the verbal terms of social categorization theories (in particular, the terms 'categories' and 'attributes') onto formal connectionist properties. This unifies social categorization and connectionist models of person perception. Moreover, our approach demonstrates a possible way to reconcile the verbal approach taken in social categorization theories with the formal approach taken in connectionist models. That is, while the connectionist model of the social categorization process provides formal precision, the intuitive concepts 'categories' and 'attributes' provide useful verbal heuristics that summarize the functional behavior of these different mental representations in (connectionist models of) person perception. This creates a bridge between the verbal theorizing in social psychology and the formal modeling in the connectionist literature, which makes it possible for the two research areas to inform each other more in the future.

Acknowledgments

This research was supported by an NWO grant 464-11-036 awarded to Ron Dotsch.

References

Andersen, S. M., & Klatzky, R. L. (1987). Traits and social stereotypes: Levels of categorization in person perception. *Journal of Personality and Social Psychology*, *53*(2), 235–246.

Fiske, S. T., & Neuberg, S. L. (1990). A continuum of impression for- mation, from category-based to individuating processes: Influences of information and motivation on attention and interpretation. In M. Zanna (Ed.), *Advances in experimental socialpsychology*. San Diego, CA: Academic Press

Freeman, J. B., & Ambady, N. (2011). A dynamic interactive theory of person construal. *Psychological Review*, *118*, 247–79.

Goldstone, R. L., & Hendrickson, A. T. (2009). Categorical perception. *Wiley Interdisciplinary Reviews: Cognitive Science, 1*, 69-78.

Kunda, Z., & Thagard, P. (1996). Forming impressions from stereotypes, traits, and behaviors : A parallel-constraint-satisfaction theory, *Psychological Review*, *103*(2), 284–308.

Macrae, C. N., & Bodenhausen, G. V. (2000). Social cognition: thinking categorically about others. *Annual Review of Psychology*, *51*, 93–120.

Schröder, T, & Thagard, P. (2014). Priming: Constraint satisfaction and interactive competition. *Social Cognition, 32*, 152-167.

Speed-accuracy trade-off behavior: Response caution adjustment or mixing task strategies?

Leendert van Maanen (lvmaanen@gmail.com)
Department of Psychology, University of Amsterdam
Weesperplein 4, 1018 XA, Amsterdam, The Netherlands

Abstract

The speed-accuracy trade-off (SAT) effect refers to the behavioral trade-off between fast yet error-prone responses and accurate but slow responses. Multiple theories on the cognitive mechanisms behind SAT exist. One theory assumes that SAT is a consequence of strategically adjusting the amount of evidence required for overt behaviors, such as perceptual choices. Another theory hypothesizes that SAT is the consequence of mixing different task strategies. In this paper these theories are disambiguated by assessing whether the fixed-point property of mixture distributions holds, in both simulations and data. I conclude that, at least for perceptual decision making, there is no evidence for mixing different task strategies to trade off accuracy of responding for speed.

Keywords: speed-accuracy trade-off; SAT; fixed-point property; fp; mixture distributions; evidence accumulator models; diffusion model.

Introduction

In sports, acting fast is often as important as acting precise. For example, a basketball player trying to make the winning shot in the dying seconds of the game may be satisfied with less precision in his attempt given the severe time pressure of the clock. On the other hand, if he has just been awarded a free throw without any time pressure, accuracy in his attempt is vital. In experimental psychology, the ability to trade speed of responding for accuracy of responding is referred to as the speed-accuracy trade-off (SAT, Schouten & Bekker, 1967; Wickelgren, 1977). SAT-related effects have been shown in many different experimental paradigms (e.g., Dutilh et al., 2011; Meyer et al., 1988; Wagenmakers et al., 2008).

Response Caution Adjustment

The most prominent theory about the neural and cognitive mechanisms of SAT is *Response Caution Adjustment* (RCA, Bogacz et al., 2010). This view entails that SAT is a consequence of strategically adjusting the amount of evidence required for overt behaviors, such as perceptual choices. According to this view, perceptual choice behavior can be best described as the accumulation of evidence for each choice alternative. That is, given a particular stimulus, the decision maker accumulates over time which alternative is most likely to be the correct response. A response is then provided once a certain minimal level of evidence is exceeded (Figure 1A). Computational models that quantify this process have accounted for many different aspects of decision-making behavior (for reviews see Mulder et al., 2014; Ratcliff & McKoon, 2008), including SAT.

SAT occurs in the accumulator framework through response caution adjustment (Figure 1B). If a decision maker is pressed for time (or has any other reason why speed-of-responding is important), the minimal level of evidence required for a response may be set to a lower value. If a decision maker is more cautious, then the minimal evel of evidence may be set to a higher value. A high value automatically results in longer decision times – and hence longer response times (RT) – since the amount of evidence required to make a decision is larger, and thus takes longer to accrue. However, because of the stochastic nature of the evidence accumulation process, the increased decision time is accompanied by a larger probability of being correct. This is because the probability of accumulating enough evidence for the *incorrect* response alternative is lower as the threshold is set higher.

Mixing Task Strategies

The RCA theory of SAT has been tested in many different studies (e.g., Rae et al., 2014; Mulder et al., 2010, 2013), and in addition is also consistent with many neuroscientific findings (Boehm et al., 2014; Forstmann et al., 2008, 2010; Ho et al., 2012; Van Maanen et al., 2011; Winkel et al., 2012).[1] Nevertheless, alternative theories have been proposed about the nature of SAT. However, no model comparison between different theoretical proposals for SAT has so far been attempted.

One alternative theory of SAT that warrants a formal comparison with RCA is what I refer to here as the *Mixing Task Strategies* (MTS) theory. This theory entails that participants switch between two modes of responding during a task, depending on the speed and accuracy requirements (Ollman, 1966; Meyer et al., 1988). Under accuracy stress, participants respond through a stimulus-controlled process, which is thought to yield optimal – yet relatively slow – performance. Under speed stress, participants are thought to recruit an additional guess process on a large proportion of trials. Because this is hypothesized to be a fast process, the average response times decreases. However, because the guess process leads to chance performance on a certain proportion of trials, accuracy drops as well. This mixture idea lies at the heart of more modern models of SAT, such as the phase-transition model by Dutilh et al. (2011) and a recent ACT-R model of SAT (Schneider & Anderson, 2012).

The essential property of the Mixing Task Strategies theory is that participants use two modes of responding, but in different proportions. In fact, a strong prediction is that any exper-

[1] For completeness, it should be mentioned that many of these formal modeling approaches also required the "non-decision time" parameter to vary between speed-stressed and accuracy-stressed conditions.

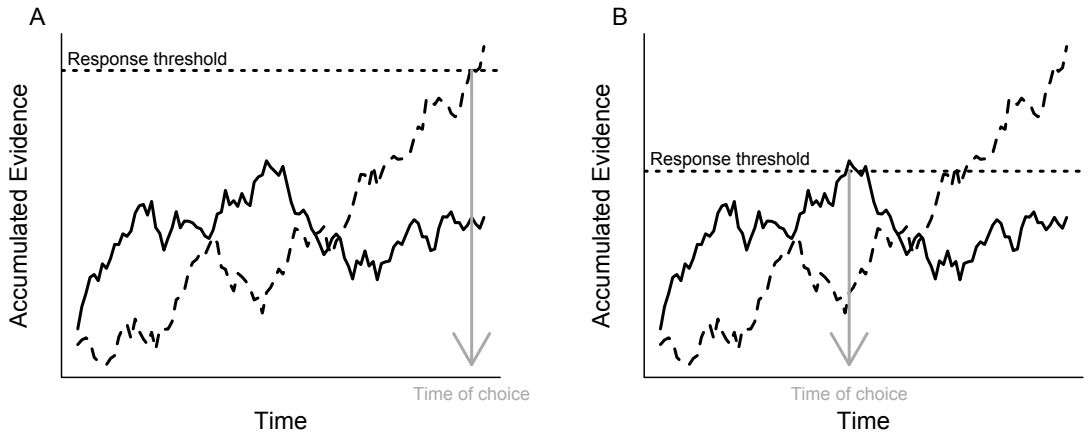


Figure 1: A. An illustration of two evidence accumulation processes, one depicted by a solid line, one by a dashed line. The process that reaches the response threshold the earliest is selected. B. A decreased threshold (panel B vs A) may yield a faster, possibly incorrect, choice.

imental condition that has intermediate speed and accurate stress, should have an in intermediate mixing proportion of the two modes as well. In this paper will test this strong prediction for a simple perceptual choice task (Forstmann et al., 2008) using the fixed-point property of mixture distributions (Falmagne, 1968).

Fixed-Point Property

The fixed-point property (Falmagne, 1968) is a general property of mixture distributions with two base distributions, that can be easily applied to response time data (Van Maanen et al., 2014). Because the probability density of a binary mixture distribution is always the weighted sum of the densities of the two base distributions, it follows that there is (at least) one value that has the same density, independent of the mixture proportions (for a proof, see Falmagne 1968; reiterated in Van Maanen et al. 2014). In terms of mixture distributions of response times, this implies that there will be one RT for which the probability of providing a response at that particular time is equal for all mixtures.

The fixed-point property is illustrated in Figure 2. The figure shows the probabilty densities of four binary mixture distributions. Each is a mixture of two shifted Wald distribution functions with common scale ($\lambda = 5000$) and shift ($\theta = 100$), but different means ($\mu_1 = 300$ and $\mu_2 = 500$).[2] The legend in Figure 2 refers to the mixture proportion, here represented as the proportion of the data that comes from the second base distribution (with $\mu_2 = 500$). As is clear from the figure, all densities cross each other at a common RT value, referred to as the *crossing point*. In the Results section below, we will test for the presence of the fixed-point property in empirical data by assessing whether across participants, the crossing points of pairs of distributions with different mixture proportions are indeed the same.

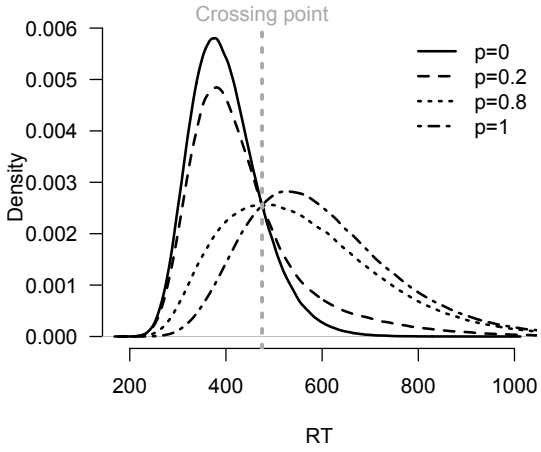


Figure 2: Binary mixture distributions with different mixture proportions always cross at a common RT.

The fixed-point property is predicted by the MTS theory of SAT. That is, if observed RT distributions in SAT are a mixture of the guess process and the stimulus-controlled process, and the mixture proportions differ as a result of the amount of speed stress, then the fixed-point property should be present in the data. On the other hand, the fixed-point property is not predicted by the RCA theory. These predictions will be fleshed out in the next section.

Simulations

To understand which of the theories of SAT predicts the fixed-point property in RT distributions, I generated data under the two theories, for three levels of speed stress. In the RCA

[2] I chose the shifted Wald distribution function as an example because of its wide applicability in RT data (e.g., Anders et al., 2015; Heathcote, 2004), but the fixed-point property does not depend on the choice of distribution function.

simulation, all trials are drawn from a simple randon-walk process with positive drift (cf. Bogacz et al., 2006), and the speed-stress levels are simulated by three different settings of an absorbing boundary. In the MTS simulation, only a proportion of the trials is drawn from that random-walk process, with the remaining trials drawn from a guess process. The three levels of speed-stress are simulated by different mixture proportions.

Response Caution Adjustment Simulation

For the RCA simulation I used a pure drift diffusion model (Bogacz et al., 2006):

$$dx = \mu dt + N(0, \sigma^2 dt) \text{ with } x(0) = a/2. \quad (1)$$

The speed of evidence accumulation is represented by the constant drift μdt, with standard deviation σ. On each trial, a decision is made once the evidence x exceeds one of two boundaries at $x = 0$ and $x = a$. The response time is then determined by the time when one of the boundaries is crossed, plus a fixed non-decision time intercept t_0. Similar models have been applied to many decision making paradigms to study the cognitive (e.g., Donkin & Van Maanen, 2014; Mulder et al., 2013; Palmer et al., 2005; Ratcliff, 1978; Van Maanen et al., 2012b,a) and neural (e.g., Forstmann et al., 2008, 2010; Ratcliff et al., 2009) mechanisms underlying choice behavior. In particular, this model has been used extensively to study SAT. Overall, SAT has been linked to changes in the boundary parameter a (e.g., Forstmann et al., 2008, 2010; Mulder et al., 2013; Van Maanen et al., 2011; Winkel et al., 2012).

To generate RT distributions for this model, I simulated 10,000 trials in each condition, with the following parameters: $\mu = 0.2; \sigma = 0.3; t_0 = 200; a_1 = 0.3; a_2 = 0.6; a_3 = 0.72$. Table 1 presents mean RTs for correct responses and accuray of these simulations, to illustrate that indeed a SAT is simulated.

Table 1: Summary of simulated data.

Model	Mean RT (ms)	Accuracy
RCA		
- $a_1 = 0.3$	458	.67
- $a_2 = 0.6$	1103	.80
- $a_3 = 0.72$	1416	.84
MTS		
- $p_1 = 0.6$	885	.68
- $p_2 = 0.75$	987	.72
- $p_3 = 1.0$	1104	.80

Figure 3A displays kernel density estimates of the RT distributions for correct responses under the RCA theory. The standard deviation of the smoothing kernel is set at 1,000 ms, above the minimal value of 1 standard deviation in the data, as suggested by Van Maanen et al. (2014). It is clear that these density functions do not all cross at the same RT. Figure 3B shows this even clearer. Here, the differences between each pair of speed-stress levels (i.e., boundary settings) are shown. The RTs where these differences are zero are the crossing points. The absence of the fixed-point property in this simulation is apparent from the multiple crossing points.

Mixing Task Strategies Simulation

The MTS simulation generates data from a stimulus-controlled and a guess process. The stimulus-controlled process is identical to the RCA simulation, except that the boundary setting of the pure drift diffusion is always set at $a = 0.6$. The guess process is simulated by a random draw from a Bernoulli process representing the choice, and an independent draw from a normal distribution with mean $\mu_{guess} = 400$ and $\sigma_{guess} = 100$ representing the response time. Of note is that the mean RT of the guess process is below the mean RT of the stimulus-controlled process, as it represents the faster speed-stressed trials (see the mean RT for the RCA simulation with $a_2 = 0.6$ in Table 1).

Table 1 again presents mean RT for correct responses as well as accuracy for the simulations under the MTS theory. This shows that MTS is indeed consistent with a general SAT effect. Figure 3C shows the kernel density estimates of the RT distributions (with the same smoothing kernel as for the RCA simulations); Figure 3D the density differences. These figures confirm that the MTS theory predicts a fixed-point in the data, as all crossing points in Figure 3D align.

Analysis of Behavioral Data

Simulation of an RCA and an MTS model suggest that a fixed-point in the data is consistent with the MTS theory, but not with the RCA theory. To disentangle these alternative accounts in the domain of perceptual decision making, I reanalyzed data from Forstmann et al. (2008). In this study, participants were asked to perform a random-dot motion task while being stressed for either speed, accuracy, or both on a trial-by-trial basis. This task has been used extensively in the context of SAT (e.g., Palmer et al., 2005; Forstmann et al., 2008; Van Maanen et al., 2011; Mulder et al., 2013) and SAT effects have been explained by the RCA theory. However, a formal comparison with the MTS theory has never been performed. In this particular experiment, the presence of three levels of speed-stress enables a test of the MTS hypothesis that the fixed-point property holds in the data. If the MTS theory is correct, then the proportion of guess responses should be lower for accuracy-stressed trials than for speed-stressed trials. Stressing both speed and accuracy (or rather not stressing anything) should yield a proportion of guess responses that is in between these two extremes.

The Task

In the random-dot motion task, participants had to indicate from a cloud of semi-randomly moving dots what the overall direction of motion is. Prior to each stimulus, participants were presented with one of three cues for 1,000 ms. The cues

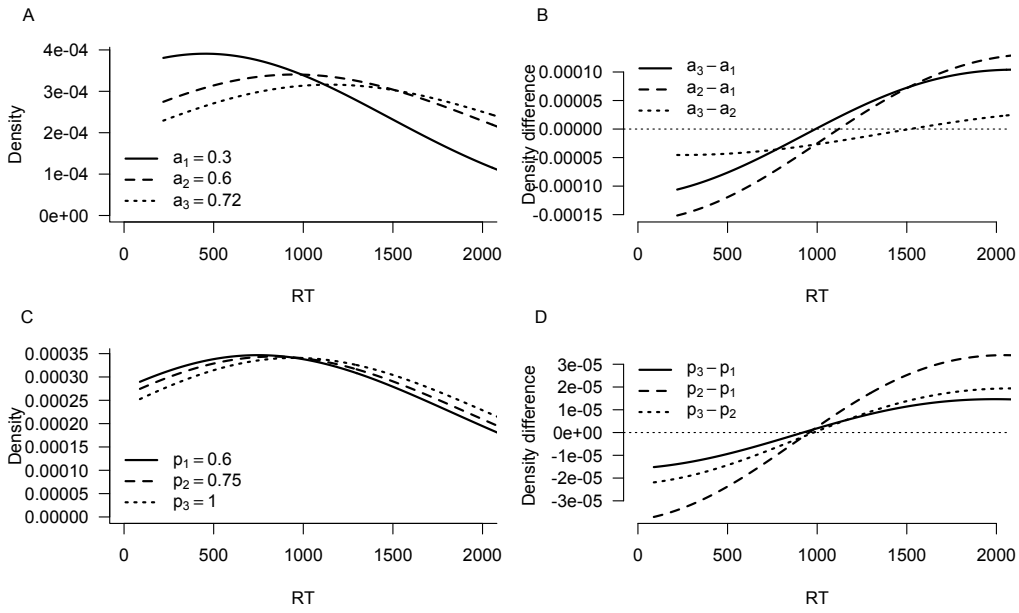


Figure 3: A. Kernel density estimates of three simulated conditions under the Response Caution Adjustment theory; lines represent different threshold settings (*a*). B. Density differences of each pair of conditions from A. C. Densities of three simulated conditions under the Mixing Task Strategies theory; lines represent the different proportions *p* of trials that are form the stimulus-controlled process. D. Density differences of each pair of conditions from C.

could be either "SN" (referring to the German "Schnell"), "NE" ("Neutral", stressing neither speed nor accuracy), or "AK" ("Akurat"). After a variable interval of 500 ms, the stimulus appeared for another 1,000 ms, followed by 350 ms feedback. Feedback reflected the previously presented cue. Thus, when the cue was either "SN" or ''NE", feedback was given on response speed; when the cue was either "AK" or "NE", feedback was given on response accuracy. The experiment consisted of 840 trials, equally distributed across the conditions. A total of 20 participants took part in the experiment (see Forstmann et al. 2008 for more details on the experimental procedure).

Results

To assess the presence of the fixed-point property, I only analyzed correct responses (additional simulations showed that the influence of incorrect responses on the crossing points was marginal). The kernel density estimates were computed using a kernel with a standard deviation of 300 ms. Figure 4A and B illustrate that there is no fixed-point in the data. For these figures I aggregated all data points to compute one density function per condition. However, to formally assess the presence of the fixed-point property would be to test within-subjects whether the crossing points are the same (Van Maanen et al., 2014). Because standard frequentist analyses can only test for the presence of a *difference* between conditions, we prefer to apply Bayesian statistics (Rouder et al., 2012). A Bayesian ANOVA (Rouder et al., 2012) quantifies the probability that the observed crossing points are sampled from one underlying population (i.e., when the fixed-point property holds) or are sampled from multiple populations (when the fixed-point property does not hold).

Crossing points of the density differences per condition and participant were computed and are presented in Figure 4C. A Bayesian within-subjects ANOVA yields a Bayes factor of 53 in favor of multiple populations of crossing points. This means that the data are 53 times more likely to be generated by such a model than by a model assuming one true population. This result is clearly not in agreement with the fixed-point property, and by extension not in agreement with the MTS theory.

Discussion & Conclusion

The data from Forstmann et al. (2008) is not consistent with an important signature of binary mixture distributions. The absence of the fixed-point property therefore speaks against a MTS theory of SAT. A Bayesian analysis shows that it is in fact 53 times more likely that the data are not from binary mixture distributions. This result is consistent with an RCA theory of SAT. To some extent, this is not surprising, given the excellent fits of cognitive models that implement the RCA theory, both on this data set as well as on related data (e.g., Forstmann et al., 2010; Van Maanen et al., 2011; Mulder et al., 2010, 2013). However, no formal model comparison had so far been attempted. Theoretically, the MTS theory could have generated data that would be excellently fit by RCA models (cf. model mimickry, Ratcliff, 1988; Ratcliff & Smith, 2004). The phase-transition model of Dutilh et al. (2011, an instance of MTS), has been compared to other mod-

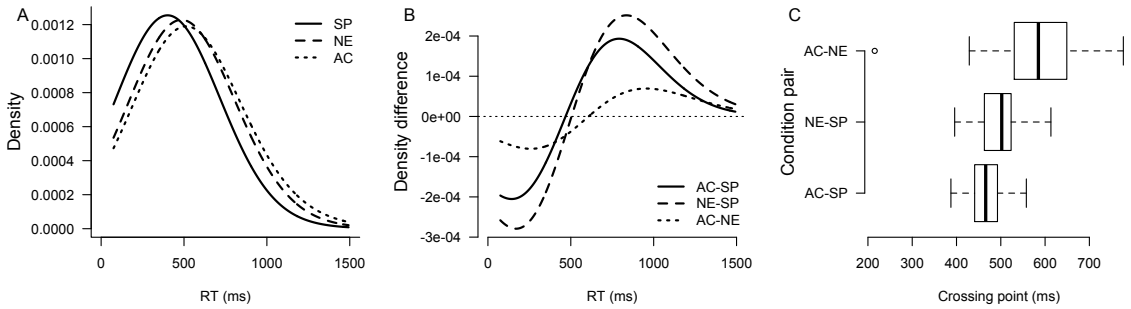


Figure 4: A: Densities of the correct RT distributions in the data. B. Density differences of each conditions pair from A. C. Boxplots indicating the distribution of crossing points per condition pair.

els, but the authors did not include an RCA model in their model comparison. Therefore, although they argue against RCA, it cannot be excluded based on their study.

It is entirely possible that the effects that are collectively referred to as SAT effect depend on different cognitive mechanisms. For example, if presenting a speed-stress cue results to increased preparation (e.g., motor preparation, Rhodes et al. 2004) independently of which mode is actually used on that specific trial, then a fixed-point would also not observed. This is because the observed response time distributions are not pure mixtures of two base distributions, but rather constitute multiple processes.

Additionally, an experimental paradigm that promotes true guessing behavior may indeed still best be explained by MTS, while an experiment where guessing never leads to satisfactory behavior may be best explained by RCA. Under this view, the best explanation of SAT may be a mixture of RCA and MTS. Nevertheless, the current model and analyses strongly suggests an important role for adjusting control when people are confronted with situations in which the importance of response speed varies.

To disentangle the MTS and RCA theories, I took advantage of the different predictions that these two models make with respect to mixtures of behaviors. The fixed-point property provides an excellent tool to test these predictions.[3] Similar predictions may be found in other domains where multiple strategies for a task may (or may not) be expected. Examples include multiple reasoning strategies that may be involved in reasoning tasks (Meijering et al., 2010) or varying proportions of fast-and-automatic processing and slow and deliberate processing, such as can be found in motor sequence learning (Rhodes et al., 2004) or developmental transitions (Van Rijn et al., 2003). For these kinds of response time data, the presence or absence of the fixed-point property seems to be an easy test of multiple competing task processes.

[3]Van Maanen et al. (2014) includes R code for testing the fixed-point property.

Acknowledgments

Thank you to Birte Forstmann for providing the data and to Hedderik van Rijn voor providing comments on an earlier version of this paper.

References

Anders, R., Alario, F. X., & Van Maanen, L. (2015). An instrumental cognitive model for speeded and/or simple response tasks. In N. A. Taatgen, M. Van Vugt, J. P. Borst, & K. Mehlhorn (Eds.), *Proceedings of the 13th International Conference on Cognitive Modeling*.

Boehm, U., Van Maanen, L., Forstmann, B. U., & Van Rijn, H. (2014). Trial-by-trial fluctuations in CNV amplitude reflect anticipatory adjustment of response caution. *Neuroimage*, *96*, 95–105.

Bogacz, R., Brown, E., Moehlis, J., Holmes, P., & Cohen, J. D. (2006). The physics of optimal decision making: a formal analysis of models of performance in two-alternative forced-choice tasks. *Psychological Review*, *113*, 700–765.

Bogacz, R., Wagenmakers, E. J., Forstmann, B. U., & Nieuwenhuis, S. (2010). The neural basis of the speed-accuracy tradeoff. *Trends in Neurosciences*, *33*, 10–16.

Donkin, C., & Van Maanen, L. (2014). Piéron's law is not just an artifact of the response mechanism. *Journal of Mathematical Psychology*, *62-63*, 22–32.

Dutilh, G., Wagenmakers, E.-J., Visser, I., & van der Maas, H. L. J. (2011). A phase transition model for the speed-accuracy trade-off in response time experiments. *Cognitive Science*, *35*, 211–250.

Falmagne, J. (1968). Note on a simple fixed-point property of binary mixtures. *British Journal of Mathematical and Statistical Psychology*, *21*, 131–132.

Forstmann, B. U., Anwander, A., Schäfer, A., Neumann, J., Brown, S. D., Wagenmakers, E.-J., Bogacz, R., & Turner, R. (2010). Cortico-striatal connections predict control over speed and accuracy in perceptual decision making. *Proceedings of the National Academy of Sciences of the United States of America*, *107*, 15916–15920.

Forstmann, B. U., Dutilh, G., Brown, S., Neumann, J., von Cramon, D. Y., Ridderinkhof, K. R., & Wagenmakers, E.-J. (2008). Striatum and pre-SMA facilitate decision-making under time pressure. *Proceedings of the National Academy of Sciences of the United States of America*, *105*, 17538–17542.

Heathcote, A. (2004). Fitting wald and ex-wald distributions to response time data: an example using functions for the s-plus package. *Behavior Research Methods Instruments & Computers*, *36*, 678–694.

Ho, T. C., Brown, S., van Maanen, L., Forstmann, B. U., Wagenmakers, E.-J., & Serences, J. T. (2012). The optimality of sensory processing during the speed-accuracy tradeoff. *Journal of Neuroscience*, *32*, 7992–8003.

Meijering, B., Van Maanen, L., Van Rijn, H., & Verbrugge, R. (2010). The facilitative effect of context on second-order social reasoning. In C. R., & S. Ohlsson (Eds.), *Proceedings of the 32nd Annual Meeting of the Cognitive Science Society*.

Meyer, D. E., Irwin, D. E., Osman, A. M., & Kounios, J. (1988). The dynamics of cognition and action: mental processes inferred from speed-accuracy decomposition. *Psychological Review*, *95*, 183–237.

Mulder, M., Bos, D., Weusten, J., van Belle, J., van Dijk, S., Simen, P., van Engeland, H., & Durston, S. (2010). Basic impairments in regulating the speed-accuracy tradeoff predict symptoms of ADHD. *Biological Psychiatry*, *68*, 1114–1119.

Mulder, M., Keuken, M., Van Maanen, L., Boekel, W., Forstmann, B. U., & Wagenmakers, E.-J. (2013). The speed and accuracy of perceptual decisions in a random-tone pitch task. *Attention, Perception & Psychophysics*, *75*, 1048–1058.

Mulder, M., Van Maanen, L., & Forstmann, B. U. (2014). Perceptual decision neurosciences – a model-based review. *Neuroscience*, (pp. 872–884).

Ollman, R. (1966). Fast guesses in choice reaction time. *Psychonomic Science*, *6*, 155–156.

Palmer, J., Huk, A. C., & Shadlen, M. N. (2005). The effect of stimulus strength on the speed and accuracy of a perceptual decision. *Journal of Vision*, *5*, 376–404.

Rae, B., Heathcote, A., Donkin, C., Averell, L., & Brown, S. (2014). The hare and the tortoise: Emphasizing speed can change the evidence used to make decisions. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *40*, 1226–1243.

Ratcliff, R. (1978). A theory of memory retrieval. *Psychological Review*, *85*, 59–108.

Ratcliff, R. (1988). A note on mimicking additive reaction time models. *Journal of Mathematical Psychology*, *32*, 192–204.

Ratcliff, R., & McKoon, G. (2008). The diffusion decision model: Theory and data for two-choice decision tasks. *Neural Computation*, *20*, 873–922.

Ratcliff, R., Philiastides, M. G., & Sajda, P. (2009). Quality of evidence for perceptual decision making is indexed by trial-to-trial variability of the eeg. *Proceedings of the National Academy of Sciences of the United States of America*, *106*, 6539–6544.

Ratcliff, R., & Smith, P. L. (2004). A comparison of sequential sampling models for two-choice reaction time. *Psychological Review*, *111*, 333–367.

Rhodes, B. J., Bullock, D., Verwey, W. B., Averbeck, B. B., & Page, M. P. A. (2004). Learning and production of movement sequences: behavioral, neurophysiological, and modeling perspectives. *Human Movement Science*, *23*, 699–746.

Rouder, J. N., Morey, R. D., Speckman, P. L., & Province, J. M. (2012). Default Bayes factors for ANOVA designs. *Journal of Mathematical Psychology*, *56*, 356–374.

Schneider, D. W., & Anderson, J. R. (2012). Modeling fan effects on the time course of associative recognition. *Cognitive Psychology*, *64*, 127–160.

Schouten, J. F., & Bekker, J. A. (1967). Reaction time and accuracy. *Acta Psychologica*, *27*, 143–153.

Van Maanen, L., Brown, S. D., Eichele, T., Wagenmakers, E. J., Ho, T. C., Serences, J. T., & Forstmann, B. U. (2011). Neural correlates of trial-to-trial fluctuations in response caution. *Journal of Neuroscience*, *31*, 17488–17495.

Van Maanen, L., De Jong, R., & Van Rijn, H. (2014). How to assess the existence of competing strategies in cognitive tasks: a primer on the fixed-point property. *PLoS One*, *9*, e106113.

Van Maanen, L., Grasman, R. P. P. P., Forstmann, B. U., Keuken, M., Brown, S. D., & Wagenmakers, E. J. (2012a). Similarity and number of alternatives in the random-dot motion paradigm. *Attention, Perception & Psychophysics*, *74*, 739–753.

Van Maanen, L., Grasman, R. P. P. P., Forstmann, B. U., & Wagenmakers, E. J. (2012b). Piéron's law and optimal behavior in perceptual decision-making. *Frontiers in Decision Neuroscience*, *5*, article 143.

Van Rijn, H., Van Someren, M., & Van der Maas, H. L. J. (2003). Modeling developmental transitions on the balance scale task. *Cognitive Science*, *27*, 227–257.

Wagenmakers, E.-J., Ratcliff, R., Gomez, P., & McKoon, G. (2008). A diffusion model account of criterion shifts in the lexical decision task. *Journal of Memory and Language*, *58*, 140–159.

Wickelgren, W. (1977). Speed-accuracy tradeoff and information-processing dynamics. *Acta Psychologica*, *41*, 67–85.

Winkel, J., Van Maanen, L., Ratcliff, R., Van der Schaaf, M., Van Schouwenburg, M., Cools, R., & Forstmann, B. U. (2012). Bromocriptine does not alter speed-accuracy tradeoff. *Frontiers in Decision Neuroscience*, *6*, Article 126.

An Instrumental Cognitive Model for Speeded and/or Simple Response Tasks

Royce Anders (royce.anders@univ-amu.fr)
LPC UMR 7290 CNRS, Aix-Marseille Université
3 place Victor Hugo, 13331, Marseille Cedex 3, France

F.-Xavier Alario (francois-xavier.alario@univ-amu.fr)
LPC UMR 7290 CNRS, Aix-Marseille Université
3 place Victor Hugo, 13331, Marseille Cedex 3, France

Leendert van Maanen (l.vanmaanen@uva.nl)
Department of Psychology, University of Amsterdam
Weesperplein 4, 1018 XA, Amsterdam, The Netherlands

Abstract

Speeded and/or simple response tasks may be cognitively modeled by a random walk process that accumulates to threshold. In cases of tasks where mainly one characteristic response is observed, at varying latencies, then random walks involving only positive drifts that each arrive at a single threshold, provide a suitable accumulation modeling account of the data; and advantageously, this accumulation model is exactly described by the shifted Wald (SW) probability density function. We will demonstrate how the SW distribution is thus a noteworthy cognitive model for these tasks, which uniquely possesses simultaneously, high utility as an objective data measurement tool for the response time (RT) distributions. Per each experiment condition, its three parameters can decompose the observed mean RT value, quantify the shape and characteristics of the observed RT distribution, and account for significant differences between distributions with near-identical mean values; regardless of whether one accepts the cognitive interpretation of the random-walk accumulation process. We present the SW model and demonstrate its efficiency and utility on both simulated and real data.

Keywords: response time analysis, shifted Wald, psychometrics, accumulation modeling

Introduction

In the psychological sciences, the efficacy of modeling the distributions of response time (RT) data, rather than only using classical methods, to obtain a deeper understanding of experiment effects and underlying processes, has been well-demonstrated in the preceding literature (Ratcliff, 1978; Luce, 1986; Ratcliff & Rouder, 1998; Andrews & Heathcote, 2001; Heathcote, 2004; Van Zandt, 2000, 2002; Ratcliff et al., 2004; Balota et al., 2008; Van Maanen et al., 2012; Staub et al., 2010; Balota & Yap, 2011). In the present paper we bring attention to a simple-yet-powerful tool for RT data analysis, that despite its utility, is not yet in general use within the psychological community.

There exist quantitative distribution measurement tools for RT data, in which the parameters describe the properties of the observed data distribution; these tools are typically closed-form probability density functions with positive skew and values, such as the shifted Wald (SW, see Chapter 8.2 Luce, 1986; Heathcote, 2004), ex-Gaussian (Heathcote et al., 1991), shifted Weibull, shifted log-normal, and Gumbel (Wagenmakers & Brown, 2007). Then there are more complicated models of RT data that model signal accumulation: such as the Linear Ballistic Accumulator (LBA, Brown & Heathcote, 2008), race model (LaBerge, 1962), and the Drift Diffusion Model (DDM, Ratcliff & Murdock, 1976; Ratcliff, 1978; Ratcliff & McKoon, 2008), however their parameters do not directly describe the distribution of RT data. We bring to attention that uniquely, the SW distribution does both at the same time, and argue that as an accumulation model, it is on par in usefulness with more complex models of accumulation, when used in the appropriate context.

The Shifted Wald

Among a number of situations, the SW for RT data is apt for experiments consisting of speeded (e.g. 500ms-2000ms) and/or simple response tasks, where in particular, the ratio of errors to correct responses is small; some concrete examples consist of visual search, picture-naming, simple detection, and go no-go tasks. One should note that being only a distribution that is fit, the SW is a very simplistic model with few assumptions. However despite its simplicity, it possesses a formidable characteristic that stands it apart from the other distributions and models listed: while its parameters directly quantify the RT distribution, they also simultaneously, directly describe the RT values in the context of a Brownian motion process (BMP) in which a latent quantity accumulates to threshold; this is the same kind of BMP, related throughout the literature to signify the signal-to-response threshold event, that is at the root of the other popular signal-accumulation models, such as the DDM, race, and LBA models. Thus while being a quantitative measurement tool that can be applied to describe the distribution of any set of magnitude RTs, the SW model also provides an opportunity for theoretical work, such as on the cognitive-behavioral response process, based on its ability to also describe the data in terms of latent signal accumulation.

As an Accumulation Model

The SW with parameters, γ, α, and θ, can directly describe the data in the context of a continuous time-stochastic process (a type of BMP), consisting of a single latent quantity, X, that is continuously accumulating until it reaches a threshold. More specifically, X, accumulates at a given rate, γ, with

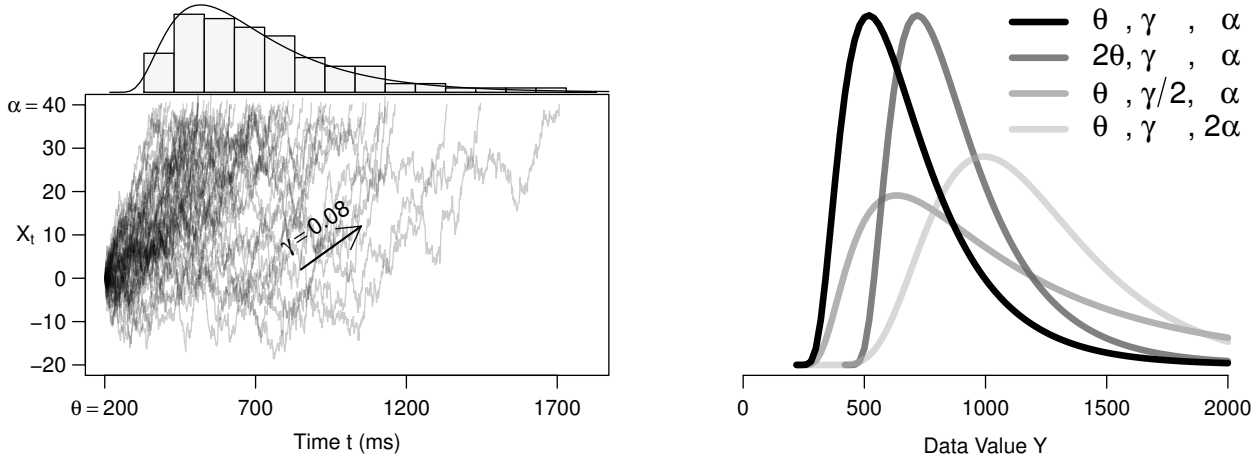


Figure 1: The SW as a cognitive-behavioral model (left), describing the RT data in the context of a latent quantity (e.g. signal) accumulating to threshold, α, at rate, γ, where θ accounts for the time lapsed outside of (around) this process. Then the SW as a distribution measurement tool (right). The black distribution has $\theta = 200$, $\gamma = 0.08$, and $\alpha = 40$, and illustrated in different shades of grey, are individual parameter adjustments that each cause unique distribution outcomes. Here they are each adjusted in the direction that results in bigger data values (e.g. slower RTs, slower mean RTs), which in each instance, results by a different distribution form.

noise until it reaches a threshold, α; and θ (the shift) is the minimal time lapsed outside of the process, which can be distributed before and after this accumulation process; the total time lapsed, T, is the data fit by the SW. This latent accumulation process provides a potential model for any data that involves a quantity accumulating over time that eventually reaches a value (or threshold). The SW thus provides the opportunity for a potentially-useful model, analogous to the signal-to-response threshold event of behavior.

In the context of RT data and the appropriate experimental task, this kind of underlying accumulation process that we note is similarly shared (by elementary adjustments) with the other aforementioned accumulation models, has been well-supported to correspond to a signal-to-response threshold, cognitive-behavioral event. In the case of the signal-to-response threshold interpretation of the SW: γ corresponds to the accumulation rate of the internal signal X, α to the threshold needed to initiate the physical response, and θ to the time distributed before and after this process (thus time lapsed outside of signal accumulation). The total time lapsed, T, is the RT recorded.

This latent accumulation process is illustrated in the left plot of Figure 1, in which many *random walks with drift* (RWDs, starting at $\theta = 200$, and having average slope $\gamma = 0.08$) as they intercept threshold $\alpha = 40$, are shown to correspond to a SW distribution with the same parameters: $\{\gamma = 0.08\,,\, \alpha = 40\,,\, \theta = 200\}$. Each of these RWDs are of the form

$$X_t = X_{t-1} + \gamma + \varepsilon\,, \tag{1}$$

where the position of a random variable X at time t, as X_t, is equal to its prior position value, X_{t-1}, plus a movement tendency, $\gamma > 0$ (known as drift), and marginal error, ε (or noise).[1]

Then note that any given threshold, $\alpha > 0$, unto which the time process terminates when X_t reaches that value, as $X_t \geq \alpha$, will produce a Wald distribution of data: letting T denote the time t at which X_t reaches α, then the data is of the form

$$\mathbf{T} = (T_i)_{1\times N}\,, \tag{2}$$

for the N times (e.g. or RT observations) that the SW distribution describes (T is also known as the *first passage time* of the BMP). Parameter θ functionally accounts for these aspects external to the RWD by shifting all values of t by a constant, in which the starting point of the accumulation process, $X_0 = 0$, instead becomes, $X_\theta = 0$. While θ shifts the distribution from the left, note that its effect, mathematically, is equivalent in being able to account for external processes that occur on either side of the accumulation event.

As a Distribution Measurement Tool

While the SW and its parameters can directly describe the data in the context of a latent quantity accumulating to threshold, the SW can also serve as an objective distribution measurement tool, in which its parameters, γ, α, and θ, will directly quantify the density of the observed RT distribution.

[1]The RWD form in (1) is the same kind used by other models of accumulation: the LBA, race, and DDM, with elementary adjustments; these are specified in the Discussion.

The SW distribution with probability density function

$$f(X \mid \gamma,\alpha,\theta) = \frac{\alpha}{\sqrt{2\pi(X-\theta)^3}} \cdot \exp\left\{-\frac{[\alpha-\gamma(X-\theta)]^2}{2(X-\theta)}\right\}, \quad (3)$$

has expected value $\alpha/\gamma+\theta$, and variance α/γ^3, for $X > \theta$. The pdf is illustrated in the right plot of Figure 1, in which the distribution in black print has parameters $\theta = 200$, $\gamma = 0.08$, and $\alpha = 40$; then in different shades of grey, the figure also illustrates the outcome obtained when each of these parameters are individually adjusted in the direction that results in bigger data values (e.g. slower RTs). In each parameter adjustment, there is a unique distribution outcome: for example one can see parameter θ will give the position of the leading edge of the distribution, and shifts the entire distribution horizontally (to the right for slower RTs); then γ and α both serve to locate the central tendency within the shifted distribution; but γ is more informative for mass in the tail, and hence steepness of the leading curve (thicker tail for slower RTs); and α for the deviation centrally around the mode value, and hence normality around the mode (larger deviation for slower RTs).

While these individual parameter adjustments illustrated in the right plot of Figure 1, each provide for a unique distribution outcome, note that some of these distributions however share similar mean RTs, such as the dark and medium-grey distributions. Such can be the case in real data, when markedly different distributions, with near-equal means, are observed across experimental manipulations. Experiment manipulations with such contrasting distribution results, yet similar mean RTs, could likely cause a Type II error in classical analyses that mainly compare the means.

The advantage of the SW as a measurement tool is its ability to parse the distribution for these features by its three-parameter *decomposition of the central tendency*, in which as noted before, $E(X) = \alpha/\gamma+\theta$. In total, noteworthy advantages of the SW may include: (1) the whole distribution being fit across each experimental manipulation; (2) experimental manipulations being quantified along three kinds of distinct distribution outcomes; (3) observed RT data means being decomposed according to their distributional make-up; (4) observed means with similar values may be revealed rather as markedly different; and (5), the RT data being fit on its natural scale by the SW, with no need for an inherently-imperfect approximation to the normal. These benefits can be further supported by early works expounding the importance of accounting for the full RT distribution by Luce (1986); also some of the aforementioned benefits are explicitly discussed by Balota et al. (2008); Balota & Yap (2011) yet in the context of the ex-Gaussian, which is also an excellent distribution measurement tool, but does not have this direct correspondence to a latent accumulation process.

Utilizing the Shifted Wald

Whether one decides to utilize the SW as a distribution measurement tool, or as an accumulation model of the data, the approach of use is the same: to simply fit the distribution, which is to estimate its three parameters. It is the same approach since the parameters of the SW simultaneously describe both the shape of the RT distribution, and the data in the context of latent accumulation to threshold.

Application to Simulated Data

We developed a fitting method that combines techniques of deviance criterion minimization of observed-versus-predicted quantile distance, and maximum likelihood (ML) estimation, to fit the model parameters. The approach is summarized as follows. In the case of the SW, given a parameter value for its shape β, the other two parameters, θ and α, may be determined by closed-form ML estimators, developed as in Nagatsuka & Balakrishnan (2013). Parameter γ is then obtainable as $\gamma = 1/\alpha\beta$. An algorithm searches the near-entire space of β, and for each β, computes the model-predicted quantiles in the near-full range at high resolution (e.g. 100 equally-spaced quantiles between the .02-.98, or .001-.999 range depending on choice of fitting outliers in the data).[2] The parameter set that leads to the smallest absolute difference in the observed-versus-predicted quantiles is selected as the fit.

We have found the SW in the context of this method, to be robust in the recovery of parameters during cases of both small numbers of observations $N = 50$, as well as large $N = 1000$; the fitting procedure finishes on the level of seconds using standard computing technology, and can be simply performed via R or MATLAB. The following table contains the simulation recovery results, which are the average Pearson r correlations between the model fit and generating parameters across 1000 recovery simulation trials; each row corresponds to the recovery of a different data set size (e.g. number of observations), N.

Table 1: Parameter Recovery, Average Pearson Correlations

Observations	γ	α	θ
$N = 1000$	0.99	0.95	0.99
$N = 500$	0.98	0.93	0.99
$N = 250$	0.97	0.89	0.99
$N = 125$	0.94	0.82	0.98
$N = 50$	0.88	0.68	0.97
$N = 15$	0.72	0.47	0.92

In addition, the fits also matched the observed data quantiles very well. Given the desirable performance of the method, we utilize the approach to fit the real data.

Application to Real Data

In this section, the fitting approach is demonstrated on a data set involving a visual search (VS) task by baboons of mixed ages, collected by Goujon & Fagot (2013); and the results

[2] For more information, see Anders et al. (in review).

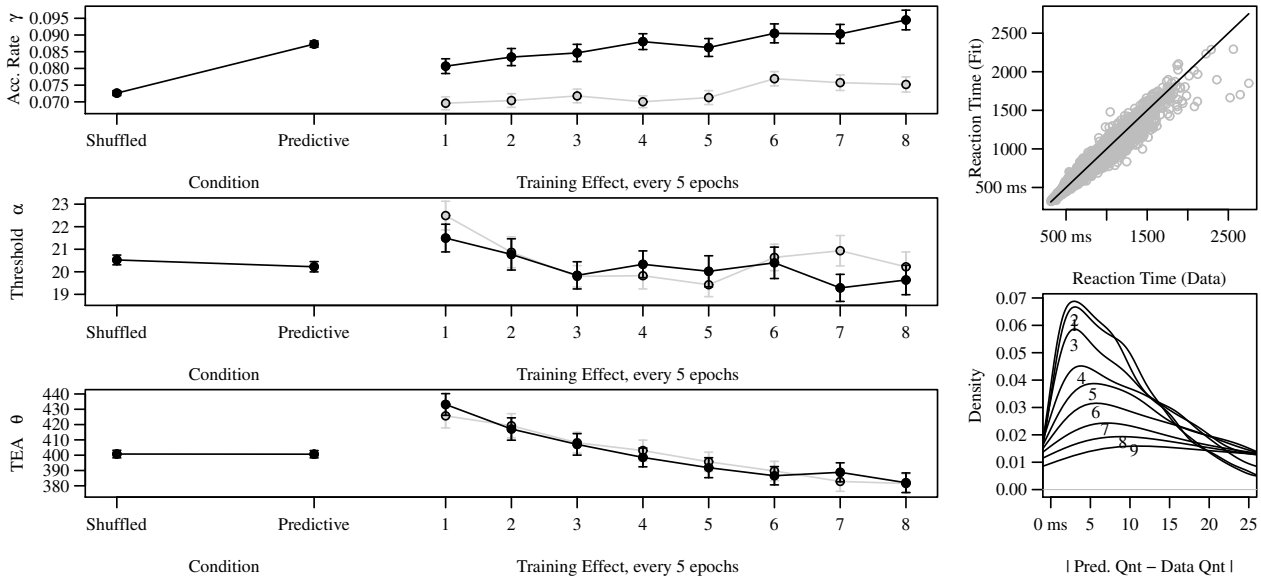


Figure 2: The SW fit to the VS task by baboons: (left) mean parameter values with bars representing standard error of the mean grouped by experiment factor, $N_C = 1080$ and $N_{Tr} = 135$, in which the grey dots show the training effect for the non-predictive (shuffled) condition, and the black for the predictive condition; (top-right) fitted-versus-observed, quantile-quantile plot for the $N = 2,158$ distributions fit for all deciles; (bottom-right) the distribution of residuals for the nine deciles across the distributions.

will be presented in the vocabulary of the SW as a latent accumulation model for the data. The experimenters explored an animal model (via baboons) of statistical learning mechanisms in humans, specifically the ability to implicitly extract and utilize statistical redundancies within the environment for goal-directed behavior. Twenty-five baboons (species *Papio papio*) were trained to perform a VS task with contextual cueing. The task consisted of visually searching for a target (the letter "T") that was embedded within configurations of distractors (letters "L"), which were either arranged predictively to locate the target (hence a contextual cue), or non-predictively (shuffled, without a cue).

As organized by the original researchers, there are three meaningful partitions: the $C = 2$ predictive vs. non-predictive contextual cue conditions; the $E = 40$ time-points (epochs) to observe training effects, in which every unit step in E consists of 5 blocks (each block contains 12 trials, and thus each E contains 60 trials); and the $B = 27$ individual baboons. These three meaningful factors provide for $N = 2158$ separate distributions to each be individually fit by the SW. The average distribution length (number of observations) is $\bar{L} = 30$, with standard deviation, $\mathrm{SD}(L) = 1.10$.

Figure 2 provides the results of the analysis on the baboon VS task. The left column of three plots respectively contains the means, and their standard errors, of the model-fit measurements of three SW parameters: γ, α, and θ, grouped by experiment factor: condition ($N_C = 1080$) and training effect, in which the levels are averages of every five proceeding epochs, to simplify the illustration ($N_{Tr} = 135$ per each of two conditions). Beginning with consideration of condition, the model clearly isolates the effect of condition (non-predictive vs. predictive) on a single parameter, the accumulation rate of signal strength (or target detection), γ; note that the standard errors of the mean in this case are in fact too small to be seen in the plot. The other parameters, α and θ, which in this task might be respectively interpreted as a certainty criterion before responding, and mechanical response/visual processing RT speed, showed no substantive change across conditions.

Next, the analysis of training effects over time are displayed for each contextual cue condition: the predictive condition in black points, and the non-predictive (shuffled) condition—which provides little information (e.g. cues) to learn from while doing the task–in grey points. The training effect appears to adjust each of the parameters over time in a way that supports faster RTs, yet interestingly in different ways. Most notably, the TEA parameter, θ, for mechanical/perceptual RT processes, benefits equally by training across epochs during both conditions–which is a rather plausible finding—as does the response caution / signal criterion parameter, α. In contrast, there is a marked difference across conditions in the benefit rate of the signal accumulation parameter, γ, by training.

Furthermore, while each of the SW parameters appears to be modulated by training, they differ in their rate of change over time, and their onsets/magnitudes of diminishing returns. For example, γ appears to benefit in a consistently-increasing linear fashion from levels 1 to 8; while α and θ speed benefits occur in uniquely different curvi-linear fash-

ions, with different diminishing or zero-return onsets, respectively near training points 5 and 7.

The right column of plots in Figure 2 provide model goodness-of-fit checks to verify if the observed data quantiles are appropriately fit by the SW. The top plot contains the deciles of all $N = 2158$ distributions fit with the SW; as one can see, nearly all of the fits match the observed deciles well in a corresponding $x = y$ fashion, with very few outliers. The bottom plot provides the distribution of residuals for each of the nine deciles across the 2158 cells fit; here it is shown that most of the deciles are similarly well fit, with a slightly larger variance for the deciles 7-9, which tend to also hold increasingly larger RT magnitude and variation in the observed data.

Discussion

The utility of the SW distribution, to serve as a cognitive model for certain response tasks by describing the data in the context of accumulation to threshold, as well as its usefulness as an objective measurement tool for RT distributions, was presented. Noteworthy and unique aspects of the SW, which set it apart from other distributions that may be used as RT distribution measurement tools, include its flexibility to accommodate a number of distribution shapes; its three-parameter decomposition of the mean, each parameter accounting for a distinct distribution outcome; its ability to be fit well during cases of few observations; and most distinctively its unique ability to also describe the data via accumulation to threshold.[3]

Important clarifications can be made to resolve confusions between the SW distribution, particularly its accumulation model characteristic, and more complex accumulation models such as the DDM, race and LBA models. Firstly, the SW distribution is the only model of the three in which its parameters directly quantify the distribution of RTs, and simultaneously directly describe the RT data in the context of a latent quantity accumulating to threshold. Secondly, it always consists of only one accumulator modeling the response process, with one threshold.

On the latent accumulator aspect of the SW, there are only minor modifications which will deliver the researcher to one of the three other prominent models: the DDM, race, and LBA. Each of these three models have the same kind of accumulator as in the SW: the DDM instead has two thresholds: a lower and upper, to model two characteristic outcomes; and hence allows for negative drift rates, e.g. $\gamma < 0$, to allow substantial observations on the lower boundary. The race model has multiple instances of the same accumulator as the SW, to model any number of characteristic responses, in which the first accumulator that reaches the threshold wins. The LBA has this same property of the race model, except the latent quantity accumulates in a constant linear fashion (known as a "random ray"), rather than as drift with random noise.

[3]Indeed other measurement distributions (e.g. ex-Gaussian, Gumbel) may also provide excellent utility or fits of RT data. However their principal difference from the SW, is they do not possess the ability to also describe the data by accumulation to threshold.

Thus all of these approaches are indeed very closely related. For example, the SW and DDM could be said to constitute the very same supra-model: they both stem from the same family process, the Wiener process, and as mentioned, arise from only subtle differences in parameter values (see respectively Chapter 3, and pages 8–24, Chhikara, 1988; Gerstein & Mandelbrot, 1964; Jones & Dzhafarov, 2014, for more information); in which some parameterizations of the Wiener process result in a closed-form probability density function (e.g. the SW), while others will not (e.g. the DDM). They are hence simply nested models, both using the same kind of RWD, or Brownian motion process designed in (1). Therefore in the context of an appropriate data application, an attack or critique on the elements of one of these models, such as the validity of the cognitive interpretation of this latent Brownian motion process, may be considered an attack on all three models.

A concrete issue of practicality however, worth mentioning between simple models, such as the SW, that have one accumulator and one threshold for the observed response, and more complex models that seek to have separate accumulators, and/or thresholds for every response option, is the large benefit of the SW in the context of the *limitation of the data*. More specifically, the limitation of the number of observations available in the data, per experimental manipulation and per response alternative, can be a problem that is exacerbated much more quickly as one increases in the extra numbers of accumulators and/or thresholds that more complex models have. For example in our baboon data application, the depth that we explore the experimental manipulations, estimating individual parameters for each combination of them, by participant, is a resolution that would not have been appropriate for the other more complex models. This is because there were insufficient amounts of observations for each response alternative, per experimental manipulation, to drive the estimation of the other models' extra parameters that arise from additional accumulators / thresholds; these models would be attempting to model data, with far too many missing observations. Thus it is important to take into account the number of observations per data cell sought to be analyzed: when selecting (1) an accumulator model variant, and (2) the depth that one parameterizes the model across cells; e.g. having enough observations (such as > 30) for each extra response option modeled, per cell fit.

The limitation of the SW, for being applied to data with many response alternatives observed per experimental manipulation, is it lacks the ability to serve as a complete generative model for the full data. For example, considering data with substantial amounts of both corrects and errors, the SW can be applied separately to the corrects and errors. Here it may serve as a distributional measurement tool to quantify distribution differences across conditions, and/or deliver a latent accumulation account across experimental manipulations, conditional on the respondent providing that observed (correct or error) response. However in this case, the SW

cannot serve as a full generative model, for example to produce near the same number of observed number of corrects and errors, by only knowing the parameters alone, and not how many were corrects and errors were observed in the first place. In contrast, a model such as the DDM, race, or LBA, can not only serve to account for differences between the experimental manipulations, but also as a complete generative model for the data, by having the *a priori* probability of a correct or error response, already pre-coded in the model, by being in the respective drift rates for each experimental manipulation; and thus are excellent tools for these multi-response option cases.

Thus in each model having its unique assumptions, benefits, and restrictions, it is up to the researcher to select the model(s) that best suit his or her research aims within the particular application. While there are certainly appropriate situations and data that could considerably benefit from a SW analysis approach, currently there are very few publications in the psychological literature that utilize the distribution. We hope to have advocated the distribution's use, as well as to have facilitated a deeper understanding of the SW, and its position in the context of accumulation modeling.

Acknowledgments

We acknowledge funding by the European Research Council under the European Community's Seventh Framework Program (FP7/2007-2013 Grant agreement n^{o} 263575), and the Brain and Language Research Institute (Aix-Marseille Université : A*MIDEX grant ANR-11-IDEX-0001-02 and LABEX grant ANR-11-LABX-0036). We thank the "Féderation de Recherche 3C" (Aix-Marseille Université) for institutional support, as well as Joël Fagot, for making available his data.

References

Anders, R., Alario, F.-X., & van Maanen, L. (in review). The shifted Wald distribution for response time data analysis, .

Andrews, S., & Heathcote, A. (2001). Distinguishing common and task-specific processes in word identification: A matter of some moment? *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *27*, 514.

Balota, D. A., & Yap, M. J. (2011). Moving beyond the mean in studies of mental chronometry the power of response time distributional analyses. *Current Directions in Psychological Science*, *20*, 160–166.

Balota, D. A., Yap, M. J., Cortese, M. J., & Watson, J. M. (2008). Beyond mean response latency: Response time distributional analyses of semantic priming. *Journal of Memory and Language*, *59*, 495–523.

Brown, S., & Heathcote, A. (2008). The simplest complete model of choice response time: Linear ballistic accumulation. *Cognitive Psychology*, *57*, 153–178.

Chhikara, R. (1988). *The Inverse Gaussian Distribution: Theory, Methodology, and Applications* volume 95. CRC Press.

Gerstein, G. L., & Mandelbrot, B. (1964). Random walk models for the spike activity of a single neuron. *Biophysical Journal*, *4*, 41–68.

Goujon, A., & Fagot, J. (2013). Learning of spatial statistics in nonhuman primates: Contextual cueing in baboons (papio papio). *Behavioural Brain Research*, *247*, 101–109.

Heathcote, A. (2004). Fitting Wald and ex-Wald distributions to response time data: An example using functions for the s-plus package. *Behavior Research Methods, Instruments, & Computers*, *36*, 678–694.

Heathcote, A., Popiel, S. J., & Mewhort, D. (1991). Analysis of response time distributions: An example using the stroop task. *Psychological Bulletin*, *109*, 340.

Jones, M., & Dzhafarov, E. N. (2014). Unfalsifiability and mutual translatability of major modeling schemes for choice reaction time. *Psychological Review*, *121*, 1.

LaBerge, D. (1962). A recruitment theory of simple behavior. *Psychometrika*, *27*, 375–396.

Luce, R. D. (1986). *Response Times: Their Role in Inferring Elementary Mental Organization*. 8. Oxford University Press.

Nagatsuka, H., & Balakrishnan, N. (2013). A consistent method of estimation for the parameters of the three-parameter inverse Gaussian distribution. *Journal of Statistical Computation and Simulation*, *83*, 1915–1931.

Ratcliff, R. (1978). A theory of memory retrieval. *Psychological Review*, *85*, 59.

Ratcliff, R., Gomez, P., & McKoon, G. (2004). A diffusion model account of the lexical decision task. *Psychological Review*, *111*, 159.

Ratcliff, R., & McKoon, G. (2008). The diffusion decision model: Theory and data for two-choice decision tasks. *Neural Computation*, *20*, 873–922.

Ratcliff, R., & Murdock, B. B. (1976). Retrieval processes in recognition memory. *Psychological Review*, *83*, 190.

Ratcliff, R., & Rouder, J. N. (1998). Modeling response times for two-choice decisions. *Psychological Science*, *9*, 347–356.

Staub, A., White, S. J., Drieghe, D., Hollway, E. C., & Rayner, K. (2010). Distributional effects of word frequency on eye fixation durations. *Journal of Experimental Psychology: Human Perception and Performance*, *36*, 1280.

Van Maanen, L., Grasman, R. P., Forstmann, B. U., & Wagenmakers, E.-J. (2012). Piéron's law and optimal behavior in perceptual decision-making. *Frontiers in Neuroscience*, *5*.

Van Zandt, T. (2000). How to fit a response time distribution. *Psychonomic Bulletin & Review*, *7*, 424–465.

Van Zandt, T. (2002). Analysis of response time distributions. *Stevens' Handbook of Experimental Psychology*, .

Wagenmakers, E.-J., & Brown, S. (2007). On the linear relation between the mean and the standard deviation of a response time distribution. *Psychological Review*, *114*, 830.

Password Entry Errors: Memory or Motor?

Kristen K. Greene (kristen.greene@nist.gov)
National Institute of Standards and Technology, 100 Bureau Drive, MS 8940
Gaithersburg, MD 20899-8940 USA

Franklin P. Tamborello, II (franklin.tamborello.ctr@nrl.navy.mil)
National Research Council Postdoctoral Research Associate, 4555 Overlook Ave SW
Washington, DC 20375 USA

Abstract

As we increasingly rely upon our computer information systems to store and operate on sensitive information, the methods we use to authenticate user identity also become more important. One of the most important such methods is the password. However, passwords that provide better security also tend to be more difficult to remember. They also tend to be difficult to type, and typing errors can have negative consequences such as being locked out of a critical information system. We present a computational cognitive model of password rehearsal and a typing extension to the ACT-R cognitive architecture intended to study human-computer interaction issues in the usable security domain.

Keywords: Human-Computer Interaction; Learning; Memory; Typing; Human Error

Introduction

As cyber attacks on user-chosen passwords abound, there are large-scale, long-term research efforts underway (e.g., the National Strategy for Trusted Identities in Cyberspace, 2011) to ultimately replace passwords as an authentication mechanism. In the near-term however, password research is still important, as better understanding of the cognitive and perceptual motor components of creating, rehearsing, recalling, and typing passwords is necessary to help inform password policies and password requirements. Furthermore, even as alternative authentication mechanisms become more prevalent (e.g., biometrics), there will undeniably be legacy systems reliant upon passwords for quite some time.

While the importance and impact of password research is clear, it can be difficult to obtain real-world password data due to security and privacy concerns, or in the case of leaked password datasets, due to ethical concerns. It would be prohibitively expensive and time-consuming to collect laboratory data from large numbers of participants across relevant password requirements, specifically different combinations of password rules for length and complexity. There are also issues of experimental control versus external validity; in researching password requirements, does one assign passwords or have participants generate their own?

As in other domains where access to human data can be challenging, behavioral data from existing password experiments can be supplemented with predictive models of human performance. Unfortunately, most password studies do not collect sufficiently detailed data to assess model validity and plausibility. The cybersecurity and modeling fields could both benefit from computational cognitive models across a variety of password-related tasks: initial learning and rehearsal strategies; recall and entry of well-memorized passwords; and cross-platform (i.e., desktop versus mobile) password typing. The current work focuses on support for modeling desktop password rehearsal and typing, specifically for complex, system-generated passwords found in higher-security enterprise environments.

Transcription Typing Versus Password Typing

There is certainly a large and longstanding body of expert typing and transcription typing literature (e.g., Coover, 1923, Gentner, 1981, Salthouse, 1986), including examination of a variety of factors such as age and skill (e.g., Salthouse, 1984). However, there are several important distinctions between general transcription typing and password typing.

In the higher-security enterprise environments for which the current work is intended, passwords are quite different from words—in fact, most password policies explicitly prohibit the sole use of words, as dictionary attacks on passwords are so successful, dating back to the late 1970s (Morris & Thompson, 1979). Higher-entropy passwords differ quite significantly from the words commonly found in most traditional transcription typing experiments. “Better” passwords are supposed to be as random as possible in order to make guessing them more difficult; they should not follow orthographic rules as do regular words. Therefore, when typing complex passwords, we cannot leverage many of the benefits of natural language. Beyond simple inclusion of lowercase and uppercase letters, most higher-entropy passwords also include numbers and special characters, making it difficult or impossible to leverage error correction techniques during password typing. Furthermore, password text is usually masked, whereas normal text is not. These factors may contribute to changes in strategy for carefully typing passwords in comparison to normal transcription typing.

In addition to behavioral studies of transcription typing (see Salthouse, 1986 for a good review), there have also been cognitive models of the task. By far the most comprehensive and well-known computational cognitive model of transcription typing is Bonnie John’s TYPIST (John, 1996). John’s TYPIST quantified 19 of the 29 previously reviewed Salthouse (1986) phenomena as well as two additional phenomena. While TYPIST quantified transcription typing along the time dimension with scheduling charts, it did not simulate decreased performance

variability with higher typing skill, nor brain areas' activation patterns, as have more recent queuing network models (e.g., Wu & Lui, 2004). Regardless, to the best of the authors' knowledge, there does not currently exist an ACT-R model of rehearsal and typing for complex, system-generated passwords on a standard desktop QWERTY keyboard. The current work is a necessary first step to begin addressing this gap.

Typing System-Generated Passwords

Prior Work

The current work was motivated by a desire to use cognitive modeling as an error exploration technique to supplement prior usable security research. The primary goal was to better understand the underlying cause of errors reported in a recent study of complex password entry on desktop computers (Stanton & Greene, 2014). This section describes relevant methodology and results of interest from said study.

Method In the Stanton and Greene (2014) study, participants were given ten system-generated[1] passwords in a random order. Passwords ranged in length from six to 14 characters (see Table 1).

Table 1: Stimuli (Stanton & Greene, 2014).

Password	Length
5c2'Qe	6
3.bH1o	6
m3)61fHw	8
ua7t?C2#	8
p4d46*3TxY	10
q80<U/C2mv	10
d51)u4;X3wrf	12
6n04%Ei'Hm3V	12
m#o)fp^2aRf207	14
4i_55fQ$2Mnh30	14

Each password had to contain at least one uppercase letter, one lowercase letter, one number, and one special character. Passwords could not end with an exclamation mark, nor could passwords begin with a capital letter. Note the variety of symbols in the preceding stimuli, many of which are not supported for typing in standard ACT-R.

Two groups of participants were tested in the Stanton and Greene (2104) study. One group was from the Washington, DC (WDC) metropolitan area in the United States, and the other was from the University College London (UCL) in the United Kingdom. This sampling distinction is important, as results differed somewhat by participant group. The authors proposed that this might be due to differences in age and/or typing ability between the two groups, as the WDC group was older than the younger UCL group, which was composed of mainly undergraduates.

In the Stanton and Greene (2014) study, participants received one password at a time, and for each password, had to complete a series of three tasks: practice, verification, and entry. During practice, the password was visible, and participants could practice typing the password in a large text field. Participants could practice typing the password as many or as few times as they wished. There was no feedback given during the practice task, and typed text was visible (i.e., not masked as in a regular password field).

During verification, the password was not visible, and participants had to enter the memorized password correctly in order to move onto the entry screen. Typed text was visible (i.e., not masked) during verification. If participants failed the verification task, they could continue to attempt verification, or choose to return to the preceding practice screen to practice the password again. Regardless, after participants completed the verification task, they moved onto the entry task.

During entry, participants had to enter the memorized password ten times. On the entry screen, the password was not visible, nor was typed text visible. Instead, it was masked with asterisks, as password fields tend to be in use. After participants completed the three phases—practice, verification, and entry—for all ten passwords, they received a surprise recall test. During the surprise recall test, typed text was visible (i.e., not masked).

Note that although modeling cognitive rehearsal and disambiguating memory from motor errors were the foci of the current work, it was necessary to include a description of the larger experiment here as well, since expanding the current model to account for additional phases of the experiment is potential future work. Furthermore, planning for future model expansion to address those additional experimental tasks was influential in determining implementation of the current password typing model.

Results Here we focus on errors rather than timing results from the Stanton and Greene (2014) study. Both are important to test the validity of a model, but the decision to emphasize errors rather than times for password typing parallels their importance in the real world. Accounts are often locked for too many erroneous login attempts, but it is virtually unheard of for a user to be locked out for typing too slowly. Furthermore, knowing which error categories were most prevalent was helpful for determining where to focus our modeling efforts, as well as for evaluating model plausibility.

There were several error classes reported in the aforementioned study. Table 2 presents results from Stanton and Greene (2014) in order of decreasing error category prevalence. Note that the table is ordered based on total

[1] Advanced Password Generator from BinaryMark was used. Disclaimer: Any mention of commercial products or reference to commercial organizations is for information only; it does not imply recommendation or endorsement by the National Institute of Standards and Technology nor does it imply that the products mentioned are necessarily the best available for the purpose.

error percentages; the order would differ slightly if based on either the WDC or UCL participant group.

Table 2: Categorized errors (Stanton & Greene, 2014), rounded to nearest percentage.

Error Category	WDC	UCL	Total
Incorrect capitalization	38%	51%	45%
Missing character	25%	10%	17%
Adjacent key	8%	10%	9%
Wrong character	6%	12%	9%
Transposition of characters	10%	6%	8%
Extra character	7%	7%	7%
Zero instead of an "O"	3%	3%	3%
Wrong place within password	3%	1%	2%

It should be immediately obvious from Table 2 above that incorrect capitalization was the largest class of errors for both participant groups. It is by far the most interesting class of errors to model for several reasons: 1) the sheer magnitude of the incorrect capitalization error class in comparison to other error categories, 2) the practical significance of that error class given current password requirements, and 3) the interesting theoretical question posed by the nature of that particular error.

1) At 45% of the total error corpus, incorrect capitalization errors were nearly three times as likely as the second most prevalent error class (missing character errors, 17% total), and incorrect capitalization errors were five times as likely as the third most prevalent error categories (adjacent key and wrong character errors, each 9% total).
2) The fact that the most frequently occurring error was incorrect capitalization is quite significant given that most modern password policies require at least one uppercase letter. Furthermore, the majority of special characters—also required by many password policies—require shifting. Twenty-one of the total 32 possible symbols require shifting, whereas only 11 do not.
3) A particularly interesting point about incorrect capitalization errors is that based purely on the behavioral data reported in Stanton and Greene (2014), it is unclear whether those errors were memory errors or motor execution errors. Answering this question would help inform typing theory specifically for complex passwords.

An ACT-R Model of Password Rehearsal

Before enabling ACT-R to type capital letters, a cognition-only (i.e., no use of the motor module) model of password rehearsal was constructed, to test whether it alone could account for the errors seen in the behavioral data.

Stimulus Selection

Given the artificiality of having people learn 10 randomly generated passwords in a single session, rather than attempt to model the entire stimuli set at once, a single password was selected for the initial model: *q80<U/C2mv*. This allowed the model focus to be on the cognitive phenomena of interest: rehearsal and retrieval of a single password, which is more reminiscent of a real-world scenario, where we attempt to rehearse a newly generated password to login to a single account. Why select that particular password though? Of the 10 passwords in Table 1, the "q80" password seemed the most interesting to model for several reasons. First, at 10 characters long, it was one of the two middle length passwords. (The shorter passwords are really too easy by today's more stringent password rules, as most higher-security enterprise environments require a minimum length of 10 to 12 characters.) Of the two length-10 passwords, the "q80" password had two non-alphanumeric symbols, whereas the other length-10 password had only a single non-alphanumeric symbol. This is important, as prior work on the linguistic and phonological difficulty of system-generated passwords suggested chunking passwords between non-alphanumeric symbols (Bergstrom, et al. 2014). Furthermore, when asked, people consistently verbalize that password as "Q eighty is less than U over C two M V" and that they "break the password up" at the non-alphanumeric symbols. Since there were no interview data asking people about their chunking or rehearsal strategies reported in the Stanton and Greene (2014) desktop study, it seemed reasonable to use such qualitative observations to inform the current desktop password rehearsal model. When verbally reciting a password, it certainly makes sense that people might chunk passwords (at least initially) in similar ways across platforms.

Model Implementation

For the initial model, the password was broken up into the following chunks based on splitting it at the non-alphanumeric symbols:

1) q80
2) <
3) U
4) /
5) C2mv

In the model's declarative memory, chunks were encoded with their contents, an ID, and a pointer to the next chunk in the sequence. A more complete model of the task would build up these chunks character-by-character. However, since participants in the Stanton and Greene (2014) study were allowed to practice each password as many or as few times as they wished, the initial practice strategies and number of practice repetitions that would account for building up such a representation could vary widely. Rather than implement different models to simulate a variety of practice methods, the model assumes the initial pieces of the password are starting knowledge, and employs a very simple rehearsal strategy. It cycles through chained retrieval of the various chunks in the password to mentally rehearse the stimulus for a period of time that is settable by the modeler. Since ACT-R did not natively support typing the less-than symbol, nor did it support typing errors of any kind, rather than having the model type the retrieved chunks, it simply output them to a file.

The set-similarities option in ACT-R benefits from a principled, ideally a priori, hypothesis as to the nature of the similarities between chunks. In this case we assume that non-alphanumeric symbols are more similar to, and thus more confusable with, one another than are letters to non-alphanumeric symbols, and letters are more similar to one another than are letters to numbers. However, the exact value to assign each similarity is still an open question, and there are 10 such pairwise similarities to set. This seems less than ideal for the current password, and even worse when considering longer passwords that contain a greater number of chunks.

Although the model did predict the nature of the jump-transposition errors humans made (where they transposed the two symbols that were separated by a single letter), it could not account for failure to capitalize the "U", nor could it account for failure to capitalize the "C", which were errors seen in the Stanton and Greene (2014) study. As capitalization errors were by far the most prevalent error in said study, a mechanism for typing capital letters in ACT-R was sorely needed.

Investigating the source of password entry errors is a perfect application opportunity for cognitive modeling to shed light on the root cause of an error (or errors) that was difficult to ascertain through prior behavioral data alone. By implementing support for an ACT-R model that can type capital letters, one could then test different models to see whether those incorrect capitalization errors were memory errors or motor execution errors (where a shift key press had been attempted but simply not executed properly, such as by prematurely releasing the shift key). The ability to type capital letters raises interesting theoretical questions. For each letter of the alphabet, do people have two distinct versions in their memory, one lowercase and one uppercase? Or is an uppercase letter encoded as the lowercase plus a required shift action?

Implementation Issues in ACT-R

In order to support modeling of incorrect capitalization typing errors, two limitations in ACT-R first required addressing: missing special characters, and lack of case-sensitivity in typing.

Missing Special Characters Of the special characters in Table 1, ACT-R previously included support only for the period, semicolon, slash, and quote (Bothell, 2014, see "key" on page 320 of the ACT-R Reference Manual). Therefore, in order to enable modeling typing of the remaining symbols in Table 1 (right parenthesis, question mark, number sign, asterisk, less-than sign, percent sign, caret, underscore, dollar sign), it was necessary to address the somewhat limited prior support for non-alphanumeric symbol typing. As we want to support modeling of *any* possible password, not merely those in Table 1, we added support for all remaining ASCII printable characters not previously supported by ACT-R.

Lack of Case-Sensitivity Regardless of whether calling ACT'R's "press-key" motor module request (Bothell, 2014, see page 317 of the ACT-R Reference Manual) with a capital or lowercase letter, the output will be the same in ACT-R's current instantiation. This is somewhat problematic for modeling incorrect capitalization errors, which requires that ACT-R be capable of press-and-hold capability for the left and right shift keys, combined with a simultaneous key press of a second key (i.e., chorded typing). Therefore we added to ACT-R a capability to type key chords and output case-sensitive text, as described in the following section.

Stochastic Typing Extension for ACT-R

The standard ACT-R distribution (Anderson, et al, 2004; Anderson 2007) does not commit any typing errors as a matter of motor error (Bothell, 2014). However, real humans, even very skilled typists, are imperfect, and tend to err at rates from 0.5% to 35% (Salthouse, 1986; Panko, 2008; Landauer, 1987). We wished to explain password entry errors, but because some errors are due to memory processes and some are due to motor processes, we had to extend our modeling framework of choice, ACT-R, so that it, too, would be capable of such motor errors. Furthermore, we needed to implement the low-frequency, non-alphanumeric characters that information systems often require their users to incorporate into their passwords as a matter of security policy, e.g. "*" or "?". Source code for the ACT-R stochastic typing extension may be downloaded from https://github.com/usnistgov/CogMod.

Motor Errors in Typing

Our typing extension for ACT-R redefines some of ACT-R's existing code so that any requested typing action can stochastically result in the output of a typed key other than the one intended. To do so it adapts the ellipsoid motor movement error equation of May (2012) and Gallagher and Byrne (2013), which leads to greater error along the axis of movement than off the axis, the off-axis error being scaled to .75 of the on-axis. However, because here the units are keys rather than pixels as in May's study, and ACT-R assumes most keys are the same width, the width term in May's equation is simplified to 1.

Hold-Key Because typing non-alphanumeric characters typically involves holding a shift key while striking another key, and standard ACT-R provides no way to hold any such modifier key, it was necessary to invent such a method. Our errorful typing extension provides two motor module request extensions (see "extend-manual-requests" on page 325 of the ACT-R Reference Manual, 2014) to enable the holding and releasing of modifier keys such as shift.

The new hold-key motor module request acts like press-key, translating the requested key to be held into a peck movement (Bothell, 2014, pp. 315-6) with the appropriate features. Once the hold-key motor movement is executed, ACT-R will have a state indicating that the appropriate key is being held. This state in turn causes ACT-R to now output a different character for the same press-key requests that follow for the given keys. The model can request the release-key function to release the given modifier key and end the modifier key state.

Additional Characters With a shift key held, ACT-R can now type a set of ASCII-compatible, non-alphanumeric characters such as "*" and "?." It can now also type capital letters as well as lower-case letters, a critical feature for case-sensitive passwords that standard ACT-R lacks.

Discussion and Future Directions

To address the question of setting appropriate chunk similarities in the initial password rehearsal model, a revised model is underway that has restructured the chunks in declarative memory, and does not use partial matching and set-similarities, instead relying upon spreading activation. This new model is now ready to interface with the stochastic typing extension.

Beyond using the new typing extension, one obvious expansion of the model would be to account for additional phases of the Stanton and Greene (2014) experiment, such as the initial practice and verification tasks. The model should also be expanded to test against the remaining nine passwords and additional stimuli. Modeling the experiment in its entirety would require interfacing with a real or virtual window to control presentation of the stimuli; this would allow the model to visually obtain the stimuli and build up representations of each password in the imaginal buffer letter-by-letter. As an initial model of a larger complex experiment, it seemed more prudent to focus the current work on a single interesting phenomenon, in this case, support for disambiguating memory from typing errors.

We chose to focus on support for disambiguating the most prevalent error in the Stanton and Greene (2014) study, which was incorrect capitalization. As "missing character" was the second most common class of errors in said study, the current stochastic typing extension for ACT-R should be modified to support typing omissions. Furthermore, there are further refinements we would like to make to the ACT-R typing extension to reflect other systematic effects that we did not yet incorporate, such as the likelihood of specific error classes should depend on which fingers are pressing which keys. For example, in traditional transcription typing studies, omissions are more likely with the weaker little finger. Adding support for ACT-R sensitivity to finger/key combinations would benefit future work.

In the future, it would be informative to construct models of the data from the Washington, DC and University College London groups separately to investigate age effects and/or typing skill differences suggested in the Stanton and Greene (2014) desktop password typing study. This would first necessitate updating ACT-R's virtual keyboard to support a standard UK QWERTY keyboard layout. We could then explore modeling parameters for older adults, as recent research suggests that they are task- and device-dependent, and strategy may interact with task and device (Howie, 2015). A deeper understanding of participants' rehearsal and memorization strategies would help inform and test future models.

Regardless of platform, it is important that ACT-R have the ability to commit motor errors when typing so that we can model both memory and typing components of password entry tasks. This is critical to determine which parts of the task are platform-agnostic versus platform-dependent. We should test the password rehearsal model on mobile password typing for smartphones and tablets. Clearly the stochastic typing extensions for ACT-R that we created for modeling desktop password typing would not be appropriate for modeling interactions with mobile keyboards. Instead, we could utilize recent work by Gallagher (2015) and Gallagher and Byrne (2015) on mobile password typing. No doubt device interacts with password complexity, but it would be interesting to see how the initial password learning and rehearsal is affected by device constraints. Are basic password rehearsal strategies similar across devices? A model that utilized the articulatory loop for rehearsal could be viable across multiple platforms.

We think typing differs qualitatively between platforms, especially between desktop and mobile touchscreen computers. Motor scheduling errors should occur in desktop typing when people are typing in parallel and depressing two keys simultaneously. Mobile password typing is more sequential (although it can be interleaved depending on one- versus two-fingered typing style) than is desktop typing. Therefore motor errors on mobile platforms should be more a matter of motor execution accuracy errors than scheduling errors. This would make sense due to the large size of the input device (i.e., a finger) in comparison to the small size of the onscreen keyboard buttons. In fact, research replicating the desktop Stanton and Greene (2014) study on mobile devices (Greene et al., 2014) found that the proportion of adjacent key errors was significantly greater on a smartphone than on a tablet, and the smartphone adjacent key errors were more than twice as prevalent as in the desktop study. Testing the current password rehearsal model across platforms would contribute significantly to disambiguating typing from memory errors.

Regardless of platform, comparison of current and future model predictions to human data could utilize more quantitative measures for comparing errors between passwords. For example, a measure of edit distance such as the Levenshtein distance or the Damerau-Levenshtein distance would be appropriate (Navarro, 2001). Both of these metrics measure differences between sequences based on the number of edit operations required to change the given string into the target sting. However, the former only allows insertions, deletions, or substitutions, while the latter allows those and also transpositions.

Overall, this work illustrated several challenges in modeling a dataset not originally intended for modeling. For example, we do not know if participants in the Stanton and Greene, (2014) study were touch typists, and ACT-R assumes a "moderately skilled touch typist" (Bothell, 2014, see page 317 of the ACT-R Reference Manual). While we made significant progress constructing a model and extending ACT-R's typing ability to better model previously reported behavioral data, it would be ideal to conduct an entirely new study for model validation purposes. A study specifically designed to inform and test model predictions should include more controlled practice with feedback and reinforcement; assign participants fewer passwords but force them to practice them many more times; use a within-

subjects design to test password entry across multiple devices (i.e., desktop and mobile); include a baseline typing test to assess whether participants are touch typists; and explicitly query participants regarding their chunking and rehearsal strategies.

Although there is certainly much work that remains to be done, we feel the current effort was an important first step toward testing theories of password learning and typing on what is still the most prevalent platform for text-heavy tasks: the desktop computer. We now have the capability to begin disentangling memory from motor errors. Both memory and motor are sources of error that must be addressed separately, but that interact with each other within a single integrated system, people.

Acknowledgments

This research was performed while Dr. Tamborello held a National Research Council Research Associateship award at the US Naval Research Laboratory.

A special thanks to Dr. Stefan M. Wierda for his mentorship of Dr. Greene during the 2014 ACT-R Master Class.

References

ACT-R Research Group. (2013). *ACT-R: Theory and architecture of cognition.* [Web page] Retrieved from http://act-r.psy.cmu.edu/

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological Review, 111(4)*, 1036-60. doi: 10.1037/0033-295X.111.4.1036

Anderson, J. R. (2007). *How can the human mind exist in the physical universe?* New York, NY: Oxford University Press. Retrieved from Google Scholar.

Bergstrom, J. R., Frisch, S. A., Hawkins, D. C., Hackenbracht, J., Greene, K. K., Theofanos, M. F., & Griepentrog, B. (2014). Development of a scale to assess the linguistic and phonological difficulty of passwords. In Cross-Cultural Design. Lecture Notes in Computer Science, Volume 8528, 131-139

Bothell, D. (2014). *ACT-R 6.0 Reference Manual.* ACT-R Research Group. Retrieved from act-r.psy.cmu.edu

Coover, J. E. (1923). A method of teaching typewriting based upon a psychological analysis of expert typing. National Education Association, 61, 561-567.

Gallagher, M. A. (2015). Modeling Password Entry on Mobile Devices: Please Check Your Password and Try Again. Doctoral Dissertation, Rice University, Houston TX.

Gallagher, M. A., & Byrne, M. D. (2015). Modeling Password Entry on a Mobile Device. To appear in *Proceedings of the International Conference on Cognitive Modeling.*

Gallagher, M. A., & Byrne, M. D. (2013). The devil is in the distribution: Refining an ACT-R model of a continuous motor task. In *Proceedings of the 12th International Conference on Cognitive Modeling*. Ottawa, Canada.

Gentner, D. (1981). Skilled finger movements in typing. Center for Information Processing, University of California, San Diego. CHIP Report 104.

Greene, K. K., Gallagher, M. A., Stanton, B. C., & Lee, P. (2014). I can't type that! P@ssw0rd entry on mobile devices. In Human Aspects of Information Security, Privacy, and Trust. Lecture Notes in Computer Science, Volume 8533, 160-171.

Howie, N. T. (2015). The generalizability of cognitive modeling parameters for older adults. Doctoral Dissertation, Rice University, Houston TX.

John, B.E. (1988). Contributions to Engineering Models of Human-Computer Interaction, Department of Psychology, Carnegie-Mellon University, Ph.D. thesis.

John, B.E. (1996). TYPIST: A theory of performance in skilled typing. Human Computer Interaction, 11, 321-355.

Landauer, T. K. (1987). Relations between cognitive psychology and computer systems design. In J. M. Carroll (Ed.), Interfacing thought: Cognitive aspects of human-computer interaction (pp. 1-25). Cambridge, MA: MIT Press.

May, K. (2012). A model of error in 2D pointing tasks. Undergraduate Honors Thesis, Rice University, Houston, TX.

Morris, R., & Thompson, K. (1979). Password Security: A Case History. *Communications of the ACM,* 22(11): 594-597.

National Strategy for Trusted Identities in Cyberspace. Enhancing Online choice, Efficiency, Security, and Privacy. (2011). Retrieved online from http://www.whitehouse.gov/sites/default/files/rss_viewer/NSTICstrategy_041511.pdf

Navarro, G. (2001). A guided tour to approximate string matching. ACM Computing Surveys 33 (1): 31–88. doi: 10.1145/375360.375365.

Panko, R. R. (n.d.). Basic error rates. [Web page] Retrieved from http://panko.shidler.hawaii.edu/HumanErr/Basic.htm

Salthouse, T. (1984). Effects of age and skill in typing. Journal of Experimental Psychology, Vol. 113, No. 3, 345-371.

Salthouse, T. (1986). Perceptual, Cognitive, and Motoric Aspects of Transcription Typing. Psychological Bulletin, Vol. 99, No. 3, 303-319.

Stanton, B. C., & Greene, K. K. (2014). Character strings, memory, and passwords: What a recall study can tell us. In Human Aspects of Information Security, Privacy, and Trust. Lecture Notes in Computer Science, Volume 8533, 195-206

Wu, C., & Liu, Y. (2004). Modeling behavioral and brain imaging phenomena in transcription typing with queuing networks and reinforcement learning algorithms. In *Proceedings of the 6th International Conference on Cognitive Modeling.* Pittsburg, PA, USA.

Toward Expert Typing in ACT-R

Robert St. Amant (stamant@csc.ncsu.edu), Prairie Rose Goodwin (prgoodwi@ncsu.edu),
Ignacio X. Domínguez (ignacioxd@ncsu.edu), and David L. Roberts (robertsd@csc.ncsu.edu)
Computer Science Department
North Carolina State University
Raleigh, NC 53706, USA

Abstract

This paper describes an effort to integrate the TYPIST theory of expert transcription typing into the ACT-R cognitive architecture. Our goal is to strike a reasonable balance between a match to the highly accurate predictions of TYPIST and the architectural constraints imposed by ACT-R. The model we have built provides good predictions of human performance on most basic typing phenomena, though less accurately than TYPIST. We present the design of the model, a description of software to support model execution and experimentation, and the results of performance tests comparing the model's predictions with human typing data in the literature.

Keywords: Cognitive modeling; ACT-R; TYPIST; transcription typing

Introduction

TYPIST (John, 1996) is a theory of transcription typing based on the Model Human Processor (MHP) (Card, Moran, & Newell, 1983). In this paper we describe an attempt to incorporate the TYPIST theory into ACT-R.

The ACT-R model of typing we have developed gives predictions qualitatively consistent with TYPIST (though typically with lower accuracy) for fourteen typing phenomena identified by Salthouse (1986). The first contribution of this paper is an ACT-R model that reflects the basic control structure of TYPIST, with some pragmatic modifications to accommodate differences between ACT-R and the MHP. Such a model exists internal to CogTool (John, Prevas, Salvucci, & Koedinger, 2004), and typing models exist in other architectures (e.g., QN-MHP (Wu & Liu, 2004)), but to our knowledge this is the first "standalone" ACT-R model of typing. The second contribution is software to support execution and evaluation of the task of transcription typing, which may be useful in other contexts. The third contribution is a set of basic typing performance results (Salthouse, 1986). Our model does not (yet) offer new insights into typing, but it extends the scope of modeling possible with ACT-R and it has helped us identify new directions for architectural work.

An ACT-R typing model

We begin with an outline of TYPIST. John (1996, p. 326) summarizes the basic method as follows:

> TYPIST perceives a *chunk*... and encodes it into an ordered list of characters (the spelling) with a perceptual operator. If it is a word or syllable, a cognitive operator retrieves the spelling of that chunk from long-term memory (LTM). The first character in the list is initiated with a cognitive operator and then executed with a motor operator. The second character is then initiated and executed... If a letter is perceived alone, then the letter is initiated immediately following the perception and executed.

The perceptual, cognitive, and motor processors of the MHP work in parallel, while operators internal to each processor are executed sequentially; data flow requirements impose constraints on operators across the processors. For example, a cognitive operator can act on a word in working memory (WM) only after it has been made available by a perceptual operator, and motor operators to type the characters in the word can execute only after its spelling has been retrieved by a cognitive operator.

TYPIST's WM has a limited capacity. It can store up to three chunks of text that have not yet been processed by cognitive operators. This capacity constrains the perceptual processor in its ability to look ahead at words to be typed.

At a conceptual level, the ACT-R model works similarly. The biggest difference is that single-character chunks are treated the same as multi-character chunks in the requirement that their spelling be retrieved from memory. This simplification was made to limit the complexity of the model.

We describe the structure of the model mainly in terms of processing words, but the model processes at the level of syllables and characters as well. Our goals in modeling TYPIST included reproducing its structure as well as its performance, while minimizing changes to the ACT-R architecture and the parameter settings used in model execution.

In automatic pre-processing for a typing trial, a sentence is first decomposed into words separated by spaces; punctuation is treated as part of a word. Each word is then decomposed either into syllables or a combination of trigrams and bigrams, using a left-to-right greedy algorithm. Each word or syllable decomposition is stored in a *spelling* chunk in declarative memory, with an ordered set of slots $c_1 \ldots c_n$ containing characters. For modeling convenience we assume that characters are not context sensitive, and are processed as individual elements like an array. However, priming studies indicate that skilled typists perceive characters as order-dependent and are thus chained together more like a linked list (Snyder & Logan, 2014).

When the model begins executing, the goal buffer is loaded with a *typing* chunk with slots to maintain the previous visual location, the current state of visual processing, the word to be read, spelled, or typed, along with its visual location, and two variable slots for the current character in the word being typed and a one-character lookahead. In other words, the typing

chunk maintains state information for perceptual, cognitive, and motor processing.

A *find-next* and an *attend-next* production perform visual processing following ACT-R modeling conventions. When a word becomes available in the visual buffer, it is recorded by an *add-to-WM* production in "working memory" (discussed below) in two ways: it is stored in the typing chunk and combined with the previously read word in a *previous/next* chunk, to record sequencing. This chunk is added to declarative memory through the imaginal buffer.

The production *initiate-word* retrieves a previous/next chunk from memory, with the word most recently typed being the previous element. (As a special case, *initiate-first-word* fires for the very first word to be typed, which does not have a predecessor.) The *spell* production then retrieves the spelling of this word from declarative memory. The spelling chunk becomes available and is maintained in the retrieval buffer. The *char-slot* in the typing chunk is set to c_1.

Several productions initiate motor actions to type individual characters under different conditions. The basic *initiate-letter* makes a request to the motor module for the current character and advances to the next character (i.e., the typing char-slot is modified from c_i to c_{i+1}). When the current character is a space, *initiate-last-letter-in-word* fires instead, performing the same function and also requesting the retrieval of the next word to be typed. The production *initiate-last-letter-in-syllable* behaves the same way, except that it fires when a special end-of-syllable marker is encountered in the one-character lookahead. Finally, *initiate-single-letter-in-syllable* is used for single-character syllables. A new keystroke can be initiated after the preparation stage of the previous keystroke is complete, following Salthouse (1986): "[I]t is assumed that the typist is executing one keystroke while simultaneously preparing the movement patterns for the next keystroke..." We discuss motor issues in more detail later in this paper.

The ACT-R model inevitably differs from TYPIST, due to the level of modeling detail. Managing data flow dependencies is complex. Some state information is in the form of the status of buffers, but a number of flags are needed in the ACT-R model to ensure proper ordering in production firing. Productions explicitly manage memory: add-to-WM transfers a word from the visual buffer to the typing chunk and creates a new previous/next chunk in memory; initiate-word retrieves the next word to type. Visual processing is also more complex, in particular when a limited preview of text is available. Further, ACT-R visual operations take cognitive processing time, which introduces additional time at word boundaries.

Finally, TYPIST's WM capacity is not simple to reproduce in ACT-R. Without this limit, the ACT-R model looks too far ahead of its keystrokes. An ad hoc solution was implemented in the model: a count is kept such that visual processing is never more than three words ahead of cognitive processing.

Some practical limitations apply to the model. Visual processing in the model assumes that the text of the sentence is on one arbitrarily long line; there is no mechanism to move attention back to the left at the end of a line. Uppercase letters are not handled, automatically being translated to lowercase.

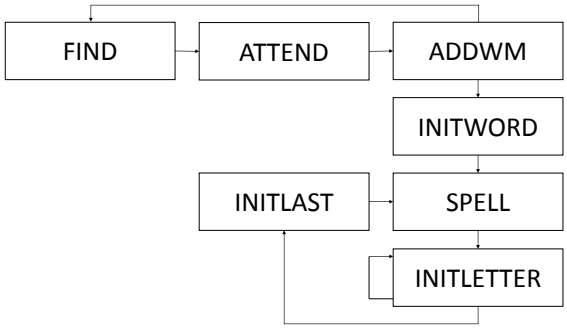


Figure 1: Graphical representation of the typing model.

Modeling support

New software was needed for model execution and experimentation. TYPIST processes text at the level of words, syllables, or letters. To support syllable functionality, we compiled a database containing the syllable decomposition of 166,000 common words. If a word is not in the database, it can be automatically broken down into trigrams and bigrams (those occurring at least 1% as frequently as the most common trigram or bigram in English) and individual letters.

The typing model depends on a small set of motor extensions to ACT-R, including a new movement style, *TYPIST-hit-key*. A finger can move to non-integer $\langle x, y \rangle$ locations, and unlike *press-key*, the finger pressing a key does not return to the home row afterwards. Further, at high typing speeds, the starting point for finger movement in a future scheduled keypress is not the current finger position; the hand/finger representation was modified to handle this possibility. Other modifications are described in the context of typing phenomena they are intended to support.

A virtual typing window displays the text to be typed, in a single line. For ease of experimentation, the window maintains text at both the word and syllable level, providing a model with either as determined by experimental settings. Variations on this window were developed to support different tests as described later in this paper.

A new virtual keyboard for typing was developed, duplicating the layout of the keyboard used for some tests of TYP-

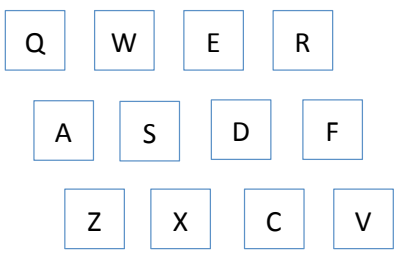


Figure 2: Portion of the typing keyboard layout.

IST (John, 1996, Fig. 16). A portion is shown in Figure 2. The typing keyboard is incomplete, including only letters, a space, a semicolon, a comma, a period, and a slash. In use, timing is very similar to the default keyboard.

Model performance

Some of John's discussion of TYPIST relies on an example sentence: "One reason is quite obvious; you can get in and out without waiting for the elevator." We used this sentence to determine basic timing for the ACT-R model as well as in some testing, as described below. With the main motor timing parameters (preparation, initialization, and burst time) at their default settings, the ACT-R model types the elevator sentence at a rate of 48 words per minute (wpm), with a mean interkey interval of 232 ms. Most of the typing tests we consider are for a 60 wpm typing rate, which can be achieved by changing feature preparation time from 50 ms to 40 ms, producing a mean interval of 207 ms. The model reaches its maximum speed of 108 wpm when these three motor parameters and the Fitts' Law coefficient are set to zero (following John's assumption that variability in typing speed is due to motor speed). TYPIST's maximum speed is 180 wpm.

TYPIST was evaluated with respect to 29 typing phenomena identified by Salthouse (1986). Twelve of these are considered basic phenomena; five concern units of typing, five errors, and the remainder skill effects (John, 1996). For many of these phenomena, TYPIST was evaluated by multiple tests. The evaluation of the ACT-R typing model is much less extensive: the basic phenomena and two units of typing phenomena are examined, and a single test is used in each case. All of the tests used to evaluate the ACT-R model were originally used for TYPIST.

Phenomenon 1. Typing is faster than choice reaction time. Salthouse (1984) describes an experiment in which participants see a stimulus (an L or an R) and type the leftmost or rightmost character key on the bottom row of the keyboard (a Z or a slash). Following the approach outlined for TYPIST, we developed a reduced model that does not store ordering or spelling information, with two new productions to map correctly between the characters. We also modified the typing window such that its contents update to a new character after each keystroke. A comparison between mean interkey interval and reaction time observed in Salthouse's experiment, TYPIST's predictions, and the ACT-R model's predictions are shown in the table below. The ACT-R numbers are from a sample run using a random string of Ls and Rs, with the baseline mean interval produced by the model typing the string as if it were a single word. Absolute errors, as a percentage of observed values, are given in parentheses.

Statistic	Target	TYPIST	ACT-R
Interval (ms)	177	195 (10.2%)	190 (7.6%)
RT (ms)	560	635 (13.4%)	505 (9.9%)

Phenomenon 2. Typing is slower than reading.

Phenomenon 3. Typing skill and comprehension are independent. These are beyond the scope of TYPIST and are not implemented in the ACT-R model.

Phenomenon 4. Typing rate is independent of word order. As with TYPIST, the treatment of words by the ACT-R model does not depend on their order of appearance. Salthouse cites a loss of 2.8% between meaningful sentences and randomly arranged words. In the ACT-R model, no differences in words per minute are seen with re-ordered words in random sentences, generated by sampling from the word database.

Phenomenon 5. Typing rate is slower with random letter order. West and Sabban (1982) describes an experiment in which participants typed easy prose sentences (EP, e.g., "I have your letter in which you ask about the prices"), sentences in which "words" were constructed by rearranging word parts but retaining the ordering of the letters (LC, letter combinations, e.g., "I veha uryo terlet ni chwhi ouy ska outab eth espric"), and sentences in which the ordering of letters in words was arbitrary (LJ, letter jumbles, e.g., "I evah uoyr rtleet ni hcihw oyu ska auobt teh rpcsei"). West and Sabban measure the percent speed increase from LJ to EP, LC to EP, and LJ to LC.

The model applies a single strategy in decomposing words: to syllables (the default behavior) for EP; to common trigrams and bigrams (the default when words are not recognized) for LC; and to individual letters for LJ (explicitly induced). For typists in the range of 55 to 69 wpm, the closest match to the model's 60 wpm, the model performs poorly, though the rank ordering of mean keystroke interval per condition is correctly predicted (EP, 231 ms; LC, 246 ms; LJ, 404 ms). The ratios would be close to observed behavior if the model's LC interval were 40% higher. TYPIST does much better, including different strategies for breaking LC and LJ words down, with an average error of 18% for one plausible combination of strategies across different typing speeds (which means that our results, limited to 60 wpm, are not directly comparable).

Statistic	Target	ACT-R
LJ-EP increase	0.677	0.748 (10.5%)
LC-EP increase	0.416	0.064 (84.6%)
LJ-LC increase	0.187	0.643 (243.7%)

Phenomenon 6. Typing rate is slower with restricted preview. Salthouse (1984) presented typists with a sentence from 60 to 83 characters on a single line. Only a preview of *n* characters for the entire sentence was displayed, with each keystroke causing the preview to advance by sliding the text leftward, removing the first character and adding a new one at the end. Preview sizes were 19, 11, 9, 7, 5, 3, and 1 character. The sentences used are not given by Salthouse; our testing substitutes the elevator sentence. We implemented a preview window that acts approximately the same way, except for the leftward movement of the text. Existing text remains on the screen as well but visual processing is strictly left to right. On

each keystroke, new text may be available to the model; the details of incremental visual processing follow that of TYPIST (John, 1996, p. 335), providing words, syllables, or characters, depending on available space given the size of the preview. TYPIST predicts performance on the elevator sentence (85 characters) at 120 gross wpm, with an error of 15.8%.

Performance of the ACT-R model on this sentence at 60 wpm, in terms of the mean interkeystroke interval in ms, is shown in the table below and in Figure 3, compared with the median interkeystroke interval in the experiment cited above (Salthouse, 1984, Table 2). Excluding the non-preview data, ACT-R model predicts Salthouse's observed data with a mean absolute error of 23.8% and $R^2 = 0.992$.

Preview	Target	ACT-R
None	181	207 (14.2%)
19	179	206 (15.2%)
11	183	214 (17.3%)
9	180	232 (28.0%)
7	185	243 (31.5%)
5	205	288 (40.7%)
3	293	381 (30.2%)
1	645	723 (12.1%)

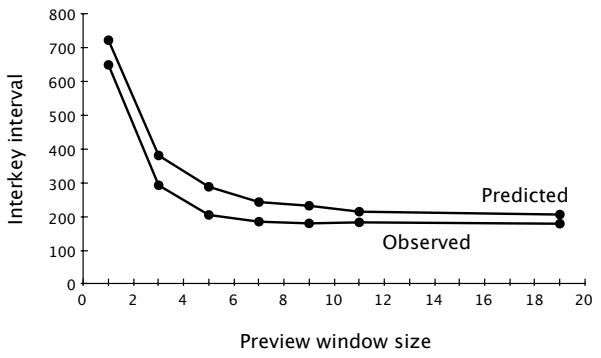


Figure 3: Phenomenon 6: Typing with a preview window

Phenomenon 7. Alternate-hand keystrokes are faster than same-hand keystrokes. Salthouse reports that successive same-hand keystrokes are slower than alternate-hand keystrokes, in the range of 30–60 ms. TYPIST predicts a 50 ms difference, for an error of 11.1% compared with the 45 ms midpoint, by including an additional cognitive operator for same-hand key sequences.

In ACT-R, alternate-hand keystrokes would ordinarily be *slower* than same-hand keystrokes due to feature preparation time. Adopting the solution in TYPIST would require the mapping between keys and hands to be explicit in the model's productions, significantly increasing its complexity. Instead, the function for computing preparation time in the ACT-R motor module was modified to produce an appropriate difference in the opposite direction. On random sentences, the ACT-R model predicts a duration of 47.5 ms, error 5.6%.

Phenomenon 8. More frequent letter pairs are typed more quickly. This is beyond the scope of TYPIST; John observes that it is a small effect, 4% of the variability after same- versus alternate-hand sequences. For the ACT-R model, a test of random sentences shows a near-zero correlation between the relative frequency of a bigram and the model's predicted duration, for the 265 most common English bigrams. These most common bigrams are typed approximately 2.5% faster than the remaining bigrams, because they are more frequently chunked as part of the syllables recognized by the model.

Phenomenon 9. Interkey intervals are independent of word length. Specifically, the duration of the first keystroke in a word is not dependent on its length, and neither are the durations of other keystrokes in the word. Both TYPIST and the ACT-R model show such independence.

Phenomenon 10. The first keystroke in a word is slower than subsequent keystrokes. Salthouse reports a 20% increase for an average typist; TYPIST predicts increases of 0% to 8.4% over typing speeds from 60 wpm to 120 wpm, when the spelling operator is on the critical path. On the elevator sentence, the ACT-R model predicts an increase of 18.5%, for a 7.7% error. On samples of random sentences, the ACT-R model predicts an increase of 18%, for an 11.4% error.

Phenomenon 11. The time for a keystroke is dependent on the specific context. Here, context means that in a sequence of keystrokes, α-β, the duration of β depends on α. TYPIST can predict key context effects with two refinements, based on John's analysis of a set of digraphs e-e, d-e, c-e, r-e, t-e, f-e, g-e, v-e, and b-e (Rumelhart & Norman, 1982). Both refinements decompose a keystroke into horizontal movement of a finger across the keyboard (*move*), key down (β_{down}), then key up (β_{up}). The first is that in the digraph α-β, $\alpha_{up} = 83$ ms for a same-hand, same-finger sequence, 50 ms for a same-hand, different-finger sequence. The second is a relaxation of the MHP assumption of strictly sequential finger movements; John derives a formula for very accurate prediction of the *move* component in these digraphs. Specifically, when the middle finger moves to press a key, the index finger moves part of the distance in the same direction, and vice versa.

TYPIST predicts performance on the digraphs with 0.9% mean absolute error (MAE) and $R^2 = 0.95$. The predictions of the ACT-R model, which does not incorporate John's analysis, have a MAE of 49.8%, $R^2 = 0.211, p = 0.21$.

Digraph	Target	TYPIST	ACT-R
e-e	165	166 (0.0%)	231 (40.0%)
d-e	201	200 (0.5%)	289 (43.8%)
c-e	215	226 (1.8%)	321 (49.3%)
r-e	145	143 (1.4%)	260 (79.3%)
t-e	159	157 (1.2%)	249 (56.6%)
f-e	168	167 (0.6%)	289 (72.0%)
g-e	178	175 (0.1%)	264 (48.3%)
v-e	178	182 (0.9%)	260 (46.1%)
b-e	195	187 (1.4%)	220 (12.8%)

To see why this is a challenge, note the duration predictions for the v-e and b-e key digraphs, as an example: the predicted order is the reverse of the observed. The partial traces below show why. The production initiate-letter fires at the same times for both digraphs (as marked by +), once preparation for the previous keystroke is complete. The v keystroke has a lower execution time than the b (as marked by /) because it is closer to the home key location of the first finger. The earlier completion of the v keystroke has no effect on the initiation of the next, however; instead it produces a longer interval between keys in the v-e digraph than in the b-e digraph, although from initiation to keypress the e keystroke is identical.

```
+ 2.085   PROCEDURAL         FIRED INITIATE-LETTER
  2.085   PROCEDURAL         CLEAR-BUFFER MANUAL
  2.085   PROCEDURAL         CONFLICT-RESOLUTION
/ 2.164   MOTOR              OUTPUT-KEY #(2.75 1.5)
  2.164   [TYPING-WINDOW]    KEY V (289)
  2.164   PROCEDURAL         CONFLICT-RESOLUTION
  2.214   PROCEDURAL         CONFLICT-RESOLUTION
  2.295   PROCEDURAL         CONFLICT-RESOLUTION
+ 2.345   PROCEDURAL         FIRED INITIATE-LETTER
  2.345   PROCEDURAL         CLEAR-BUFFER MANUAL
  2.345   PROCEDURAL         CONFLICT-RESOLUTION
/ 2.424   MOTOR              OUTPUT-KEY #(1.5 0.0)
  2.424   [TYPING-WINDOW]    KEY E (260)

+ 2.085   PROCEDURAL         FIRED INITIATE-LETTER
  2.085   PROCEDURAL         CLEAR-BUFFER MANUAL
  2.085   PROCEDURAL         CONFLICT-RESOLUTION
/ 2.204   MOTOR              OUTPUT-KEY #(3.5 1.5)
  2.204   [TYPING-WINDOW]    KEY B (329)
  2.204   PROCEDURAL         CONFLICT-RESOLUTION
  2.254   PROCEDURAL         CONFLICT-RESOLUTION
  2.295   PROCEDURAL         CONFLICT-RESOLUTION
+ 2.345   PROCEDURAL         FIRED INITIATE-LETTER
  2.345   PROCEDURAL         CLEAR-BUFFER MANUAL
  2.345   PROCEDURAL         CONFLICT-RESOLUTION
/ 2.424   MOTOR              OUTPUT-KEY #(1.5 0.0)
  2.424   [TYPING-WINDOW]    KEY E (220)
```

It is possible to integrate John's analysis into the ACT-R model, even if Rumelhart and Norman's data are for a typist about 25% faster than the ACT-R model. We developed new motor code, modifying execution and finish time computations to match John's analysis. Manual requests were triggered on "state free" rather than "preparation free", and preparation time was zeroed out. These changes allow the model to reproduce TYPIST's performance almost exactly, but they are an awkward fit for ACT-R.

Recall that for a given digraph α-β, the duration of the α_{up} component depends on β. The duration of a keystroke is computed when it is initiated, but when α is initiated, β is not yet available to the motor module. The modified motor code requires manual requests to be made as pairs of keystrokes, current and future, based on the one-key lookahead (in the retrieval buffer) used to handle the end of syllables—an ad hoc solution without theoretical justification. It further proved difficult to accommodate Phenomenon 7 (alternate-hand timing) as a motor phenomenon in the changed code. Incorporating key context into the design of the existing model introduces complexity that we leave for future work.

Phenomenon 12. A concurrent task does not affect typing. Salthouse and Saults (1987) describe an experiment in which typists were asked to type while performing a simultaneous auditory reaction time task. While typing, when the typists heard an auditory cue, they were to press a foot pedal. For this phenomenon, a simple foot pedal motor extension was added to ACT-R. A *pedal* buffer in the motor module might be appropriate, but this was not implemented; instead *press-pedal* requests are interleaved with the hand movements.

Typing performance degraded only slightly for participants. TYPIST and ACT-R were evaluated on the elevator sentence, with an auditory cue beginning at 25 different random locations.

Statistic	Target	TYPIST	ACT-R
Single (ms)	181	195 (7.7%)	210 (16.2%)
Concurrent (ms)	185	196 (5.9%)	212 (14.6%)
Pedal (ms)	431	435 (0.9%)	445 (3.2%)

Phenomenon 13. Copy span is 7–40 characters. The copy span is the number of characters which a typist can continue typing after a single inspection of the material, without its being visible during typing. Salthouse (1986) describes an experiment in which the display was erased after predetermined number of keystrokes by participants, after which they continued typing as much as they remembered. TYPIST and the ACT-R model were evaluated on an equivalent task, in which the elevator sentence was typed and the copy span was determined after each character.

Statistic	Target	TYPIST	ACT-R
Copy span (ms)	14.6	12.5 (14.4%)	15.2 (4.3%)

Phenomenon 14. Stopping span is between one and two characters. The stopping span is the number of characters to which a typist commits to after a signal to stop typing. Logan (1982) describes an experiment in which participants were asked to type single words of 3, 5, or 7 letters; after a predetermined amount of time (500, 650, 800, or 950 ms), an auditory cue was given. TYPIST And ACT-R were evaluated using the same time values, on words that covered all combinations of same- and alternate-hand keystrokes. The results comparing Logan's, TYPIST, and ACT-R model are as follows:

Statistic	Target	TYPIST	ACT-R
Stopping span (char)	1.57	1.76 (12.1%)	2.11 (34.2%)

Overall, the model improves on results obtainable by text entry in CogTool, which types the elevator sentence at approximately 50 wpm. CogTool does not model the differences between words and random letter order, producing mean keystroke intervals for EP, LC, and LJ that differ by at most 5 ms (Phenomenon 5). Alternate-hand keystrokes are 39.5 ms faster than single-hand keystrokes (Phenomenon 7, 12% error). The first keystroke in a word is 17.5% slower than the remaining keystrokes (Phenomenon 10, 12.5% error). CogTool does not model key context (Phenomenon 11).

Discussion

The ACT-R model gives predictions of human performance on basic typing phenomena that are at least qualitatively correct, except for context dependence in keystroke duration and the stopping span being outside an observed boundary.

Motor processing in ACT-R is based on that of EPIC (Kieras & Meyer, 1997), in which the duration of a keystroke depends on several factors: preparation time, motor initiation time, and a minimum burst time, as well as Fitts' Law movement time. By design, the duration of preparation for a movement increases with the number of features that differ from the previous movement (different hands, fingers, direction and distance of movement). Using the default ACT-R keyboard, the press-key movement style, default parameter settings and no motor changes, the model's performance is roughly similar to that described in the previous section. Leaving aside the different typing speed, the differences are on alternate hands (Phenomenon 7, 150% error), first keystroke duration (Phenomenon 10, 161% error), and stopping span (Phenomenon 14, 5.3% error).

We have only lightly explored the space of ACT-R motor parameters, settling on modifications to preparation time as the simplest way to bring modeling results in line with human performance. For the typing model, the feature preparation computations are modified such that preparation of each keystroke has a default duration (50 ms) plus an extra increment when the previous keystroke was with the same hand. Our repurposing of feature preparation for typing is not theoretically well-motivated, in part because theory is sparse. In an updated analysis of the motor literature and EPIC, Kieras (2009) eliminates the dependence of visually aimed manual and ocular movements on feature preparation. He further asks, "Should feature preparation be discarded for keypress movements as well?" For typing the answer appears to be yes, where feature preparation is replaced by functionality that approximates the timing of overlapping, interdependent physical movements with the sequential movements required by the architecture.

Despite the its limitations, we believe this work is important for a few reasons. First is the pragmatic accomplishment of extending ACT-R to a very common task; some human experiments that involve typing as a primary or secondary task can now be taken on. Our work provides evidence for the soundness of TYPIST's design in a symbolic architecture. Second, the performance limitations of the model suggest new directions for research on the architecture, with well-defined tasks and clear empirical targets. Third, the MHP representation makes TYPIST performance easier to analyze than that of the ACT-R model, but we also find value in running trials over large sets of sentences and analyzing aggregate data. Finally, the model raises questions dealing with strategies, visual processing, and how typists learn to adjust their reading speed to working memory limitations, which we will examine in future work.

Acknowledgments

This work was funded by grants from the Army Research Office [W911NF-08-1-0105] and the National Science Foundation [IIS-1451172]. Thanks to Dan Bothell, Bonnie John, Dave Kieras, Don Morrison, and Frank Ritter for code, advice, and suggestions.

References

Card, S. K., Moran, T. P., & Newell, A. (1983). *The psychology of human-computer interaction*. Hillsdale, NJ: Lawrence Erlbaum Associates.

John, B. E. (1996). TYPIST: A theory of performance in skilled typing. *Human-Computer Interaction*, *11*(4), 321–355.

John, B. E., Prevas, K., Salvucci, D. D., & Koedinger, K. (2004). Predictive human performance modeling made easy. In *Proceedings of the SIGCHI conference on human factors in computing systems* (pp. 455–462).

Kieras, D. E. (2009). Why EPIC was wrong about motor feature programming. In *Proceedings of the international conference on cognitive modeling* (pp. 68–73).

Kieras, D. E., & Meyer, D. E. (1997). An overview of the EPIC architecture for cognition and performance with application to human-computer interaction. *Human-computer interaction*, *12*(4), 391–438.

Logan, G. D. (1982). On the ability to inhibit complex movements: A stop-signal study of typewriting. *Journal of Experimental Psychology: Human Perception and Performance*, *8*(6), 778.

Rumelhart, D. E., & Norman, D. A. (1982). Simulating a skilled typist: A study of skilled cognitive-motor performance. *Cognitive Science*, *6*(1), 1–36.

Salthouse, T. A. (1984). Effects of age and skill in typing. *Journal of Experimental Psychology: General*, *113*(3), 345.

Salthouse, T. A. (1986). Perceptual, cognitive, and motoric aspects of transcription typing. *Psychological bulletin*, *99*(3), 303.

Salthouse, T. A., & Saults, J. S. (1987). Multiple spans in transcription typing. *Journal of Applied Psychology*, *72*(2), 187.

Snyder, K. M., & Logan, G. D. (2014). The problem of serial order in skilled typing. *Journal of Experimental Psychology: Human Perception and Performance*, *40*(4), 1697–1717.

West, L. J., & Sabban, Y. (1982). Hierarchy of stroking habits at the typewriter. *Journal of Applied Psychology*, *67*(3), 370.

Wu, C., & Liu, Y. (2004). Modeling human transcription typing with queuing network-model human processor (QN-MHP). In *Proceedings of the human factors and ergonomics society annual meeting* (Vol. 48, pp. 381–385).

A Predictive Model of Human Error based on User Interface Development Models and a Cognitive Architecture

Marc Halbrügge (marc.halbruegge@tu-berlin.de)
Quality and Usability Lab, Telekom Innovation Laboratories
Technische Universität Berlin, Ernst-Reuter-Platz 7, 10587 Berlin

Michael Quade (michael.quade@dai-labor.de)
DAI-Labor, Technische Universität Berlin
Ernst-Reuter-Platz 7, 10587 Berlin

Klaus-Peter Engelbrecht (klaus-peter.engelbrecht@telekom.de)
Quality and Usability Lab, Telekom Innovation Laboratories
Technische Universität Berlin, Ernst-Reuter-Platz 7, 10587 Berlin

Abstract

The concept of device- vs. task-orientation allows to identify subtasks that are especially prone to errors. Device-oriented tasks occur whenever a user interface requires additional steps that do not directly contribute to the users' goals. They comprise, but are not limited to, initialization errors and post-completion errors (e.g., removing a bank card after having received money). The vulnerability of device-oriented tasks is often counteracted by making them obligatory (e.g., by not handing out the money before the bank card has been removed), making it even harder to predict where users will have problems with a given interface without dedicated user tests. In this paper we show how cognitive modeling can be used to predict error rates of device-oriented and task-oriented subtasks with respect to a given application logic. The process is facilitated by exploiting user interface meta information from model-based user interface development.

Keywords: Human Error; Cognitive Modeling; Model-based User Interface Development; Memory for Goals; ACT-R

Introduction

Our everyday life is dominated by routine activities, i.e., well-learned, rule-based tasks like making coffee or buying a train ticket. And even though we have performed them hundreds of times, we are still making occasional errors during their execution (Reason, 1990). While the base error rate for routine tasks is low, some procedural steps are more problematic than others. The best-known examples are post-completion errors, when users fail to perform an additional step in a procedure *after* they have already reached their main goal (Byrne & Davis, 2006; Trafton, Altmann, & Ratwani, 2011; Ratwani & Trafton, 2011). This concept can be expanded to any step that doesn't directly contribute to the users' goals, but is imposed on them by the system and has been coined *device-orientation* (Ament, Blandford, & Cox, 2009; Gray, 2000). The opposite, i.e., steps that are noticeably related to the users' goals, is called *task-orientation* in contrast.

Jef Raskin popularized the complaint that "a dialog box that has no choices (e.g., you can only press ENTER before you can do any other task) has a productivity of 0" (Raskin, 1997), because the user cannot transfer any knowledge to the system using it. Raskin's information theoretic concept fits nicely with our understanding of device-oriented tasks: they do not convey any information that is specific to the user's current goal.

Even when human interface designers try to avoid device-oriented tasks in the first place, software or hardware constraints do not allow the removal of every single one. As long as bank cards are used, they need to be placed into some kind of slot and removed later on. One popular design strategy in this situation is to make device-oriented tasks *obligatory*, i.e., their position in the action sequence is pulled forward so that the user's goal cannot be achieved without performing the device-oriented task. But how does this change affect error rates?

The objective of this paper is to shed some light on the impact of device-orientation and subtask necessity[1] on user errors. We derive a computational user model from the memory for goals theory (MFG, Altmann & Trafton, 2002) and apply it to a class of user interfaces (UI) that have been created using the model-based UI development process (MBUID, Vanderdonckt, 2008; Calvary et al., 2003). As a by-product of this process, such interfaces provide meta-information about their elements (e.g., buttons) that can be used for the creation of more generic cognitive models (Quade, Halbrügge, Engelbrecht, Albayrak, & Möller, 2014).

Action Control and Human Error

According to Rasmussen (1983), human action control can be described on three levels: skill-, rule-, and knowledge-based behavior. Skill-based behavior on Rasmussen's lowest level is generated from highly automated sensory-motor actions without conscious control. Knowledge-based behavior on the other hand is characterized by explicit planning and problem solving in unknown environments. In between the skill and the knowledge levels is rule-based behavior. While being goal-directed, rule-based actions do not need explicit planning or conscious processing of the current goal. The stream of actions follows stored procedures and rules that have been developed during earlier encounters or through instruction.

[1] With respect to user tasks, we are using the terms *obligatory*, *mandatory* and *necessary* synonymously in the course of this paper.

Interacting with computer systems after having received some training is predominantly located on Rasmussen's rule-based level, with contributions of the skills level to a lesser degree. We are therefore concentrating our modeling effort on rule-based behavior. On this this level of action control, human error is characterized by deviation from the stored procedure, i.e., either leaving out a step or adding an additional unnecessary one.

Model Based User Interface Development (MBUID)

The main goals of model-based user interface development are the reduction of complexity and an increased reusability of existing patterns and solutions during UI development (Vanderdonckt, 2008). For this purpose, there exist frameworks and description languages that conform to the CAMELEON reference framework (Calvary et al., 2003) for structuring the processes and information of MBUID. While *abstract UI* (AUI) models describe UI elements on a modality-independent level, the more *concrete UI* models hold modality-specific information and the *final UI* models represent specific implementations for different platforms. Besides these models there exist conceptual models related to the domain of the application and the task model for describing the interaction logic.

A widely used approach for modeling tasks in MBUID is the Concurrent Task Tree (CTT) notation (Paternò, 1999) which also allows executing tasks in conjunction with UI models (Mori, Paternò, & Santoro, 2004) by establishing mappings between these models. CTT comes with several operators for grouping tasks into hierarchical structures, defining temporal relationships and for describing information flow between user, application and tasks. Namely *interaction tasks* describe tasks that user and application perform together, while *application tasks* relate to actions by the system.

By combining these different UI development models, information about the application becomes available that goes far beyond what is visible to the user. In the context of this paper, the task information contained within the UI development models is especially interesting, as it corresponds directly to Rasmussen's rule-based behavior level.

Experiment

The scarcity (yet pervasiveness) of procedural errors during routine tasks (Reason, 1990) makes statistical analysis difficult. Researchers have used secondary tasks (e.g., Byrne & Davis, 2006; Ruh, Cooper, & Mareschal, 2010) or interruptions (e.g., Li, Blandford, Cairns, & Young, 2008; Altmann, Trafton, & Hambrick, 2014) to increase error rates. We rejected both options for reasons of ecological validity, and chose repeated measures and a medium sized sample instead.

The experiment focuses on procedural errors during the usage of a kitchen assistance system for ambient assisted living. The kitchen assistant helps preparing a meal for a given number of persons with its searchable recipe library, adapted shopping list generator, and by providing interactive cooking or baking instructions.

Ambient systems like the one used here are characterized by interconnecting a multitude of physical devices. We selected among these a personal computer with large touch screen and a tablet computer for the experiment. To match the characteristics of each device, we created two versions of the user interface of the kitchen assistant. The first version is a tablet-oriented, *simple* one, optimized for portrait mode, with larger buttons and fonts, and rather few elements per screen. The other UI version is more *complex*, using smaller buttons and fonts so that more elements fit on the screen. Some screens of the simple UI are shown side-by-side in the complex version, thereby reducing the necessity of navigation between screens. The complex UI targets the large personal computer and is optimized for landscape mode. Annotated screenshots of the two UI versions are shown in Figure 1.

Analysis of the UI Development Models

Because we are working with an already existing application, we cannot freely manipulate whether a UI element and its related subtask is device- or task-oriented. But we can use the MBUID models to derive this property from them in a generalizable way. By analyzing the operators of the task model and the types of AUI elements it is possible to identify device- and task-oriented subtasks and make these explicit to the evaluator or modeler.

On the one hand, task-oriented interaction steps are characterized by interaction on UI elements that are modeled on the level of the AUI model as *FreeInput* or *Choice*. These elements are generally used to provide task-oriented information from the user to the application using *interaction tasks*, e.g., by using text fields, radio buttons or checkboxes. On the other hand, device-oriented interaction steps describe actions which let the dialogue proceed to a further step, e.g., when executing buttons labeled "Next" or "Done". The abstract type *Command* denotes such UI elements on the level of the AUI model which can be used to trigger application tasks.

In order to identify device-oriented subtasks, the models have to be checked for *interaction tasks* that are modeled using *Command* elements which then *enable* application tasks. The process is visualized in Figure 1. In the case of the kitchen assistant, device-oriented elements are buttons that lead to the next screen or modify entries from the ingredients list. Task-oriented UI elements are buttons for toggling search attributes and selections in search results lists.

Method

Participants Twenty members of the Technische Universität Berlin paid participant pool took part in the experiment. There were 5 men and 15 women, with an age range from 18 to 59 (M=32.3, SD=11.9). As the instructions were given in German, only fluent German speakers were allowed to take part.

Materials The experiment was conducted in our fully equipped lab kitchen. A personal computer with 27" (68.6 cm, landscape mode) touch screen and a 10" (25.7 cm, por-

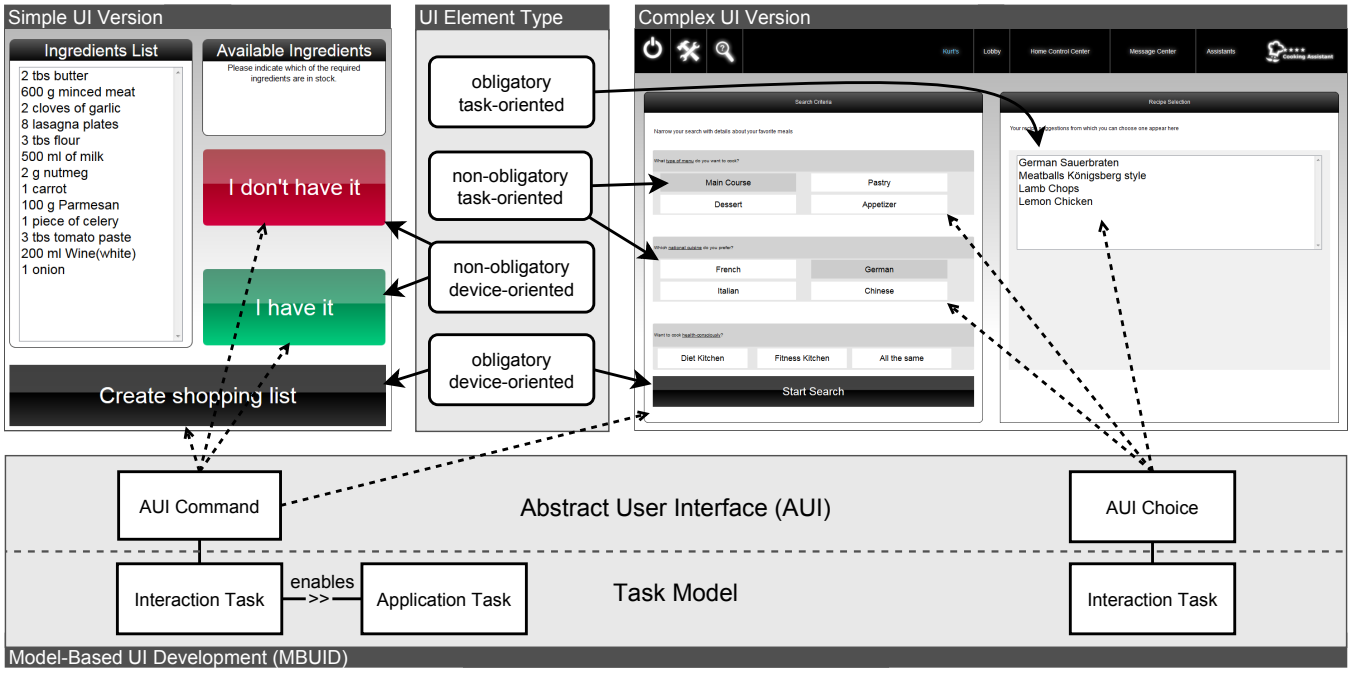


Figure 1: Screenshots of the kitchen assistant. Ingredients list of the simple UI on the left, recipe search of the complex UI with two screens side-by-side on the right. UI element type is indicated by solid arrows, AUI type indicated by dashed arrows.

trait mode) tablet, both placed near the sink, were used to display the interface of the kitchen assistant. All user actions were recorded by the computer system. The subjects' performance was additionally recorded on videotape for subsequent error identification.

Design We applied a four-factor within-subjects design, the factors being UI version (simple vs. complex), physical device (screen vs. tablet), device- vs. task-orientation, and task necessity (non-obligatory vs. oligatory). Every subject went through all four combinations of UI version and physical device in randomized order. User tasks were analogously grouped into four blocks of eleven to twelve individual tasks. Block orders were counterbalanced across participants as well. We call the combination of the device-orientation and task necessity factors *UI element type* in the following. While device-orientation can be derived from the MBUID models (see above), task necessity is only implicitly represented in the assistant's interaction logic. Mandatory subtasks are related to all elements that lead to the next screen or unhide buttons on the current screen, the latter being the case for the selection of a recipe in the search result list (see Figure 1).

Procedure Every block started with comparatively easy recipe search tasks, e.g., "search for German main dishes and select lamb chops".[2] Users would then have to change the search attributes, e.g., "change the dish from appetizer to dessert and select baked apples". The second half of each block was made of more complex tasks that were spread over more screens of the interface and/or needed memorizing more items. The subjects either had to create ingredients lists for a given number of servings, or had to make shopping lists using a subset of the ingredients list, e.g., without salt and flour. All instructions were read to the subjects by the experimenter. Every individual trial was closed by a simple question the subjects had to answer to keep them focused on the kitchen setting, e.g., "how long does the preparation take?" With an initial training phase and exit questions the whole procedure took less than an hour.

Results

We identified errors by comparing the observed click sequences with optimal ones. Whenever a step of the optimal sequence was missing, we recorded this as an *omission*. Unnecessary additional steps performed by the subjects were analogously recorded as *intrusions*. A special case of intrusions are *perseverations*, when an action is erroneously repeated. The resulting taxonomy follows Ruh et al. (2010) and is deliberately using phenotypic descriptions instead of genotypic explanations (in contrast to, e.g., Norman, 1988).

In total, 6359 clicks were observed. 104 (1.6%) of these were classified as user errors. 56 were omissions, 38 were intrusions, and 10 were perseveration errors. The perseveration errors were bound to a button for increasing the number of servings that needed several consecutive presses. As the video recordings clearly showed that all perseverations were not caused by the users, but by occasional excesses of UI lag, they were removed from further analysis.

[2]We give English translations of the actual instructions here for reasons of comprehensibility. The original instructions are available for download at http://www.tu-berlin.de/?id=135088

Due to the scarcity of errors and the use of repeated measures, χ^2-based significance tests could not be used (Agresti, 2014). The error rates were instead evaluated using a mixed logit model with subject as random effect (Bates, Maechler, Bolker, & Walker, 2014). They do not vary with device, UI version, or task necessity (all $p > .3$). The main effect of device-orientation points to the expected direction with device-oriented subtasks showing higher error rates, but narrowly missed significance ($z = -1.82, p = .069$). We found a significant interaction between necessity and device-orientation ($z = 4.07, p < .001$). Error rates and 95%-confidence intervals are given in Figure 2.

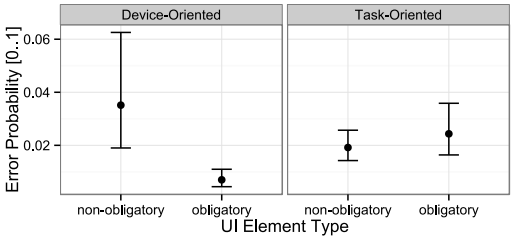


Figure 2: Error probabilities for different UI elements.

Cognitive User Model

The theoretical foundation for our model is the memory for goals theory (MFG, Altmann & Trafton, 2002). The MFG postulates that goals and subgoals are subject to the characteristics of human memory traces, in particular time-dependent and noisy *activation*, *interference*, and associative *priming*. Lack of activation of a subgoal, possibly connected to little priming, can cause omissions, while interference with other subgoals can result in intrusions. While the MFG theory initially has been validated on the basis of Tower-of-Hanoi experiments, i.e., rather artificial problem-solving tasks in the laboratory, it has been shown to generalize well to sequence errors during software use and has been extensively used in the human-computer interaction domain (e.g., Li et al., 2008; Trafton et al., 2011).

The cognitive user model presented here has been developed using the cognitive architecture ACT-R (Anderson et al., 2004). As shown in previous research, associative priming can be considered an acceptable explanation of *temporal* disadvantages of device-oriented steps (Halbrügge & Engelbrecht, 2014). The work presented here provides a direct link from device-orientation to user errors.

This was achieved by using two additional concepts: firstly, *partial matching* mimics human memory imperfections by responding to memory retrievals with similar, but not completely fitting chunks of information. And secondly, humans as embodied and situated beings tend to use environmental cues to reduce cognitive load (Wilson, 2002). This led to the addition of a *knowledge-in-the-world* (Norman, 1988) strategy where the user scans the UI for "inviting" elements instead of relying on the internal representation of the current task, only. This attempt also goes in line with Salvucci's (2010) criticism of the MFG theory being too focused on memory while neglecting the user's interaction with the environment.

Technically, the model is run inside the standard Lisp distribution of ACT-R 6.0.[3] The ability to interact with the HTML interface of the kitchen assistant is provided by ACT-CV (Halbrügge, 2013). The model receives its instructions through ACT-R's auditory system and tries to memorize the necessary steps for the current trial. No specific knowledge about the kitchen assistant is hard-coded into the model. When pursuing a goal, the model first uses a knowledge-in-the-head strategy, i.e., it tries to follow the memorized step sequence (left part of Figure 3). Once memory gets weak, the knowledge-in-the-world strategy takes over. Elements on the screen are randomly attended and a memory recall heuristic is used to determine whether this element was part of the current action sequence. If no matching goal chunk is found, the visual search for possible targets is continued (right part of Figure 3).

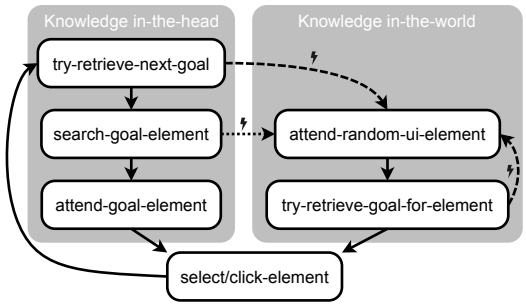


Figure 3: Schematic flow chart of the knowledge-in-the-head and knowledge-in-the-world strategies of the cognitive model. Dashed arrows denote retrieval failure, dotted arrows denote visual search failure.

How does the Model Predict Errors?

Memory activation (and its noise) is the main explanatory construct used by the model. Omissions are caused by the activation of the respective subgoal being too low. As activation decays over time, omissions are more likely for longer task sequences and for subgoals that appear late in a sequence. Task-oriented elements of a sequence receive additional activation through priming (Halbrügge & Engelbrecht, 2014) and are therefore in principle less prone to omissions. This effect can be mitigated by the application logic, though. If a subgoal can no longer be retrieved by the knowledge-in-the-head strategy, its corresponding UI element can nevertheless be found by the knowledge-in-the-world strategy (see Figure 3). This is especially probable for mandatory subtasks like navigating to the following screen because these mark

[3]The Lisp source code of the model is available for download at http://www.tu-berlin.de/?id=135088

situations where no other applicable UI element can be found by the model.

Intrusions happen when the activation of a similar subgoal of a previous trial exceeds the activation of the current subgoal. The partial matching mechanism adds an additional penalty to the intruding subgoal's activation depending on its dissimilarity to the retrieval request.[4] Activation noise is essential for intrusions, but they can also be caused by an old subgoal receiving "misguided" priming from the current goal, e.g., when there is substantial overlap between two consecutive trials. Because task-oriented subgoals receive more priming than device-oriented ones, the model predicts higher intrusion rates for task-oriented UI elements.

Another cause for intrusions is the knowledge-in-the-world strategy. ACT-R's activation spreading mechanism allows priming from the current focus of the visual system to declarative memory elements, comparable to the horizontal dimension of the "Dual Systems" theory (Cooper & Shallice, 2000). When the model searches the screen for "inviting" elements (*attend-random-ui-element* in Figure 3), the currently attended element primes subgoals that correspond to it regardless of whether they belong to the current goal or a previous one.

Model Fit

The model predictions are sensitive to several global ACT-R parameters that affect activation. We kept activation decay (bll) at the default of .5, varied activation noise (ans) between .5 and .6, set priming (mas) to 3.5 and varied the partial matching penalty (mp) between 4.0 and 4.5.

The model was run 1000 times and all errors made were collected (dotted lines in Figure 4). The quantitative fit as computed based on the error probability for each combination of omissions and intrusions and the four UI element types defined above is promising with R^2 = .915 and RMSE = .003.

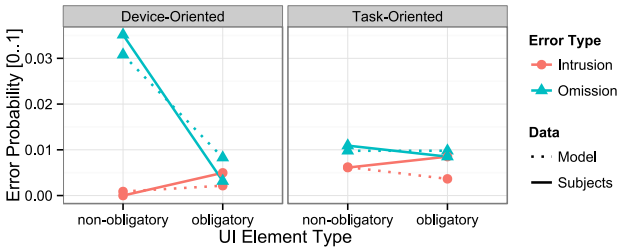


Figure 4: Empirical error probabilities and model predictions for different UI elements.

Discussion

We conducted a usability study to examine the effects of device-orientation and task necessity on procedural errors. Unsurprisingly, mandatory tasks were less likely to be omitted. They showed slightly higher intrusion rates than their non-obligatory counterparts, though. Screen elements that were both device-oriented and non-obligatory showed by far the highest error, i.e., omission rates.

These results are valuable in themselves for several reasons. First, they show that human error can be studied well without adding secondary tasks or interrupting the subjects during their tasks. Secondly, the concept of device- vs. task-orientation has proven beneficial in principle. And finally, our results highlight how little we can predict when we base our predictions on nothing but theoretical concepts (here device-orientation). Only when we take the application's interaction logic into account (here focused on obligatory vs. non-obligatory task steps) we get significant differences that are worth further analysis.

At first thought, this might lead into a problem: The application logic is specific to a single application most of the time. If we base our analysis on it, we are in danger of limiting the scope of our research to nothing but that single application. The choice of cognitive modeling as a methodology helps resolving this issue. Cognitive models can integrate other knowledge as long as it is machine readable. In our case, the meta-information encoded within the MBUID user interface models provides the possibility to model the activation gains of task-oriented subgoals in a generalizable way. And even more important, the influencing factor of task necessity is only implicitly represented within the application. The interaction effect of device-orientation and task necessity is an emergent phenomenon that is only elicited during the execution of the cognitive model, i.e., by performing tasks using the actual application.

We created a cognitive model based on the memory for goals theory, integrated the MBUID information, and added a knowledge-in-the-world strategy that the model applies after facing memory retrieval failure. While the overall goodness-of-fit of the model is good and especially the case of non-obligatory device-oriented tasks showing the highest error rate is reproduced well (see Figure 4), there are several limitations. The sheer number of mechanisms used (decay, priming, etc.) leads to a complex cognitive model that is rather sensitive to changes of the respective ACT-R parameters. Future research will show whether this affects the generalizability of the model. The same holds for the empirical basis of the model and the representativity of the tasks used during the experiment.

The model nevertheless provides several improvements when compared to existing MFG models of procedural error. Most important, while Altmann and Trafton (2002) discuss how the environment can provide cues that prime pending goals, our model is the first one that actively uses this strategy. As a by-product, this leads to the prediction of intrusions, an important error class that is often not well captured (e.g., Trafton et al., 2011; Li et al., 2008).

[4]The dissimilarities are computed by ACT-R based on the number of mismatching information units, here trial and current subgoal. No user-specified similarity function is used by this model.

Concluding Remarks

As of today, the body of theory and empirical research on human error is growing, but validated methods for predicting user errors exist only for restricted areas (e.g., Ratwani & Trafton, 2011). We present a computational user model grounded in cognitive science research that aims at more general error predictions. Taking advantage of UI meta information collected during model-based development of different user interfaces, our cognitive user model can well reproduce the findings from a usability study conducted before.

In the future, we plan to apply the model to new and different interfaces and develop the connection to MBUID towards higher automation. This way, interface designers could receive error predictions at early stages of the development cycle, making error prevention much easier.

Acknowledgements

We gratefully acknowledge financial support from the German Research Foundation (DFG) for the project "Automatische Usability-Evaluierung modellbasierter Interaktionssysteme für Ambient Assisted Living" (AL-561/13-1).

References

Agresti, A. (2014). *Categorical data analysis*. John Wiley & Sons.

Altmann, E. M., & Trafton, J. G. (2002). Memory for goals: An activation-based model. *Cognitive science*, *26*(1), 39–83.

Altmann, E. M., Trafton, J. G., & Hambrick, D. Z. (2014). Momentary interruptions can derail the train of thought. *Journal of Experimental Psychology: General*, *143*(1), 215–226.

Ament, M. G., Blandford, A., & Cox, A. L. (2009). Different cognitive mechanisms account for different types of procedural steps. In *Proc. COGSCI '09* (pp. 2170–2175).

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological review*, *111*(4), 1036–1060.

Bates, D., Maechler, M., Bolker, B., & Walker, S. (2014). lme4: Linear mixed-effects models using Eigen and S4 [Computer software manual]. (R package version 1.1-7)

Byrne, M. D., & Davis, E. M. (2006). Task structure and postcompletion error in the execution of a routine procedure. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, *48*(4), 627–638.

Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., & Vanderdonckt, J. (2003). A unifying reference framework for multi-target user interfaces. *Interacting with Computers*, *15*(3), 289–308.

Cooper, R., & Shallice, T. (2000). Contention scheduling and the control of routine activities. *Cognitive Neuropsychology*, *17*(4), 297–338.

Gray, W. D. (2000). The nature and processing of errors in interactive behavior. *Cognitive Science*, *24*(2), 205-248.

Halbrügge, M. (2013). ACT-CV: Bridging the gap between cognitive models and the outer world. In E. Brandenburg, L. Doria, A. Gross, T. Günzlera, & H. Smieszek (Eds.), *Grundlagen und anwendungen der mensch-maschine-interaktion* (pp. 205–210). Berlin: Universitätsverlag der TU Berlin.

Halbrügge, M., & Engelbrecht, K.-P. (2014). An activation-based model of execution delays of specific task steps. *Cognitive Processing*, *15*, S107-S110.

Li, S. Y., Blandford, A., Cairns, P., & Young, R. M. (2008). The effect of interruptions on postcompletion and other procedural errors: An account based on the activation-based goal memory model. *Journal of Experimental Psychology: Applied*, *14*(4), 314.

Mori, G., Paternò, F., & Santoro, C. (2004). Design and development of multidevice user interfaces through multiple logical descriptions. *IEEE Trans. Softw. Eng.*, *30*(8), 507–520.

Norman, D. A. (1988). *The psychology of everyday things*. New York, NY: Basic books.

Paternò, F. (1999). *Model-based design and evaluation of interactive applications*. Springer-Verlag.

Quade, M., Halbrügge, M., Engelbrecht, K.-P., Albayrak, S., & Möller, S. (2014). Predicting task execution times by deriving enhanced cognitive models from user interface development models. In *Proceedings of the 2014 ACM SIGCHI symposium on engineering interactive computing systems* (pp. 139–148). New York: ACM.

Raskin, J. (1997). Looking for a humane interface: will computers ever become easy to use? *Communications of the ACM*, *40*(2), 98–101.

Rasmussen, J. (1983). Skills, rules, and knowledge; signals, signs, and symbols, and other distinctions in human performance models. *Systems, Man and Cybernetics, IEEE Transactions on*, *13*, 257–266.

Ratwani, R. M., & Trafton, J. G. (2011). A real-time eye tracking system for predicting and preventing postcompletion errors. *Human–Computer Interaction*, *26*(3), 205–245.

Reason, J. (1990). *Human error*. New York, NY: Cambridge University Press.

Ruh, N., Cooper, R. P., & Mareschal, D. (2010). Action selection in complex routinized sequential behaviors. *Journal of Experimental Psychology: Human Perception and Performance*, *36*(4), 955.

Salvucci, D. D. (2010). On reconstruction of task context after interruption. In *CHI 2010: Proceedings of the SIGCHI conference on human factors in computing systems* (pp. 89–92).

Trafton, J. G., Altmann, E. M., & Ratwani, R. M. (2011). A memory for goals model of sequence errors. *Cognitive Systems Research*, *12*, 134–143.

Vanderdonckt, J. (2008). Model-driven engineering of user interfaces: Promises, successes, failures, and challenges. In *Proc. ROCHI 2008.*

Wilson, M. (2002). Six views of embodied cognition. *Psychonomic bulletin & review*, *9*(4), 625–636.

An Activation-Based Model of Routine Sequence Errors

Laura M. Hiatt (laura.hiatt@nrl.navy.mil)
J. Gregory Trafton (greg.trafton@nrl.navy.mil)
U.S. Naval Research Laboratory
Washington, DC 20375 USA

Abstract

Models of errors during routine sequential action are typically interface-independent. We present here evidence that different task spatial layouts, however, result in different patterns of sequence errors. We explain this data by expanding upon the Memory for Goals framework's activation-based, sequential process to include environmental (such as visual) contextual cues, as well as a richer priming structure. We show a strong qualitative and quantitative fit to experimental data.

Keywords: Priming; routine sequence errors; cognitive modeling.

Introduction

Sequence errors are errors in the order of steps ideally taken to complete a task. Typically, routine sequence errors take the form of either repeating previous steps (perseveration errors), or skipping one or more steps (anticipation errors) (Reason, 1984).

Various accounts exist for sequence errors (Cooper & Shallice, 2006; Botvinick & Plaut, 2006). One successful model is the memory for goals (MFG) model (Trafton, Altmann, & Ratwani, 2011), which uses episodic control codes to direct step-by-step progression through the task. At any point, the most activated code is selected as the current step to work on; this is based on activation strengthening (i.e., frequency and recency of use), as well as activation from priming effects (i.e., associated cues from the current goal).

These current theories of sequence errors are interface-independent; that is, they do not depend on spatial specifics of the task interface. We present here evidence, however, that changes in a task's spatial layout can lead to different patterns of sequence errors. This difference could be explained by spatial reasoning, but there are not currently spatial reasoning theories integrated into sequence error theories. Instead, we explain this data with a model that utilizes the main principles of MFG, including activation and priming; crucially, however, it expands MFG's notion of priming with a richer priming structure, and allows it to capture a fuller environmental context (Hiatt & Trafton, 2013; Thomson, Bennati, & Lebiere, 2014).

In this work, priming can stem from anything in working memory, including visual representations, and can be the result of explicit correspondences between concepts, as well as more implicit relationships, such as co-occurrence. This fuller view on priming allows our model to explain the changes in error patterns stemming from different task interfaces, in large part because our account of priming includes visual cues. We next describe an experiment showing how task layout affects error patterns; then we discuss our model, show that we provide a good qualitative and quantitative account for the data, and end with a discussion of the implications of our approach.

Experiment

Forty-three participants performed a version of Ratwani and Trafton's financial management task (Ratwani & Trafton, 2011). The task is a form-filling task where steps need to be performed in a specific order to buy or sell stocks. The layout primarily consists of two columns; unlike previous versions of the task, where the task step sequence moves down the columns (a columnar layout; e.g., Ratwani & Trafton, 2011; Trafton et al., 2011), here, the step sequence goes across before going down (a horizontal layout; see Figure 1). Each step, save the first and the last, consists of selecting the appropriate widget, and then selecting the appropriate value before hitting a "submit" button. The first step consisted of choosing a stock to trade; the last involved hitting a "Complete Order" button (a post completion step; Byrne & Bovair, 1997). The task had no place-keeping, so upon completing a step, there were no cues about the correct step to take next.

Participants performed three training trials, followed by 20 testing trials. Occasionally, after completing a step, the screen cleared and the participants were interrupted to perform a simple arithmetic task; the interruption lasted 15 seconds. After the interruption, the participants were expected to resume the task and continue with the next appropriate step. For five of the testing trials, there were three interruptions; another five trials had two interruptions each; five trials had one interruption each; and five trials had zero interruptions. The trial order was randomly determined outside the knowledge of the participants to keep participants from guessing whether a step would be followed by an interruption.

Occasionally, participants made an error by selecting the wrong step to work on next. The highest percentage of the errors by participants occurred when resuming from this interruption, especially given that there was no place-keeping; these errors are what we analyze and model here. We describe these sequence errors in terms of how far ahead or behind of the correct step the selected step was. So, if a step is repeated, it is considered a -1 (perseveration) error, since the selected step is one step behind the correct step. If a step is skipped, it is considered a +1 (anticipation) error, since the selected step is one step ahead of the correct step. If a step two steps back is repeated (such as performing step 7, "Associate", after performing step 8, "Order Info" in Figure 1), that would be a -2 error, since the selected step is two behind the correct next step; and so forth. If an incorrect step was selected, the

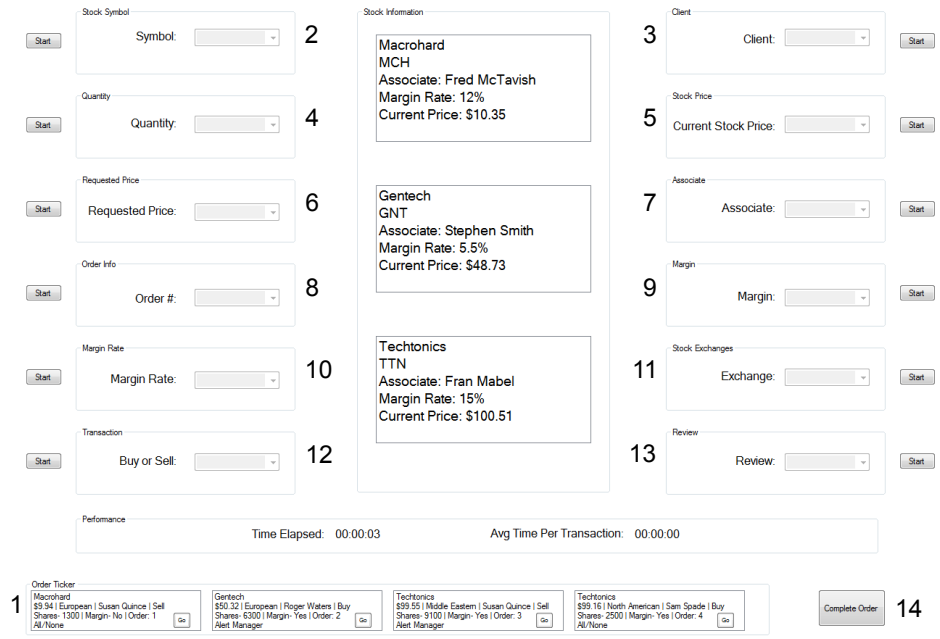


Figure 1: Horizontal stock trader interface. The steps are numbered to indicate the order in which they should be completed; they are shown here for illustrative purposes only.

system beeped and the correct step was highlighted in a red color to allow the participant to recover and continue.

The data from the horizontal task show both similarities and differences with the columnar versions of this task (Figure 2). As with the columnar versions (and other routine sequence error tasks, as well), the most common error for the horizontal task was the immediate (-1) perseveration error, and there were more perseveration errors overall than anticipation errors. Also in accordance with the columnar data, the distribution of errors clusters around the +/-1 errors, and falls away in both directions as the error type gets farther from the correct step (Altmann, Trafton, & Hambrick, 2014).

The horizontal data, however, also show a different pattern of this gradation: namely, a higher proportion of +/-2 errors that occur, compared to +/-1 errors, caused by its distinct spatial layout; an effect which other approaches are unable to explain. The horizontal data also have a wider distribution spread, overall. To preview our approach, we explain the higher proportion of perseveration errors as due to differences in strengthening and priming activation values, which are true regardless of the task interface. In contrast, the difference between the two patterns of data stems primarily from visual priming, which can lead to interface dependent effects. We discuss this further below.

Model Framework

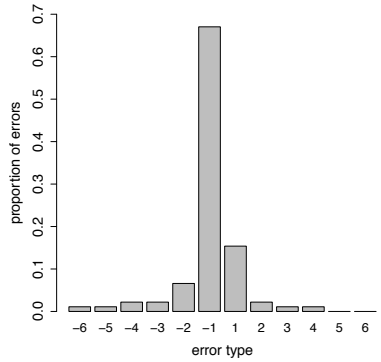
We investigate our account of error prediction within the cognitive architecture ACT-R/E (Trafton et al., 2013), an embodied version of ACT-R (Anderson, Bothell, Lebiere, & Matessa, 1998). ACT-R is an integrated theory of human cognition in which a "production system operates on a declarative memory" (Anderson et al., 1998). In ACT-R, activation of memories has three main components – strengthening, priming, and noise – which are added together to represent a memory's total activation. We next discuss each in turn.

Activation Strengthening

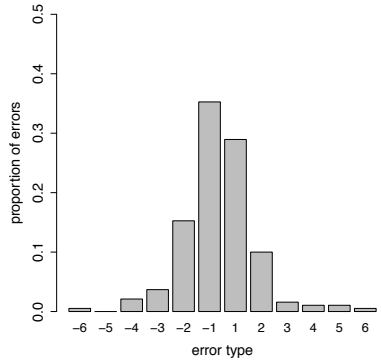
ACT-R's well-established theory of activation strengthening has been shown to be a very good predictor of human declarative memory (Anderson et al., 1998; Anderson, 2007; Schneider & Anderson, 2011). Intuitively, activation strengthening depends on how frequently and recently a memory has been relevant in the past. It is designed to represent the activation of a memory over longer periods of time and, generally, is highest right after the memory has been accessed in working memory, slowly decaying as time passes. Activation strengthening is calculated according to:

$$A_s = \ln\left(\sum_{j=1}^{n} t_j^{-d}\right)$$

where n is the number of times a memory i has been *referenced* (e.g., used in working memory) in the past, t_j is the time that has passed since the jth reference, and d is the a strengthening learning parameter, which defaults to 0.5. Importantly, the negative exponent in this equation implies that recent memories are more differentiated from each other than memories farther in the past.



(a) Distribution of sequence error after an interruption in a columnar interface. Data is from Trafton et al., (2011).



(b) Distribution of sequence errors after an interruption in a horizontal interface. Note the higher proportion of +/-2 errors, and wider distribution spread, that occur.

Figure 2: Contrasting patterns of errors are produced when task versions have different spatial layouts.

Activation Priming

While priming has long been a part of the ACT-R framework (e.g., Anderson, 1983), we adopt a newer, richer notion of priming as part of our approach (Harrison & Trafton, 2010; Hiatt & Trafton, 2013; Thomson et al., 2014). One substantial difference is that, here, activation priming sources from any part of the model's working memory, including the model's goal, intermediate problem representations, and visual representations of what the model is looking at. It then spreads, along associations, to other memories related to those in working memory.

Another main difference is the richer structure of associations. Relevant to our discussion here, associations can be created not only because of explicit correspondences, but also due to *co-occurrence* and *residual* relationships. Co-occurrence associations are created between memories i and j when they are both referenced in working memory at the same time. Residual associations are created between memories that have been referenced in working memory in temporal proximity to one another, even if they are not in working memory at the same time.

Once established, associations have an associated strength value which affects how much activation is spread along them. Mathematically, the strengths (S_{ji}) are:

$$S_{ji} = mas \cdot e^{\frac{-1}{al \cdot R_{ji}}}$$

$$R_{ji} = \frac{f(N_i C_j)}{f(C_j) - f(N_i C_j) + 1}$$

These equations reflect two parameters: *mas*, the maximum associative strength; and *al*, the associative learning rate. The function f tallies the number of times that memory j has been referenced, either independently (C_j) or at similar times to when i has been referenced ($N_i C_j$). An associative strength, intuitively, reflects how strongly a memory, when currently being referenced in working memory, predicts that a memory it primes will be referenced next, and are a function of how often the two memories are referenced by working memory at the same time, versus how often each one is referenced in working memory without the other (represented by R_{ji}). These equations are explained further in Hiatt and Trafton (2013); residual associations are discussed further in Thomson et al. (2014). The associative strengths' qualitative properties are what are key here: namely, that residual associations are typically weaker than co-occurrence associations.

To summarize, associative priming provides the models built in this framework with a rich network for spreading activation that can capture correspondences between memories that are frequently relevant at roughly the same time, as well as correspondences between memories or concepts of different modalities. We rely on both of these features of priming for our model, described below.

Activation Noise

The activation noise of a memory is drawn from a logistic distribution with mean 0 and standard deviation the parameter σ_c. It is a transient value that changes each time it is used, and models the neuronal noise found in the human brain.

Perception and Action

Finally, the model interacts with the world using ACT-R/E's built-in functionality for interacting with the world. Models can view computer interfaces on a simulated monitor; they can act on the world by pushing keys on a simulated keyboard and clicking a simulated mouse.

Activation-Based Model of Sequence Errors

The model's general principles are that it uses activation strengthening and priming to drive progression through a sequential task's steps. At all times, the model maintains in working memory a representation of its goal of completing the task. Before any given step, the model decides what step to perform next by first performing a free retrieval of an episodic code representing the last step that has been completed (or, if there is already an episodic code in working

memory, it simply retains it). Once there is an episodic code in working memory, it then performs a free retrieval of a step, and considers the retrieved step to be the correct next step to perform. It then repeats this process to move on to the next step. At any of these points, as we will show below, priming can come from any item in working memory, including what the model is working at, and can greatly influence the progression of the model through the steps.

Specific to this task, we assume that the representation of the step includes a visual location for where the step is located on the screen, which the model uses as a guide when moving to complete that step. The model is abstract in the sense that it was not concerned with the actual values to fill in to the widgets; instead, its primary responsibility is to attempt to complete the steps of the task in the correct order during each trial. After selecting a widget to work on, it thus clicks the submit button without filling in any values. At the end of a task, the entire working memory is cleared before the model begins the next trial. The model does not perform the post-completion step (e.g., step 14 of Figure 1).

During an interruption, the arithmetic task requires the use of the entire working memory, and so all stock task-related memories (including goals and episodic codes) are removed from working memory. Upon completion of the interrupting task, the model adds a new task goal to working memory, and decides what step to perform next using the process described above. Here, however, in addition to performing a free retrieval of an episodic code, the model looks at the position of the previous step in the interface before attempting to retrieve the next step. Eye-tracking data collected during the experiment showed that, after an interruption, participants look at the correct next step only 13% of the time. Participants, instead, first looked most often at the previous step upon resumption (15%), with the rest generally looking at locations or steps in close proximity to the previous step (such as the step above or below). We assume that this wide spread of where participants look is, in part, due to error in the eye tracker as well as due to error stemming from participants noisily remembering the last location at which they were looking before the interruption. We do not have a theoretical model of visual location memory, but instead model the visual location noise by adding a small amount of Gaussian noise to the position of the last step, and focusing the model's visual attention on the step nearest to that noisy location.

As the sequence process unfolds, many associations are created between the various components involved. Critical to our approach, associations are created between nearby sequential steps, as well as between the visual representation of a step and the next step. Figure 3 illustrates associations for the eighth step of the stock trader experiment; all of the associations are created from co-occurrence except for those between 6.RequestedPrice and 8.OrderInfo, and between 8.OrderInfo and 10.MarginRate, which are residual associations.

During a normal, non-resumption step, the free retrieval

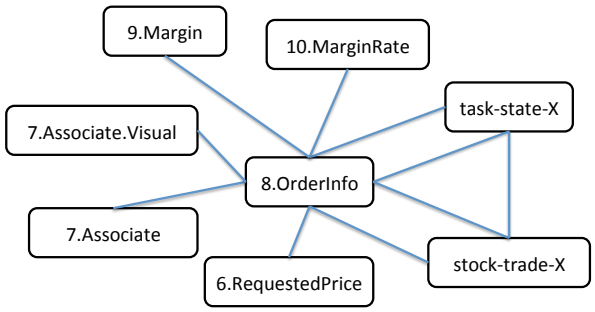


Figure 3: Illustrative associations for the eighth step of the stock trader experiment, Order Info. "Task-state-X" is a placeholder for various episodic codes associated with Order Info over time. "Stock-trade-X" is a placeholder for various goals associated with Order Info over time; episodic codes are associated with the goal in working memory that they co-occur with. Other associations are made, as well, but for clarity we omit those not relevant to our discussion.

of the episodic code is highly affected by both the very near recency of the episodic code (activation strengthening), and from priming from the current goal. Then, the free retrieval of the next step is primarily influenced by priming stemming from the episodic code of the prior step in working memory. These priming and strengthening effects result in a very low error rate during normal task sequence execution.

When resuming after an interruption, however, the model's path is not as clear-cut. For example, since some time has passed since the previous episodic code was last in use, it is more apt to be confused with previous episodic codes, potentially leading to an incorrect retrieval; this is exacerbated by the new goal in working memory, which does not provide priming cues to the previous episodic code. The model also may look at the wrong previous step due to its noisy visual location memory. These potentially incorrect sources of activation mean that the model does not always retrieve the correct step to perform after an interruption. Additionally, even if both these steps do go correctly, priming from the previous episodic code may lead to an anticipation error, because of the residual associations between non-sequential steps. These competing sources of activation comprise the crux of our approach and are explained in more detail, below.

Model Predictions

The model makes a number of activation-based predictions for post-interruption errors in this task. There are two factors that contribute to the final set of activations for a resumption step: the prior episodic code that is retrieved, and what the model is looking at. Different outcomes of these two potential process affect the overall pattern of errors for the retrieved step:

- Retrieve the correct episodic code: Here, priming activation from the correct, prior episodic code biases the model towards the correct answer. It also spreads some activation,

however, to the +1 anticipation step via residual associations (such as how 6.RequestedPrice primes 8.OrderInfo in Figure 3), leading to a possibility of a +1 anticipation error. Occasionally, residual associations can also result in a +2/+3/etc. error.

- Retrieve the wrong episodic code: This happens because of activation strengthening decay. After an interruption, recent episodic codes are close enough in activation that earlier codes may be retrieved. This always results in a bias towards perseveration. A bias towards errors of -1 are the most common, here, but it is possible that errors of type -2/-3/etc. could stem from an incorrectly retrieved prior episodic code as well.
- Look at the right previous step: This biases the model towards performing the correct step next.
- Look at the wrong previous step: Because of visual proximity, the wrong step being looked at is either above or below the previous one. This will bias the model towards the incorrect step being looked at.

Based on these potential process errors, the model makes several predictions for sequence errors. First, the model predicts more perseveration errors than anticipation errors because, intuitively, the model is more likely to retrieve an incorrect past episodic code (potentially leading to an anticipation error) than it is to retrieve an incorrect step based on residual priming (potentially leading to an anticipation step). More technically, the difference in activation strengthening between the past episodic codes is less than the difference in activation priming that an episodic code spreads to the correct vs. future step, leading to more errors occurring there. It follows that this difference also predicts that the most common error type is the -1 perseveration errors. These predictions are interface-independent.

Because of its inclusion of visual priming, the model also predicts the pattern of errors will differ depending on the task's spatial layout. In the previous columnar layout, looking at the wrong previous step spreads activation to the +/-1 steps. In the horizontal layout, however, looking at the wrong previous step spreads activation to the +/-2 steps. Based on this, the model makes two spatial-dependent predictions for the horizontal task version.

First, it intuitively predicts a relatively high proportion of +/-2 errors because of the increase in priming activation those steps receive when the model looks at the incorrect step. Second, it predicts a wider distribution spread, overall. This is because the set of steps commonly competing for retrieval (e.g., {-2, -1, 0, 1, 2}) is larger than the set commonly competing in the columnar task version (e.g., {-1, 0, -1}), making the distribution of steps ultimately selected more spread out.

To reiterate, the key difference between this model and the original memory for goals model is the depth to which priming is utilized by the model. In the MFG model, priming derived from explicit correspondences between the goal and episodic code, and so environmental context (such as priming from visual sources) was not a factor; this makes it unable to capture correspondences between visual objects and other memories, and so it does not predict any shift in error patterns between the two task interface layouts. In addition, MFG does not include residual priming associations in its account, making it very difficult for it to account for +2 errors, even in the columnar version of this task. Finally, in MFG, priming relied upon explicit correspondences between features, some of which were assumed *a priori*, unlike our model which assumes no associations to begin with and builds up its rich network as it trains for, and then tests on, the task.

Model Fit

We ran the model 43 times to simulate data from each of the 43 participants from the horizontal stock trader study. Before beginning testing, the model first performs 3 training trials, where it assumes it is being instructed with the task sequence as it moves through the steps. During these trials, the framework of associations is set up that it will rely upon as it continues on to perform the 20 testing trials, where it continues to learn and update associations as well. Interruptions during the experiment had the same structure as in the original study.

ACT-R/E includes several parameters that affect activation dynamics and, thus, model behavior. The associative learning rate, which affects the rate at which associations are strengthened, was set to 6.5, which represents a fairly brisk rate of learning. There is no standard value for this parameter. The maximum associative strength was within its normal range at 3.0. The activation noise parameter σ_c was 0.08, which is also within its typical range. All other parameters were set to their default values.

We compared the proportion of errors of each type that the model made with the proportion of errors of each type from the study; this allows us to compare the data both qualitatively (overall error trends) and quantitatively (specific distribution of results). The results are shown in Figure 4. Overall, the model's results matched the data very well, with $R^2 = 0.99$ and $RSE = 3.3$. It also qualitatively matches the data's trends, with -1 perseveration errors being the most common error type, and with a higher proportion of +/-2 errors and relatively wider distribution.

Discussion

This work utilizes the underlying principles of memory for goals – that sequential steps are driven by episodic codes, and that those episodic codes are selected based on activation – while expanding its scope. In addition to the strengthening, goal-based priming and noise activation components present in the memory for goals model, our model provides an expanded view of priming that includes priming activation from all items in working memory, and fosters a richer priming structure enabled by additional types of associations. This allows our approach to account for data from sequential tasks with different spatial layouts, something the MFG model was not previously able to do, while also keeping it connected with existing MFG models of sequential errors, post-completion errors, and recovery time that have been shown to

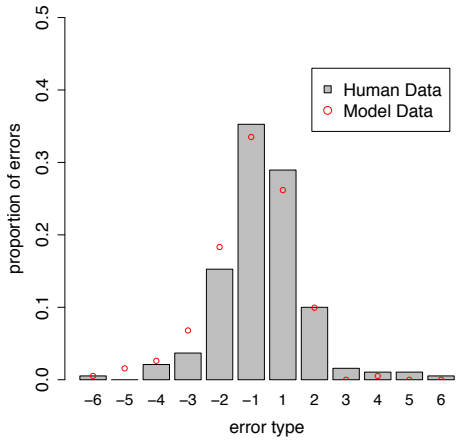


Figure 4: Graph showing the proportion of each type of sequence error from both the experiment and the model.

be successful (Altmann & Trafton, 2002, 2007; Tamborello, II & Trafton, 2014).

Other models of routine sequence errors, such as the interactive activation network (IAN) model (Cooper & Shallice, 2006) and the simple recurrent network (SRN) model (Botvinick & Plaut, 2006), also ignore the specifics of the task interface, and so cannot account for the differences in error patterns that results from an interface layout shift. The ideas behind our expanded priming approach, however, could apply to IAN, which uses environmental and contextual activation to select between schemas that determine the next step.

Although our model does not account for other types of errors, such as capture errors, it does provide some intuition about how those errors take place. Capture errors, for example, occur when a task sequence switches, mid-execution, to a task sequence from a different, but usually related, task; for example, checking e-mail after sitting down at a computer when one originally intended to check the weather. Capture errors, intuitively, involve much environmental context and visual cues, which can be accounted for by our approach.

Acknowledgments

We would like to thank Malcolm McCurry and Kevin Zish for providing the experimental data we utilize here. We would also like to thank Frank Tamborello for his helpful comments. This work was supported by the Office of the Secretary of Defense / Assistant Secretary of Defense for Research and Engineering (LH) and the Office of Naval Research (GT). The views and conclusions contained in this paper do not represent the official policies of the U.S. Navy.

References

Altmann, E. M., & Trafton, J. G. (2002). Memory for goals: An activation-based model. *Cognitive Science*, *26*, 39-83.

Altmann, E. M., & Trafton, J. G. (2007). Time course of recovery from task interruption: Data and a model. *Psychonomics Bulletin & Review*, *14*(6), 1079-1084.

Altmann, E. M., Trafton, J. G., & Hambrick, D. Z. (2014). Momentary interruptions can derail the train of thought. *Journal of Experimental Psychology: General*, *143*, 215-226.

Anderson, J. R. (1983). A spreading activation theory of memory. *Journal of Verbal Learning and Verbal Behavior*, *22*(3), 261–295.

Anderson, J. R. (2007). *How can the human mind occur in the physical universe?* Oxford University Press.

Anderson, J. R., Bothell, D., Lebiere, C., & Matessa, M. (1998). An integrated theory of list memory. *Journal of Memory and Language*, *38*(4), 341–380.

Botvinick, M. M., & Plaut, D. C. (2006). Such stuff as habits are made on: A reply to cooper and shallice (2006). *Psychological Review*.

Byrne, M. D., & Bovair, S. (1997). A working memory model of a common procedural error. *Cognitive Science*, *21*, 31-61.

Cooper, R. P., & Shallice, T. (2006). Hierarchical schemas and goals in the control of sequential behavior. *Psychological Review*.

Harrison, A. M., & Trafton, J. G. (2010). Cognition for action: an architectural account for "grounded interaction". In *Proceedings of the Annual Meeting of the Cognitive Science Society.*

Hiatt, L. M., & Trafton, J. G. (2013). The role of familiarity, priming and perception in similarity judgments. In *Proceedings of the Annual Meeting of the Cognitive Science Society.*

Ratwani, R. M., & Trafton, J. G. (2011). A real-time eye tracking system for predicting and preventing postcompletion errors. *Human-Computer Interaction*, *26*(3), 205-245.

Reason, J. T. (1984). Absent-mindedness and cognitive control. *Everyday Memory, Actions and Absent-mindedness*.

Schneider, D. W., & Anderson, J. R. (2011). A memory-based model of hick's law. *Cognitive Psychology*, *62*(3), 193–222.

Tamborello, II, F. P., & Trafton, J. G. (2014). A generalized process model of human action selection and error and its application to error prediction. In *Proceedings of the Annual Meeting of the Cognitive Science Society.*

Thomson, R., Bennati, S., & Lebiere, C. (2014). Extending the influence of contextual information in ACT-R using buffer decay. In *Proceedings of the Annual Meeting of the Cognitive Science Society.*

Trafton, J. G., Altmann, E. M., & Ratwani, R. M. (2011). A memory for goals model of sequence errors. *Cognitive Systems Research*, *12*, 134-143.

Trafton, J. G., Hiatt, L. M., Harrison, A. M., Tamborello, II, F. P., Khemlani, S. S., & Schultz, A. C. (2013). ACT-R/E: An embodied cognitive architecture for human-robot interaction. *Journal of Human-Robot Interaction*, *2*(1), 30-55.

Unified Theories of Cognition: Newell's Vision after 25 Years

Glenn Gunzelmann (glenn.gunzelmann@us.af.mil)
Cognitive Models & Agents Branch, Air Force Research Laboratory
711 HPW/RHAC, 2620 Q St., Building 852
Wright Patterson Air Force Base, OH 85212-6061 USA

Introduction

It has been 25 years since Unified Theories of Cognition was published (Newell, 1990). In it, Newell outlines a vision to inspire generations of cognitive scientists and cognitive modelers; a quest for theories that provide comprehensive accounts of the human mind. As he put it:

> "A single system (mind) produces all aspects of behavior. It is one mind that minds them all. Even if the mind has parts, modules, components, or whatever, they all mesh together to produce behavior… If a theory covers only one part or component, it flirts with trouble from the start. It goes without saying that there are dissociations, interdependencies, impenetrabilities, and modularities… But they don't remove the necessity of a theory that provides the total picture and explains the role of the parts and why they exist." (Newell, 1990, pp. 17-18).

The intervening years have produced a wealth of research progress on many fronts. One important measure is the number of candidate theories that have emerged. In his book, Newell explicitly pointed to the need for multiple unified theories of cognition to drive progress through model comparison. The 1990's and early 2000's saw a large expansion in theories providing broad accounts of human cognitive capacities (see, e.g., Gluck & Pew, 2005). In addition, there have been some attempts to formally compare alternative theories to address their associated strengths and weaknesses (e.g., Gluck & Pew, 2005; Gonzalez, Lebiere, & Warwick, 2009).

In conjunction with the increase in candidate architectures, the scope of these theories also has broadened, leading to more complete theories that have incorporated many critical aspects of human cognition (e.g., Anderson, 2007; Bach, 2008; Laird, 2012). This was a critical component of Newell's vision, addressing the need to account for a wider array of cognitive mechanisms to provide ever more comprehensive and inclusive accounts of the capacities and limitations of human cognition. It is worthwhile to take stock of these achievements.

Despite the evidence of progress and sustained contributions to cognitive science that have emerged from the pursuit of unified theories of cognition, the zeitgeist has evolved. Opinions differ significantly within the community represented at this conference regarding the current state of cognitive architectures, trends in their development, and where they should go in the future (e.g., Kurup, Gunzelmann, Lewis, Salvucci, & Taatgen, 2012).

In the broader cognitive science community, there is also a tendency to focus on phenomena and challenges that play to the strengths of the modeling formalisms that are used for model development. As McClelland (2009) notes, different approaches are often adopted because they are "particularly apt for addressing certain types of cognitive processes and phenomena. Each has its core domains of relative advantage, its strengths and weaknesses, and its zones of contention where there is competition with other approaches" (p. 25). This perspective is not new. It contrasts with Newell's vision, which stood in opposition to his perception of the prevailing trends in the field. Specifically, Newell perceived that cognitive science had become "too focused on specific issues and had lost sight of the big picture needed to understand the human mind." (Anderson & Lebiere, 2003, p. 587).

The International Conference on Cognitive Modeling is the premier venue for cognitive modeling research. Moreover, it emerged from research pursuits directly aligned with Newell's vision. This year's conference provides a unique opportunity to revisit Newell's vision, and look to the future of our community.

Presenters

The presenters in this symposium will focus on Newell's vision for unified theories of cognition, discuss whether and how that vision drives their research, and comment on the extent to which it still defines an appropriate vision for the community. The participants in the symposium have been selected to represent a cross-section of the community, each of whom will provide a unique perspective on the topic.

Glenn Gunzelmann

Current cognitive architectures capture many of the capacities and limitations of human cognition. However, the current state of the art falls well short of Newell's vision for unified theories. Many critical cognitive abilities identified by Newell remain poorly understood (e.g., perception, language, emotions). In addition, the overwhelming majority of computational cognitive models are based on the implicit assumption that the human mind continually operates in an efficient, effective, and goal-directed manner. Our models do not get hungry, fatigued, angry, or distracted. Part of Newell's vision entailed developing theories that make it "further down the list" of phenomena that characterize the human mind (Newell, 1990, p. 16). Unfortunately, too little research in the cognitive modeling

community addresses this challenge today. Instead, unified theories are used increasingly to explain isolated phenomena and validate micro-theories. For cognitive architectures to remain relevant in the future of cognitive science, the community must take seriously Newell's vision, and refocus on the challenges of developing a theory that explains the roles for the various components, why they exist, and how they are integrated to create the human mind.

Paul Rosenbloom

Newell's call for integrated approaches to cognition is as relevant as ever, but broad progress over the past 25 years in both the natural and artificial sciences enables, and even demands, we be even more ambitious today when thinking about integration. Can we build single systems that span from the biological band, through the cognitive and rational bands, up to the social band? Can we complete the processing path from perception and attention, through cognition and affect, out to motor control without arbitrary boundaries between these parts? And can integrated approaches inform us about both natural and artificial cognition? I will discuss how an attempt to answer such questions, toward ultimately yielding what could be called a *grand unification*, has driven the development of the Sigma cognitive architecture and system (Rosenbloom, 2013).

Dario Salvucci

Newell's vision for unified theories of cognition has no doubt stood as the centerpiece of cognitive-architecture research since his seminal "20 Questions" paper (Newell, 1973). In this paper, Newell proposed three complementary activities in this effort: the use of "complete processing models," exemplified by production systems; the analysis of complex tasks, beyond those involved in simple psychological paradigms; and the development of "one program for many tasks," a single model that acts in a variety of task domains. Arguably the cognitive-architecture community has focused largely on the first and second activities, while the third activity has received much less attention. I will discuss some recent efforts (e.g., Salvucci, 2013) that aim to extend the capabilities of cognitive architectures in this third direction.

Iris van Rooij

New formal and conceptual tools for theorizing about cognition have developed since Newell voiced his concerns about experimental psychology in his seminal "20 questions" paper, and proposed specific ways of dealing with them. Using these new tools we can cast our theoretical net even wider than Newell perhaps envisioned. For instance, important advances have been made in theorizing about cognition at a level *above* that of mechanism, viz., what Marr (1982) called the 'computational level' (and Anderson (1990) calls the 'rational level'). I will discuss how theorizing at this level may be useful for addressing a challenge that remains to this day: How to make models that can scale beyond specific experimental tasks and explain cognition in its full domain generality?

Marieke van Vugt

As a relative outsider, it struck me that the adoption of Unified Theories of Cognition (UTCs) is relatively low. Moreover, the community faces criticisms of not being falsifiable. Those criticisms are raised especially by modelers who focus on modeling a single essential cognitive operation. Indeed, I share these concerns, and I think it is crucial that the community develops good methodology for comparing and testing models, and their dependence on model parameters. On the other hand, I think UTCs can provide insight not only in what happens during the trials of a single task but also what happens between those trials as a person gets bored, tired, etc. Furthermore, in the domain of neuroscience UTCs have the potential to describe how different parts of the brain collaborate to guide information flows across different task stages. In the age of big data, both simple mathematical models and UTCs are more necessary than ever.

References

Anderson, J. R. (1990). The adaptive character of thought. Psychology Press.

Anderson, J. R. (2007). *How can the human mind occur in the physical universe?*. Oxford: Oxford University Press.

Anderson, J. R., & Lebiere, C. (2003). The Newell test for a theory of cognition. *Behavioral and Brain Sciences, 26(05)*, 587-601.

Bach, J. (2009). *Principles of Synthetic Intelligence. Psi, an architecture of motivated cognition*. Oxford: Oxford University Press.

Gluck, K. A., & Pew, R. W. (Eds.). (2005). *Modeling human behavior with integrated cognitive architectures: Comparison, evaluation, and validation*. New York, NY: Psychology Press.

Lebiere, C., Gonzalez, C., & Warwick, W. (2009). A comparative approach to understanding general intelligence: Predicting cognitive performance in an open-ended dynamic task. In B. Goertzel, P. Hitzler, & M. Hutter (Eds.), *Proceedings of the Second Conference on Artificial General Intelligence* (pp. 103-107). Amsterdam, The Netherlands: Atlantis Press.

Kurup, U., M. D., Gunzelmann, G., Lewis, C., Salvucci, D. & Taatgen, N. (2012, July). Cognitive architectures: State, trends, and roadmap. Symposium presented at the *19th Annual ACT-R Workshop*. Pittsburgh, PA.

Laird, J. (2012). *The Soar cognitive architecture*. Boston, MA: MIT Press.

Marr, D. (1982). *Vision*. Freeman: New York.

McClelland, J.L. (2009). The place of modeling in cognitive science. *Topics in Cognitive Science, 1(1)*, 11-38.

Newell, A. (1990). *Unified Theories of Cognition*. Cambridge, MA: Harvard University Press.

Newell, A. (1973). You can't play 20 questions with nature and win: Projective comments on the papers of this symposium. In *Visual Information Processing* (pp. 283-308). New York: Academic Press.

Rosenbloom, P. S. (2013). The Sigma cognitive architecture and system. *AISB Quarterly*, *136*, 4-13.

Salvucci, D. D. (2013). Integration and reuse in cognitive skill acquisition. *Cognitive Science, 37*, 829-860.

Modeling mind-wandering: a tool to better understand distraction

Marieke K. van Vugt (m.k.van.vugt@rug.nl) & Niels A. Taatgen (n.a.taatgen@rug.nl)
Institute of Artificial Intelligence and Cognitive Engineering, University of Groningen
Nijenborgh 9, 9747 AG Groningen, The Netherlands

Jérôme Sackur (jerome.sackur@gmail.com) & Mikaël Bastian (bastian.mikael@gmail.com)
Ecole Normale Superieure, Paris, France.

Abstract

When we get distracted, we may engage in mind-wandering, or task-unrelated thinking, which impairs performance on cognitive tasks. Yet, we do not have cognitive models that make this process explicit. On the basis of both recent experiments that have started to investigate mind-wandering and introspective knowledge from for example meditators, we built a process model of distraction in the form of mind-wandering. We then tested the model by predicting performance on tasks used in mind-wandering studies. We showed that we could both predict task performance as well as the participants' responses to questions about what they were thinking about. This improved understanding of mind-wandering could be used in the future to revise our models of when, how, and why distraction occurs. For example, our model could be used to examine how the effect of distraction on task performance depends on the type of mind-wandering (e.g., rumination versus day-dreaming).

Keywords: ACT-R; mind-wandering; meditation; distraction

Introduction

Reports suggest that we spend more than half of our waking time mind-wandering (Killingsworth & Gilbert, 2010). While it is known that mind-wandering also affects task performance, there are very few studies that examine mind-wandering experimentally, and models of this cognitive process are even more scarce. In fact, most studies consider these distraction processes as some form of mental noise. However, it is likely that mind-wandering is not a unitary process but is a collection of different processes. For example, rumination about ones' fears may be very different from daydreaming about an upcoming beach trip. It will be much more difficult to disengage from the rumination than from the day-dreaming, and rumination will activate a much smaller set of memories more strongly. Differentiating between the effects of these types of mind-wandering requires an explicit model.

Mind-wandering refers to processes of task-unrelated thinking (see Smallwood & Schooler, 2014, for a review). This is a process that is not triggered by external distractors, but rather triggered by the mind itself. Recent studies have started to investigate mind-wandering experimentally by means of various tasks in which people are known to zone out such as slow sustained attention tasks (e.g., Cheyne, Carriere, & Smilek, 2009) or reading a boring text (e.g., McVay & Kane, 2012). To assess mind-wandering, experimenters may insert *thought probes* into their task, which ask the participants about whether they were on-task or off-task (e.g., Cheyne et al., 2009). It has been found that there are more errors and response time variability increases during self-reported mind-wandering (Bastian & Sackur, 2013).

Mind-wandering may–depending on circumstances–be either adaptive or non-adaptive. While mind-wandering during a task that requires continuous cognitive control may be problematic, mind-wandering during a task that does not require continuous attention may in fact contribute to improved problem-solving and creativity, since it frequently involves prospective memory (Baird et al., 2012). At present, there are several theories of mind-wandering that have emphasized different aspects of this process. The executive failure theory states that mind-wandering occurs out of a failure to focus attention on relevant information (McVay & Kane, 2009), while the perceptual decoupling theory states that mind-wandering is primarily a process of decoupling from the external environment, such that it can be devoted to internal processes (Smallwood, Beach, Schooler, & Handy, 2008; Smallwood et al., 2011). Evidence for perceptual decoupling comes from studies that have found that the amplitude of evoked potentials is reduced during states of task-unrelated thought (Smallwood et al., 2008). In addition, the pupil responds less to presented stimuli during states of mind-wandering (Smallwood et al., 2011). Instead of processing perceptual stimuli, the brain appears to be engaged in episodic processing during the periods of distraction (Andrews-Hanna, Smallwood, & Spreng, 2014). Executive failure theory is based on studies that relate cognitive control abilities to the ability to resist mind-wandering (Kane & McVay, 2012). Alternatively, it has been suggested that mind-wandering results from failures in meta-cognition, the ability to observe ones' thoughts (Fox & Christoff, 2014).

None of the above-mentioned theories has been formalized in computational models. Closest related to studying mind-wandering come models of distraction and fatigue. For example, Gunzelmann, Gross, Gluck, and Dinges (2009) investigated the effects of fatigue on performance on a monotonous psychomotor vigilance task. According to his model, fatigue impacted a parameter used to compute the utility of particular task strategies (this parameter has been associated with motivation). This parameter change made random key presses more likely as the participant became more tired. In addition, they modelled lapses in behavior by having productions that had a sufficiently low activation that they would only be performed as a result of random fluctuations in their activation parameter. Note how their model does not model distraction through mind-wandering as an explicit process, but instead assumes that the cognitive system is not functioning during distraction. Similarly, Gonzalez, Best, Healy, Kole,

and Bourne Jr. (2011) modelled the effects of fatigue on a data entry task as a reduction in motivation in combination with a reduction in attentional control. This attentional control parameter affects the activation of different pieces of information, and the larger this parameter is, the better these pieces of information can be distinguished.

A previous ACT-R model of a sustained attention task that is often used in mind-wandering studies (Peebles & Bothell, 2010) focused primarily on explaining response times decrease just preceding an attentional lapse (as reflected in an error). They produced this phenomenon by a competition between two response strategies: one strategy is responding whenever a stimulus is detected, which is very fast, while an alternative strategy first checks the stimulus before responding. When the fast strategy fails then ACT-R will switch to the most costly slow strategy. Note that this model does not implement an explicit cognitive mechanism for what happens during distraction. Here we intend to build on that previous model by implementing a competition between a "distracted" and an "attentive" model, where the distracted model makes the mind-wandering process explicit.

Model

Our model of distraction (Figure 1) consists primarily of a competition between a sub-model for paying attention to the task and a sub-model for mind-wandering. The model was implemented in the Adaptive Control of Thought-Rational (ACT-R) cognitive architecture (Anderson, 2007). Tasks are implemented in this cognitive architecture by specifying a set of if-then statements (production rules) that describe how different cognitive resources interact. Two ACT-R mechanisms are of crucial importance for our model. First, ACT-R has a memory store, where the activation of each memory chunk determines its use and its retrieval time. The activation in turn is determined by how often a chunk is retrieved, its activation at baseline, and how much activation spreads from other, related memory chunks. The second mechanism that determines what happens in the model at a particular moment is the utility associated with each production rule. When production rules help to generate rewards, their utility goes up, leading them to be used more frequently. However, given that in mind-wandering there are no external reward processes that guide the process, we will not make use of this second mechanism in our model.

In this application, the model starts out by focusing its attention on the stimulus on the screen. When there is a stimulus, it will process the stimulus and perform the appropriate action. When there is no stimulus, it will continually run a production which checks what the most active goal ("paying attention" or "distraction") is in declarative memory ("check whether attending" in Figure 1). The activations of the goals in declarative memory are governed by rules from episodic memory decay (Altmann & Gray, 2008). This means that items that are retrieved in activation, but over time the activation decays. At the start of the task, the "paying attention"

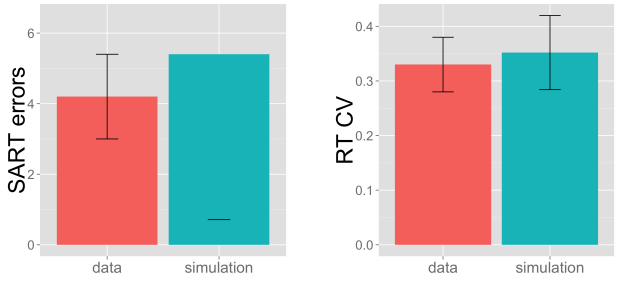


Figure 2: Model simulation of performance in the SART task described by Mrazek and colleagues (2012). The model (blue) captures both the number of SART errors and variability in response time observed empirically (red).

goal is activated because it has been retrieved from episodic memory. This goal then decays over time, and at some point the "distraction" goal becomes stronger (Figure 1). When the "distracted" goal is retrieved by this checking production, the mind-wandering model commences.

Mind-wandering consists of a continuous retrieval of declarative memories. The retrieval process keeps continuing until at some point a memory that says "remember to attend" is retrieved. At that point, the model returns to paying attention and the whole cycle can start again. There is spreading activation between memories, which ensures that–as in real life–memories that are of the same valence (positive, negative, or neutral) tend to be recalled in sequence (van Vugt, Hitchcock, Shahar, & Britton, 2012).

Our main goal in this paper is to find out whether the hypothesized mind-wandering model can in fact describe empirical mind-wandering data. Studies have experimentally studied mind-wandering by giving participants a very boring task, in which participants are likely to drift off. Here, we will model data from two experiments: Mrazek, Smallwood, and Schooler (2012) (Experiment 1) and Bastian and Sackur (2013) (Experiment 2). Both experiments are variants of the sustained attention to response task (SART), in which participants are requested to press a button as quickly as possible every time a target is presented, but to withhold a button press to a more rarely presented non-target (Cheyne et al., 2009; Smallwood et al., 2004).

When the distraction model is inserted in a model of the SART task, we assume performance is determined by the following mechanisms, building on Peebles and Bothell (2010)'s model. When task stimuli are presented while the model is in paying attention mode, the model will look at the stimuli and retrieve the relevant stimulus-response mapping from episodic memory. Conversely, when the model is distracted, it will *not* retrieve the stimulus-response mapping from episodic memory but instead respond with the habitual response. However, responding may take a little while, because the model will only be able to respond when it is not busy retrieving a memory in its mind-wandering train. This

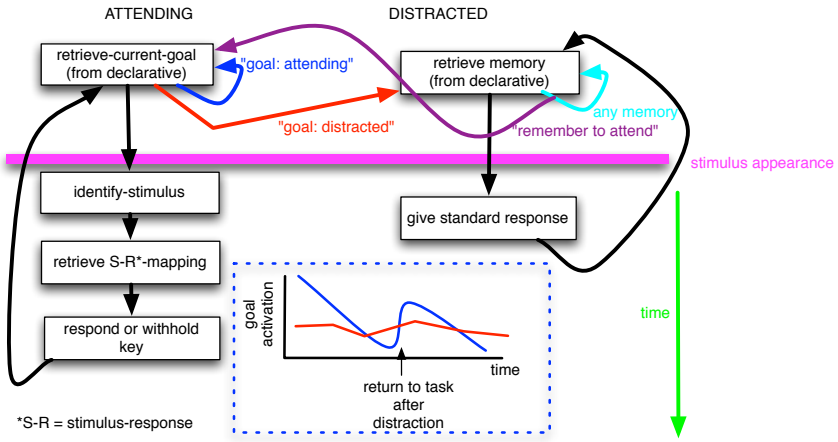


Figure 1: Model time line. Each box corresponds to a production (some less important productions have been left out). The model starts on the left top with retrieving its current goal, corresponding to goal checking. Initially, the "attending" goal has the highest activation (see dashed blue box), but over time the attending goal declines in activation to become similar to the distracted goal. When this "distracted" goal is retrieved, the model switches to retrieving memories from declarative memory, representing mind-wandering. Mind-wandering (cyan) continues until "remember to attend" (purple) is retrieved. At that time, the model goes back to monitoring goals. When a stimulus is presented (pink line), then the model identifies it and retrieves the stimulus-response mapping in case it is attending. When it is distracted, it finishes retrieving the current distraction and then presses the default response.

potential delay before responding is responsible for creating the increase in response time variability that is typically observed in mind-wandering studies (Bastian & Sackur, 2013; Mrazek et al., 2012). In Experiment 2, thought probes may also be presented. Whenever a thought probe occurs, the model will press the "on-task" button whenever it is in paying attention mode, while it will press the "off-task" button when it is busy retrieving memories from episodic memory during distraction. The models can be retrieved from `http://www.ai.rug.nl/~mkvanvugt/mindwanderingModels.zip`. A flow chart of the model is shown in Figure 1.

Model testing

Experiment 1

We first used our model to simulate the average data published by Mrazek et al. (2012). In this experiment, the targets consisted of the letter "O", and non-targets consisted of the letter "Q." Stimuli were presented for 2 s with an interstimulus interval of 2500 ms. There were in total 240 stimuli; 216 targets and 24 non-targets.

Figure 2 shows that the simulated performance of the model reproduces both the observed number of SART errors and the coefficient of variation of the response time. Errors are produced whenever the model is mind-wandering. The coefficient of variation results from variability in memory retrieval time.

Having established the model can produce behavior similar to human participants, it becomes possible to examine *how* the model produces this behavior. Figure 6 shows that according to our model, the frequency of distractions shows a U-shape: initially, there are quite a few distractions, which

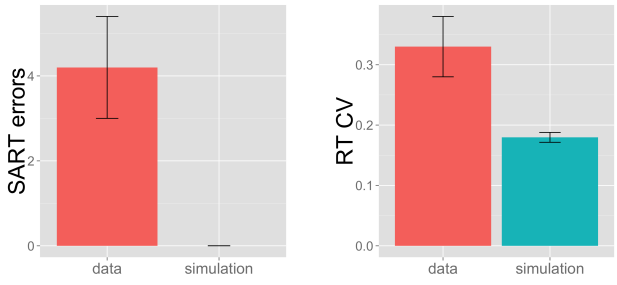


Figure 3: Behavior produced by a model in which the thought pump is ended with a production "end-thought-pump" rather than a specific memory retrieval.

reduces in the middle of the task, and increases at the end. While there is evidence for an increase in the frequency of distraction towards the end of the task (Bastian & Sackur, 2013), it is not clear whether the distraction at the beginning of the task is plausible. Future studies that have better measures of the frequency of distraction (e.g., Katidioti et al., submitted) should clarify this issue.

An important question is how crucial the proposed mechanism is for terminating mind-wandering. A simpler mechanism for achieving this goal may be a direct competition between the "distraction" and "paying attention" goals. In other words, at any moment during mind-wandering, a production could fire that reflects the end of the mind-wandering process. Figure 3 shows that this alternative mechanism makes too few mistakes because the episodes of mind-wandering are

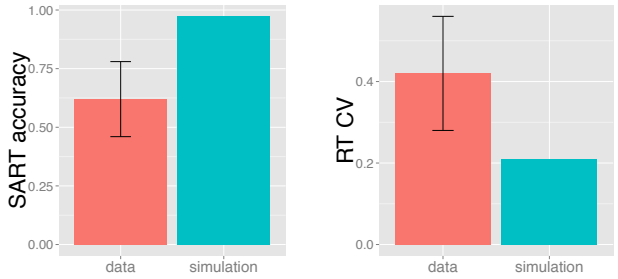


Figure 4: Model simulation of performance in the SART task described by Bastian & Sackur (2013). The model (blue) captures both accuracy and variability in response time observed empirically (red) reasonably well.

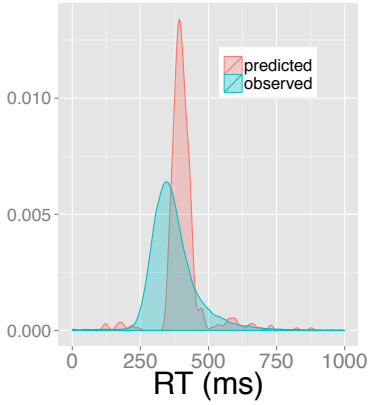


Figure 5: Response time distribution of SART performance of Experiment 2, overlaying actual data (blue) with model predictions (red).

terminated much too quickly. A caveat in this assertion is that there may potentially be ways to change ACT-R parameters to increase the duration of mind-wandering episodes.

The duration of mind-wandering is determined by the episodic memory retrievals that make up the mind-wandering process. When the pool of to-be-retrieved memories is larger, then distractions will tend to persist longer, because the chance that the distraction-ending memory is retrieved is smaller. A larger number of retrievable memories corresponds to something akin to the number of retrieval cues. In some contexts, people may be able to think of many different things, while in other context they can only retrieve a limited number of items. A further determinant of distraction duration is the association structure of the distracting memories. When memories spread activation to the memory that ends the distraction, this will decrease distraction duration; when they spread activation to other memories, this increases distraction duration. These factors could potentially be manipulated to account for individual differences in distractability.

Together, these results show that it is possible to use our model of mind-wandering to simulate performance on a

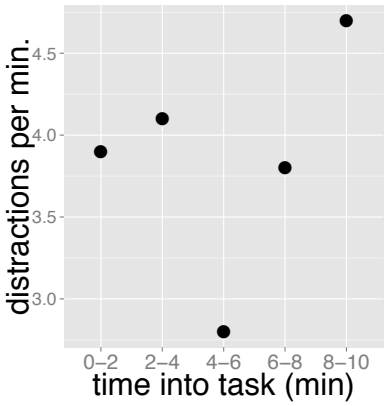


Figure 6: Predicted frequency of distractions in Experiment 1.

SART task. However, the results are fairly weak since we only fit two average numbers: the number of errors and response time variability. More data are needed to adequately constrain our cognitive model. We therefore use the complete dataset collected by Bastian and Sackur (2013) to further test the model, which allows us to examine more behavioral measures. An additional advantage of that dataset is that the task was interspersed with thought probes that asked the participant to report on the content of their thoughts. The responses to thought probes are another constraining factor for our model. Moreover, it highlights an important advantage of modeling mind-wandering explicitly, as we did here. When a model has no explicit process description of mind-wandering, it cannot predict responses to thought probes.

Experiment 2

In Experiment 2, participants performed a very similar task as in Experiment 1, although the timing was a little bit different. Importantly, we did not change the model parameters at all to predict performance in this task. In this experiment, the non-target consisted of the digit 3, and the target consisted of all other digits. The digits were presented for 500 ms with an interstimulus interval of 1500 ms. There were in total 888 stimuli; 811 targets and 77 non-targets. In addition, 24 thought probes that were randomly interspersed in the task. These thought probes asked a series of four questions about task performance. First, participants were asked "How focused were you on the task? 0: on-task, 1: task-related thought, 2: distraction, 3: mind wandering." Secondly, "Did you know that you were in the just-reported mental state or did you only notice it when asked? 0=aware, 1=unaware." The third question concerned the phenomenology/type of the thoughts, while the fourth question assessed the temporal orientation of the thoughts (past, present, future, or no particular time). In this paper, we will only model the question about whether the participant is on-task.

Figure 4 shows that task performance could be modelled accurately with the model for Experiment 1, although in this

case, the model is performing slightly too well for the participants. Potentially, model fits could be improved by adjusting parameters.

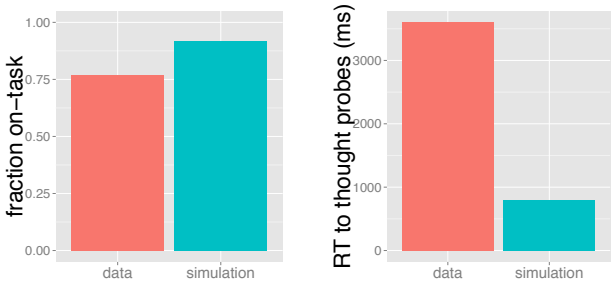
In addition to average responses, it is also important to consider the entire response time distribution (e.g., Ratcliff, 2002). Figure 5 shows that the modeled and observed response time distributions for task performance overlay considerably, although the response time variability predicted by the model is too small.

Finally, our explicit model of mind-wandering allows us to model the responses to questions about the contents of thoughts. At random moments in the task, the participant is asked whether they were on-task or off-task. Figure 7 shows that the model over-estimates the proportion of being on-task relative to human participants, which is consistent with the model's overperformance evident in Figure 4. Another notable feature visible in Figure 7 is that participants require about 3–4 seconds to formulate their response to the question "Were you on-task." This response time is much longer than those observed for cognitive tasks, and our model is not able to predict it. Two potential mechanisms that could be involved in generating this time are (1) the conversion of a pre-verbal into a verbal response (Teasdale & Chaskalson, 2011) or (2) mental time travel to several moments before the thought probe appeared to retrieve the memories that occurred at that time (Howard & Kahana, 2002; Tulving, 2002). Future modeling efforts should investigate these ideas.

Discussion

We proposed a model that describes mind-wandering mechanistically. We showed how it could account for task performance in two experiments featuring the Sustained Attention to Performance task (without changing model parameters between the two). While previous models only treat distraction abstractly as noise in the cognitive system (VandeKerckhove & Tuerlinckx, 2007) or an absence of cognitive activity (Gunzelmann et al., 2009), we made an explicit model of the mind-wandering process. This allowed us to not only model task performance, but also responses to thought probes. Our model provides a potential implementation of the executive failure theory, where in our case executive failure is implemented as a failure to keep checking what the current goal is. It is also related to perceptual decoupling in that perceived stimuli are not further analyzed, but it places the constraints at a higher level than initial stimulus processing.

In the future, our explicit model of mind-wandering could allow us to examine the effect of different types of mind-wandering on cognitive processing. For example, depressive rumination impairs task performance, and our model can make predictions about exactly how it does so. By describing the thought process from moment to moment, we will be able to investigate how not only cognitive control factors affect task performance, but also the content of thought. For this to be done, it will be important to populate the model with memories that reflect the distribution of memories that is observed in actual participants. Such data can be obtained from studies that report the content of thoughts in thought probes (Bastian & Sackur, 2013).



(a) (b)

Figure 7: Observed (red) and modeled (blue) responses to thought probe ("Are you on-task?") in Experiment 2. (a) fraction of on-task responses, (b) median time taken to respond to thought probes.

While our model makes a promising start with modeling task performance, there is still some work to be done. Our model predicts better performance in Experiment 2 than is produced by the participants (Figure 4). In addition, the frequency of mind-wandering episodes (Figure 6) shows a U-shape, rather than the previously reported increase Bastian and Sackur (2013).

The most dramatic discrepancy is in the response times to thought probes, which are much faster in our model than in real participants. A future iteration of our model may need to include a mechanism by which the participant converts the content of thoughts into a verbal report.

At the same time, we have relatively few datapoints and cannot make strong inferences about our model. One possible future direction may be predicting the frequency of distractions. We have recently started to measure those by means of eye movements to an ambient video monitor (Katidioti et al., submitted). Our model could potentially describe how the frequency of distraction changes over time, and depends on different factors such as task difficulty. This is particularly important because it is often thought that introspective judgments are unreliable (Larson, Perlstein, Stigge-Kaufman, Kelly, & Dotson, 2006).

In short, we have developed a mechanistic model of mind-wandering. This model can in the future be used to disentangle different types of mind-wandering. In addition, future experiments should elucidate the neural correlates of distraction and mind-wandering, such that those measures can be used to track distraction online (Bengson, Mangun, & Mazaheri, 2012).

Acknowledgments

MvV developed the basic model during a stay as Mind & Life visiting scholar at Amherst College.

References

Altmann, E. M., & Gray, W. D. (2008). An integrated model of cognitive control in task switching. *Psychological review*, *115*(3), 602.

Anderson, J. R. (2007). *How can the human mind occur in the physical universe?* Oxford University Press.

Andrews-Hanna, J. R., Smallwood, J., & Spreng, R. N. (2014). The default network and self-generated thought: component processes, dynamic control, and clinical relevance. *Annals of the New York Academy of Sciences*, *early view*. doi: 10.1111/nyas.12360

Baird, B., Smallwood, J., Mrazek, M. D., Kam, J. W. Y., Frank, M. J., & Schooler, J. W. (2012). Inspired by distraction. mind wandering facilitates creative incubation. *Psychological Science*, *23*(10), 1117-1122. doi: 10.1177/0956797612446024

Bastian, M., & Sackur, J. (2013). Mind wandering at the fingertips: automatic parsing of subjective states based on response time variability. *Frontiers in Psychology*, *4*, 573. doi: 10.3389/fpsyg.2013.00573

Bengson, J. J., Mangun, G. R., & Mazaheri, A. (2012). The neural markers of an imminent failure of response inhibition. *NeuroImage*, *59*(12), 1534-1539. doi: 10.1016/j.neuroimage.2011.08.034

Cheyne, D., Carriere, J. S. A., & Smilek, D. (2009). Absent minds and absent agents: Attention-lapse induced alienation of agency. *Consciousness and Cognition*, *18*, 481-493. doi: 10.1016/j.concog.2009.01.005

Fox, K. C., & Christoff, K. (2014). Metacognitive facilitation of spontaneous thought processes: When metacognition helps the wandering mind find its way. In *The cognitive neuroscience of metacognition* (pp. 293–319). Springer. doi: 10.1007/978-3-642-45190-4_13

Gonzalez, C., Best, B., Healy, A. F., Kole, J. A., & Bourne Jr., L. E. (2011). A cognitive modeling account of simultaneous learning and fatigue effects. *Cognitive Systems Research*, *12*, 19-32. doi: 10.1016/j.cogsys.2010.06.004

Gunzelmann, G., Gross, J. B., Gluck, K. A., & Dinges, D. F. (2009). Sleep deprivation and sustained attention performance: Integrating mathematical and cognitive modeling. *Cognitive Science*, *33*, 880-991.

Howard, M. W., & Kahana, M. J. (2002). A distributed representation of temporal context. *Journal of Mathematical Psychology*, *46*, 269-299.

Kane, M. J., & McVay, J. C. (2012). What mind wandering reveals about executive-control abilities and failures. *Current Directions in Psychological Science*, *21*, 348354.

Killingsworth, M. A., & Gilbert, D. T. (2010). A wandering mind is an unhappy mind. *Science*, *330*(6006), 932.

Larson, M. J., Perlstein, W. M., Stigge-Kaufman, D., Kelly, K. G., & Dotson, V. M. (2006). Affective context-induced modulation of the error-related negativity. *NeuroReport*, *17*(3), 329–333.

McVay, J. C., & Kane, M. J. (2009). Conducting the train of thought: Working memory capacity, goal neglect, and mind-wandering in an executive-control task. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *35*, 196-204. doi: 10.1037/a0014104

McVay, J. C., & Kane, M. J. (2012). Why does working memory capacity predict variation in reading comprehension? on the influence of mind wandering and executive attention. *Journal of experimental psychology: general*, *141*(2), 302.

Mrazek, M. D., Smallwood, J., & Schooler, J. W. (2012). Mindfulness and mind-wandering: finding convergence through opposing constructs. *Emotion*, *12*(3), 442-448. doi: 10.1037/a0026678

Peebles, D., & Bothell, D. (2010). Modelling performance in the sustained attention to response task. In *Proceedings of the international conference on cognitive modelling.*

Ratcliff, R. (2002). A diffusion model account of response time and accuracy in a brightness discrimination task: Fitting real data and failing to fit fake but plausible data. *Psychonomic Bulletin & Review*, *9*(2), 278–291.

Smallwood, J., Beach, E., Schooler, J. W., & Handy, T. C. (2008). Going AWOL in the brain: mind wandering reduces cortical analysis of external events. *Journal of Cognitive Neuroscience*, *20*(3), 458–469.

Smallwood, J., Brown, K. S., Tipper, C., Giesbrecht, B., Franklin, M. S., Mrazek, M. D., ... Schooler, J. W. (2011). Pupillometric evidence for the decoupling of attention from perceptual input during offline thought. *PLoS ONE*, *6*(3), e18298. doi: 10.1371/journal.pone.0018298

Smallwood, J., Davies, J. B., Heim, D., Finnigan, F., Sudberry, M., O'Connor, R., & Obonsawin, M. (2004). Subjective experience and the attentional lapse: Task engagement and disengagement during sustained attention. *Consciousness and Cognition*, *13*, 657-690. doi: 10.1016/j.concog.2004.06.003

Smallwood, J., & Schooler, J. W. (2014). The science of mind wandering: Empirically navigating the stream of consciousness. *Annual review of psychology*.

Teasdale, J. D., & Chaskalson, M. (2011). How does mindfulness transform suffering? ii: the transformation of dukkha. *Contemporary Buddhism*, *12*(01), 103–124.

Tulving, E. (2002). Episodic memory: from mind to brain. *Annual Review of Psychology*, *53*, 1-25.

van Vugt, M. K., Hitchcock, P., Shahar, B., & Britton, W. (2012). The effects of MBCT on affective memory associations in depression: measuring recall dynamics in a randomized controlled trial. *Frontiers in Human Neuroscience*, *6*, 257.

VandeKerckhove, J. A., & Tuerlinckx, F. (2007). Fitting the Ratcliff diffusion model to experimental data. *Psychonomic Bulletin & Review*, *14*(6), 1011–1026.

Two Ways to Model the Effects of Sleep Fatigue on Cognition

Christopher L. Dancy (c.l.dancy@gmail.com)

Frank E. Ritter (frank.ritter@psu.edu)
College of IST, Penn State
University Park, PA USA
Glenn Gunzelmann (glenn.gunzelmann@wpafb.af.mil)
Cognitive Models and Agents Branch, Air Force Research Laboratory
Wright-Patterson Air Force Base, OH

Abstract

We compare how the same cognitive model completes a task within two alternative modifications to a cognitive architecture to represent sleep deprivation. One modification (ACT-R/F) has a module that uses a biomathematical model of the effects of sleep deprivation on performance to drive parameter changes in the architecture that impact behavior and performance. The second, new, modification (ACT-R/Φ) represents the effects of sleep deprivation on physiological systems and has these systems modulate cognition. The model completes the psychomotor vigilance task (PVT) within both ACT-R/Φ and ACT-R/F. We found that the two implementations produced similar response times (means) in simulated days one and two. However, the distribution of the response times across the two days of sleep deprivation varied between models. The ACT-R/Φ model shows a wider distribution in both days 1 and 2 due to an increased and modulating production utility noise that affects its ability to select the correct rules consistently. Though they represent sleep deprivation in different ways, and on different levels, both of these implementations lead us towards a more unified understanding of how sleep deprivation affects our bodies, how we think and behave over time, and how to represent these effects.

Keywords: ACT-R, sleep deprivation, behavioral moderators, HumMod.

Introduction

Extensive empirical research has demonstrated that performance varies in systematic ways over time as a result of time awake, time on task, circadian rhythms, and a variety of other factors that impact the effectiveness and efficiency of cognitive processing (e.g., Gluck & Gunzelmann, 2013). Despite the obvious importance of these factors, collectively referred to as cognitive moderators (e.g., Ritter et al., 2007; Ritter et al., 2003; Silverman et al., 2006), their roles in human cognition, are rarely considered in cognitive science research. Instead, nearly all computational and mathematical models in the literature treat the cognitive system as an optimally functioning information processing machine, which does not waver in its performance over seconds, minutes, hours, or days of performance. As increasingly sophisticated models of various cognitive processes are developed, it is critical to improve the fidelity of moderating functions to capture human performance across the broad range of situations being modeled.

This research focuses on one of these factors—the impact of fatigue brought on by extended time awake. Sleep and circadian rhythms are features of nearly all life on earth, yet their function and impact on cognitive functioning and performance remain poorly understood and infrequently modeled. We examine two ways to model these features.

ACT-R/F

In recent years, some research using computational modeling has begun to expose how human information processing is impacted by fatigue and related factors (e.g., Gunzelmann et al., 2009). This research manipulates parameters in a cognitive architecture, ACT-R, to capture performance changes associated with time awake and circadian rhythms (e.g., Gunzelmann et al., 2012; Gunzelmann et al., 2009), as well as time on task (e.g.,Gartenberg et al., 2014; Gunzelmann et al., 2010). At a theoretical level, the approach integrates a theory of the dynamics of alertness into the ACT-R architecture, creating the ACT-R/F (ACT-R/Fatigue) system. At its core, it demonstrates how fluctuations in alertness can influence performance by impacting the functioning of information processing mechanisms within the cognitive system.

The primary component of the theory is a mechanism associated with fatigue that disrupts ongoing cognitive processing. In the model, the disruptions are implemented as micro lapses, which are small gaps in the information processing in central cognition. These gaps lead to small delays in performance (10's of ms). However, their probability increases with fatigue, which can lead to substantial impairments in performance. In conjunction with this mechanism, there is a compensation mechanism that reduces the likelihood of microlapses, but also increases the likelihood of executing inappropriate, or less useful, cognitive actions.

The mechanisms are implemented in ACT-R's central cognitive module, a production system that coordinates the activity of the other modules to maintain goal-directed cognitive activity (Anderson, 2007). This system operates in cycles, each lasting about 50 ms each. On each cycle, where appropriate actions are identified, one is selected based on a utility calculation, and then the action is taken, provided the utility surpasses the utility threshold. Microlapses occur when the threshold is not reached. In the traditional version

of ACT-R, the model run is terminated when actions fail to exceed the utility threshold. When this situation occurs in ACT-R/F, the cognitive cycle is "skipped," producing a gap of about 50 ms in the goal-directed processing of the model. Because there is noise in the utility calculation mechanism, it is possible that a production will exceed the utility threshold on a subsequent cycle. Compensation is represented in the architecture by reducing the utility threshold. Although this decreases the likelihood of a microlapse, it also increases the probability that actions with a lower utility will be selected and executed in the model.

To control the dynamics associated with fluctuations in alertness, a biomathematical model of alertness was integrated into ACT-R as a new module. The biomathematical model is described in detail in McCauley et al. (2013). Generally, the McCauley model accounts for changes in overall cognitive functioning stemming from time awake and circadian rhythms, incorporating the two-process theory of alertness (Achermann & Borbély, 1992).

The biomathematical model produces a numerical estimate of fatigue. The function of the module is to connect the numerical output of the biomathematical model to parameters in the ACT-R architecture related to the information processing mechanisms that are hypothesized to be affected by fatigue. For instance, within central cognition biomathematical model outputs, F, influence the utilities of candidate actions and the utility threshold. This is achieved by computing scaling factors, FP and FT, for the utility of productions and the threshold as follows:

Eq. 1 $FP = 1 - a_{FP}F$
Eq. 2 $FT = 1 - a_{FT}F$

In these equations, a_{FP} and a_{FT} are parameters that define the slope of a linear mapping of fatigue values to the utility and threshold scalars, respectively. FP and FT are constrained to be between 0 and 1. The scaling factors, in turn, influence the utility values and threshold in ACT-R:

Eq. 3 $U_i' = a_{FP}FU_i$
Eq. 4 $T_i' = a_{FP}FT_i$

Here, Ui' is the computed utility value for production (i), and T_i' is the utility threshold used to determine if the selected production is executed. These mechanisms have been demonstrated to capture in detail changes in human performance on a sustained attention task with sleep deprivation. The model predicts the response time distribution of individual participants, at a level of precision that is equivalent to the detail provided by a diffusion model of the same task (Walsh et al., 2014).

ACT-R/Φ

As more models of cognition and information processing moderators are developed, it will also be important to find a way to tie these separate models together. However, understanding the interactions between moderators can be difficult as the models are often developed in isolation.

One way to make the modeling of the interactions between moderators and the effects of these interactions on information processing more straightforward and tractable is to model these effects on the physiological, as well as the cognitive, level. Common physiological systems involved in changes in cognitive mechanisms can be used as a basis for understanding the interactions between moderators.

The ACT-R/Φ architecture (Dancy et al., In Press) combines a cognitive architecture (ACT-R) and an integrative computational model of physiology (HumMod; Hester et al., 2011) so that the bidirectional connections between physiological and cognitive systems can be simulated. HumMod is a computational modeling and simulation system that provides an integrative computational model of human physiology, to simulate the interaction between physiological, affective, and cognitive change. The ACT-R/Φ architecture has been used to model the dynamic effects of physiological change due to a psychological stressor (Dancy et al., In Press) and due to affective thirst (Dancy & Kaulakis, 2013). These moderators affect some of the same basic cognitive mechanisms in the architecturesImportantly, because these models have been developed within a single unified architecture, their interactions can also be modeled.

The stress-related pathways between physiological and cognitive processes in ACT-R/Φ are also important for modeling the effects of sleep loss due to the involvement of the Locus Coeruleus (LC) System and Hypothalamic-Pituitary-Adrenal (HPA) axis, which are important in circadian components of sleep (Saper et al., 2005).

The ACT-R/Φ architecture uses variables from the physiological (using HumMod) and affective systems (using theory from affective neuroscience and emotion research) to determine a level of memory-based arousal. Arousal is determined using cortisol, epinephrine, corticotrophin releasing hormone (CRH), and a FEAR value (e.g., Panksepp et al., 2011) as shown in equation 5.

Eq. 5 $Arousal = f(cort) * [\alpha * g(crh) + \beta * h(epi)]$

The equation reflects evidence that cortisol seems to serve more of a multiplicative than additive role in memory-based arousal due to the LC system (e.g., Roozendaal & McGaugh, 2011; Roozendaal et al., 2006). In Equation 5, α and β are parameters that determine the slope of the linear relation between deviation from the normal physiological state; $f(cort)$, $g(crh)$, and $h(epi)$ is a function of the change in cortisol, CRH, and epinephrine (respectively) from the baseline state.

The systems involved in stress and arousal (e.g., the LC system and the HPA axis) have also been shown to modulate both declarative and procedural memory (e.g., see Sara & Bouret, 2012; Schwabe & Wolf, 2013). Thus, this arousal factor affects both declarative and procedural memory in the ACT-R/Φ architecture by affecting related noise parameters,

that is, $:ans$ (declarative memory noise) and $:egs$ (procedural memory utility noise) are both modulated using Equation 6 (A stands for *Arousal*).

Eq. 6 $$noise = \begin{cases} \frac{1-A}{0.5} - 1 \; \forall \; A \leq 0.5 \\ \frac{A}{0.5} - 1 \; \forall \; A > 0.5 \end{cases}$$

Both low arousal (below a nominal value) and high arousal cause an increase in noise, making it more difficult to retrieve chunks (declarative memory) and to select the correct productions (procedural memory). In addition to its effects on procedural memory noise, *arousal* also modulates utility threshold of matched rules when it goes below the nominal arousal value. We chose to alter both noise and threshold in this case because of existing evidence that as neural arousal decreases below basal values (as measured by activity in the LC-system), distractibility tends to increase (e.g., Aston-Jones & Cohen, 2005). One way to interpret this result is that decision utilities are more affected by a noise as neural arousal lowers.

Implementing Biomathematical Models of Fatigue in HumMod

We implemented a mathematical model of the HPA-axis in HumMod to simulate circadian and sleep homeostatic changes in adrenocorticotropic hormone (ACTH) and cortisol. We modified the effect of CRH on ACTH so that ACTH levels show circadian fluctuations; this causes related circadian fluctuations in cortisol levels. A sleep homeostatic variable was also added that directly affects cortisol. This variable represents the direct modulatory effects the SCN can have on cortisol outside of the HPA-axis (e.g., see Saper et al., 2005).

The arousal representation was modified (Equation 7a) to include a neural sleep homeostatic variable that decreases (and has an accelerating decline) as time awake increases. H_{S_N} is a neural sleep homeostatic variable that causes arousal to decrease as the time awake increases. As with equation 5, the parameters α and β are parameters that determine the slope of the linear relation between deviation from the normal physiological state.

Eq. 7

A $$Arousal = H_{S\_N} * f(cort) * [\alpha * g(crh) + \beta * h(epi)]$$

B $$cort = H_{S\_C} + Secretion_{Constant} * ACTHEffect + Degradation$$

C $$H_{S\_C} = \begin{cases} SA_{Mag} * \left(1 - {SA_{Rate}}^{T_S}\right) \; [while \; asleep] \\ WA_{Mag} * \left(1 - {WA_{Rate}}^{Tw}\right) \; [while \; awake] \end{cases}$$

D $$ACTH = Secretion_{constant} * CRHEffect * \sum_{i=1}^{4}\left[\rho_i * \left(Sin\left[\frac{i*\pi*t}{720} - \theta\right]\right)\right] + 1$$

Cortisol (Equation 7-B) fluctuates over the course of the day due to circadian rhythms and a sleep homeostatic parameter (Equation 7-C). ACTH (Equation 7-D) has circadian fluctuations, and this variable directly modulates cortisol secretion via the *ACTHEffect* variable, though the proportion of cortisol secretion that is caused due to ACTH varies by time of day and sleep-wake transition time[1]. In this equation, t (for current time of the day) is represented at minutes.

These equations in the HumMod physiological model create a fluctuating HPA-axis, governed by time of day (assuming a stable entrained normal sleep and wake time) and homeostatic pressure created by time spent asleep or awake. Figure 1 shows changing cortisol levels over the course of two days in the updated model. The model displays a peak in cortisol levels at the point of waking (6am in this case) and a trough at 12am.

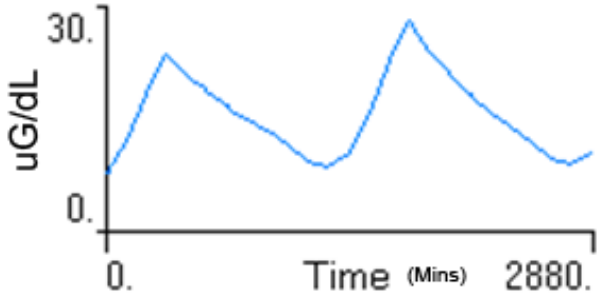


Figure 1. Cortisol levels over the course of two days ([uG/dL]/minutes).

If we cause the model to go two days without sleeping, cortisol in HumMod shows a different, but still circadian, rhythm (Figure 2). The peaks and troughs are roughly at the same positions, but Figure 1 shows a higher minimum and maximum that occur near sleep-wake transitions. The cortisol profile of the sleep deprived model shows a steady increase in peak and trough across days as sleep deprivation time increases.

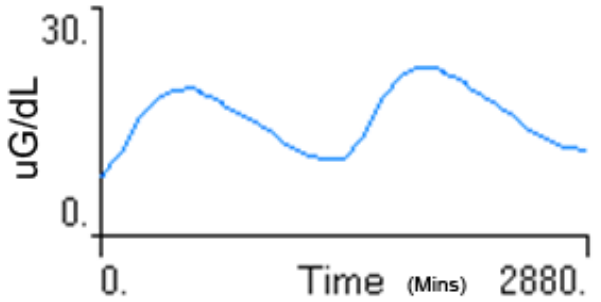


Figure 2. Cortisol levels over the course of two days with the model sleep deprived ([uG/dL]/minutes).

ACT-R/F, ACT-R/Φ, and the PVT

To get a further understanding of how model behavior may change when using these alternative implementations, we implemented a model of the psychomotor vigilance task

[1] Equations 7-C & -D are modified from Thorsley et al. (2012)

(PVT) within both ACT-R/F and ACT-R/Φ. In the task, a millisecond counter is presented at the center of a monitor at a random delay of 2-10 seconds from the previous response. The task is to respond to the appearance of the counter by pressing a response button.

The pervasive use of this task in sleep research makes it an important task for theories of fatigue to address. In addition, the subtle changes to response time distributions of fatigued individuals in the task imposes a critical test of the capacity of a computational theory to make detailed, quantitative predictions about human performance.

The model includes three rules—one each for waiting, attending to a stimulus, and responding to the stimulus. Partial-matching is enabled in the model to allow rules that match some, but not necessarily all, of the rule conditions. Thus, when the rules are affected more by noise, whether by increasing the actual noise (ACT-R/Φ) or lowering the utility values of all of the rules (ACT-R/F), false starts (responding before a stimulus is presented) can occur.

Overall means (and std. dev.) of response times were similar between models for both days 1 and 2: 237.4 (14.84) and 235.1 (15.26) for the ACT-R/ Φ model, and 238.3 (12.85) and 238.4 (11.48) for the ACT-R/F model. Despite the overall similarity, the distributions of means for days 1 and 2 differed between models (Figure 3 and 4).

Figure 4 shows slightly different mean response time distribution between days 1 and 2 with day 1 showing a more uniform density distribution. The increased noise due to physiological change in the ACT-R/Φ implementation caused a wider distribution of response times as compared to the ACT-R/F implementation.

Discussion and Conclusions

Both of the implementations discussed provide a novel way of modeling and simulating the effects of sleep deprivation on cognition, albeit in different ways and on different levels of representation. The ACT-R/F implementation takes a tested biomathematical performance model and applies it to procedural memory in the ACT-R architecture (see also Gunzelmann et al., 2012). Implementing sleep deprivation in ACT-R/Φ required adding circadian rhythms and sleep homeostatic modulation to physiological variables and having these variables modulate cognitive systems.

Comparison of the two architectures

We found that there are similarities and differences between the two approaches. Both approaches include aspects of sleep behavior, and the resulting predictions are similar in how they predict that there are increases and decreases in performance across a day.

They are different in the initial quality of their predictions and their extendibility. ACT-R/F is more accurate in its predictions. ACT-R/Φ is more extendable, in that it would be very feasible to represent in a plausible way how other factors will interact with sleep, such as caffeine.

Including HumMod in ACT-R/Φ raises the question of usability, however. HumMod, while a useful system, is another large system that has to be run and understood. It

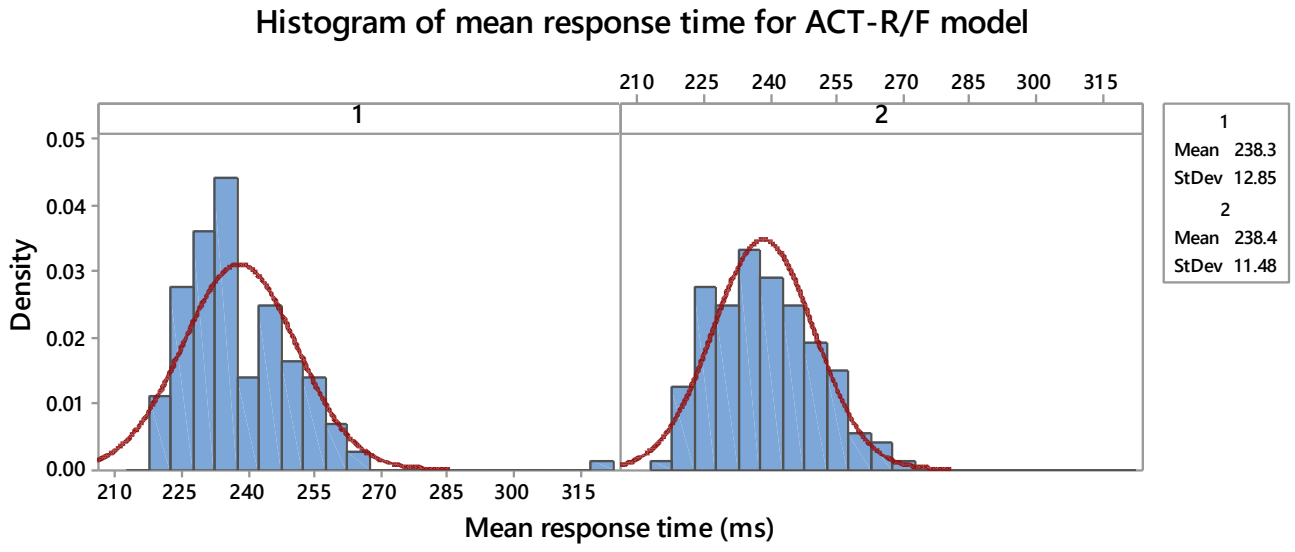


Figure 3. Mean response times of the PVT model used in the ACT-R/F extension. Distribution density shape of show a different pattern between day 1 and 2 than the model using ACT-R/Φ extensions (Figure 4).

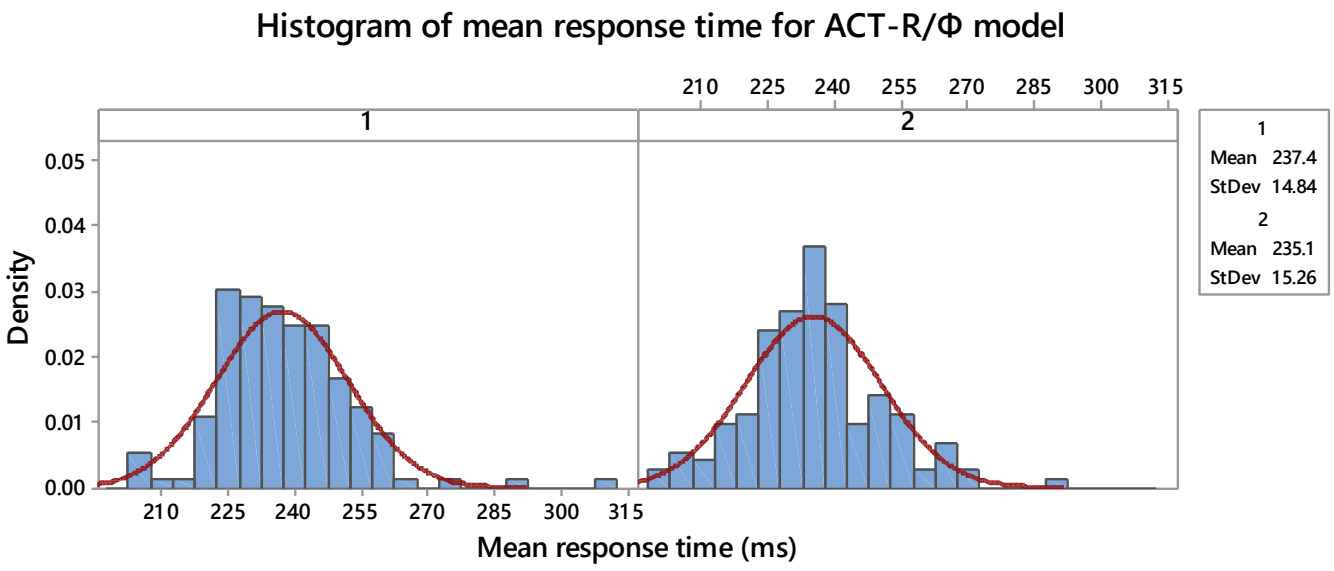


Figure 4. Mean response times of the PVT model used in the ACT-R/ Φ extensions. Distribution density shape of mean values changes between day 1 and day 2.

takes ACT-R/Φ longer to run, and it takes a little more interpretation. Future work will need to explore potential software optimization methods and implementations that can be used with high performance computing (e.g., Harris et al., 2009). It will also be useful to explore possible combinations of the approaches as some mathematical variables in the performance model used in ACT-R/F have been connected to neural representations (McCauley et al., 2013). At this point, we are not able to make recommendations about which is better, but the two approaches are at least different.

This work will also raise new problems about understanding architectures. The combined architecture, that is, ACT-R/Φ, will have further variables and will require further validation, crossing between cognitive psychology and physiology. This will raise new challenges.

Future Work

We will expand upon the new ACT-R/Φ implementation by performing parameter sweeps so that the most realistic model predictions can be found. The performance costs inherent in running the HumMod and expanded ACT-R systems in tandem means that using existing work and theory related to the parameters/variables for performance optimization will be especially important. In addition to continuing to solidify and validate components of these implementations, there are two particular research directions in which this work can expanded: the effects of caffeine on cognition and interactions between sleep deprivation and stress.

As caffeine continues to play a significant role in modern society, it will be important to have a quantifiable understanding of its modulation of cognitive performance over time and to have the same understanding of the ways time of day may interact with this modulation. More recent work in modeling the effects of caffeine on vigilance (Ramakrishnan et al., 2014) and on declarative memory (e.g., Ritter et al., 2009) provide a useful roadmap for continued expansion of the work presented here. Working within HumMod to represent the effects of caffeine on physiology and then and thus on cognition provides a principled way to combine the effects of moderators.

There also exist several parallels between work on sleep deprivation and work on stress systems. It has even been suggested that sleep deprivation can be seen as a form of stress, causing allostatic physiological and behavioral change (McEwen, 2006). Many of the neural systems implicated in the behavioral change due to sleep deprivation and stress systems overlap and are influenced by one another (e.g., the LC system, basal ganglia, and hippocampus). Thus, the generalization of these implementations to the study of stress would be a fairly natural evolution of the work.

Summary

As we continue to study the ways behavioral moderators affect the way we think, feel, and perform during daily activities, it will be vital to keep in mind the effects of these moderators across time, and during different parts of the day. In addition, it will be important that the models and architectures we develop to describe and predict these behavior are generalizable and can be understood in the context of separate, but related cognitive, affective, and physiological behavior. Both of these implementations lead us towards a more unified understanding of how sleep deprivation affects our bodies, as well as the way we think and behave over time.

Acknowledgments

This was funded by the Air Force Research Laboratory's 711 Human Performance Wing through a contract from L3. The opinions and assertions contained herein are the personal views of the authors and are not to be construed as official or as reflecting the views of the US Air Force, or the US Department of Defense.

References

Achermann, P., & Borbély, A. A. (1992). Combining different models of sleep regulation. *Journal of Sleep Research, 1*(2), 144-147.

Anderson, J. R. (2007). *How can the human mind occur in the physical universe?* New York, NY: OUP.

Aston-Jones, G., & Cohen, J. D. (2005). An integrative theory of locus coeruleus-norepinephrine function: Adaptive gain and optimal performance. *Annual Review of Neuroscience, 28*(1), 403-450.

Dancy, C. L., & Kaulakis, R. (2013). Towards adding bottom-up homeostatic affect to ACT-R. In *Proceedings of the 12th International Conference on Cognitive Modeling*, pp. 316-321. Ottawa, Canada.

Dancy, C. L., Ritter, F. E., Berry, K., & Klein, L. C. (In Press). Using a cognitive architecture with a physiological substrate to represent effects of psychological stress on cognition. *Computational and Mathematical Organization Theory*.

Gartenberg, D., Veksler, B. Z., Gunzelmann, G., & Trafton, J. (2014, September 1, 2014). An ACT-R process model of the signal duration phenomenon of vigilance. In *Proceedings of the Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, pp. 909-913. Thousand Oaks, CA.

Gunzelmann, G., Gluck, K. A., Moore, L. R., & Dinges, D. F. (2012). Diminished access to declarative knowledge with sleep deprivation. *Cognitive Systems Research, 13*(1), 1-11.

Gunzelmann, G., Gross, J. B., Gluck, K. A., & Dinges, D. F. (2009). Sleep deprivation and sustained attention performance: Integrating mathematical and cognitive modeling. *Cognitive Science, 33*(5), 880-910.

Gunzelmann, G., Moore, L. R., Gluck, K., Van Dongen, H. P. A., & Dinges, D. F. (2010). Fatigue in sustained attention: Generalizing mechanisms for time awake to time on task. In P. L. Ackerman (Ed.), *Cognitive fatigue: Multidisciplinary perspectives on current research and future applications* (pp. 83-96). Washington, DC: American Psychological Association.

Harris, J., Gluck, K. A., T., M., & Moore Jr, L. (2009). MindModeling@Home ...and anywhere else you have idle processors. In *Proceedings of the 9th International Conference of Cognitive Modeling*, Manchester, United Kingdom.

Hester, R. L., Brown, A. J., Husband, L., Iliescu, R., Pruett, D., Summers, R., & Coleman, T. G. (2011). HumMod: A modeling environment for the simulation of integrative human physiology. *Frontiers in Physiology, 2*(12).

McCauley, P., Kalachev, L. V., Mollicone, D. J., Banks, S., Dinges, D. F., & Van Dongen, H. P. A. (2013). Dynamic circadian modulation in a biomathematical model for the effects of sleep and sleep loss on waking neurobehavioral performance. *Sleep, 36*(12), 1987-1997.

McEwen, B. S. (2006). Sleep deprivation as a neurobiologic and physiologic stressor: Allostasis and allostatic load. *Metabolism - Clinical and Experimental, 55*, S20-S23.

Panksepp, J., Fuchs, T., & Iacobucci, P. (2011). The basic neuroscience of emotional experiences in mammals: The case of subcortical FEAR circuitry and implications for clinical anxiety. *Applied Animal Behaviour Science, 129*(1), 1-17.

Ramakrishnan, S., Laxminarayan, S., Wesensten, N. J., Kamimori, G. H., Balkin, T. J., & Reifman, J. (2014). Dose-dependent model of caffeine effects on human vigilance during total sleep deprivation. *Journal of Theoretical Biology, 358*(0), 11-24.

Ritter, F. E., Kase, S. E., Klein, L. C., Bennett, J., & Schoelles, M. (2009). Fitting a model to behavior tells us what changes cognitively when under stress and with caffeine. In *Proceedings of the the Biologically Inspired Cognitive Architectures Symposium at the AAAI Fall Symposium Series. Keynote presentation*, pp. 109-115. Washington, DC.

Ritter, F. E., Reifers, A. L., Klein, L. C., & Schoelles, M. J. (2007). Lessons from defining theories of stress for cognitive architectures. In W. D. Gray (Ed.), *Integrated Models of Cognitive Systems* (pp. 254-262). New York, NY: OUP.

Ritter, F. E., Shadbolt, N. R., Elliman, D., Young, R. M., Gobet, F., & Baxter, G. D. (2003). *Techniques for modeling human performance in synthetic environments: A supplementary review*. DTIC Document. Wright-Patterson Air Force Base, OH: Human Systems Information Analysis Center.

Roozendaal, B., & McGaugh, J. L. (2011). Memory modulation. *Behavioral Neuroscience, 125*(6), 797-824.

Roozendaal, B., Okuda, S., Van der Zee, E. A., & McGaugh, J. L. (2006). Glucocorticoid enhancement of memory requires arousal-induced noradrenergic activation in the basolateral amygdala. *Proceedings of the National Academy of Sciences, 103*(17), 6741-6746.

Saper, C. B., Scammell, T. E., & Lu, J. (2005). Hypothalamic regulation of sleep and circadian rhythms. [10.1038/nature04284]. *Nature, 437*(7063), 1257-1263.

Sara, S. J., & Bouret, S. (2012). Orienting and reorienting: The locus coeruleus mediates cognition through arousal. *Neuron, 76*(1), 130-141.

Schwabe, L., & Wolf, O. T. (2013). Stress and multiple memory systems: From 'thinking' to 'doing'. *Trends in Cognitive Sciences, 17*(2), 60-68.

Silverman, B. G., Johns, M., Cornwell, J., & O'Brien, K. (2006). Human behavior models for agents in simulators and games: part I: enabling science with PMFserv. *Presence: Teleoperators & Virtual Environments, 15*(2), 139-162.

Thorsley, D., Leproult, R., Spiegel, K., & Reifman, J. (2012). A phenomenological model for circadian and sleep allostatic modulation of plasma cortisol concentration. [10.1152/ajpendo.00271.2012]. *American Journal of Physiology - Endocrinology and Metabolism, 303*(10), E1190-E1201.

Walsh, M. M., Gunzelmann, G., & Van Dongen, H. P. A. (2014). Comparing accounts of psychomotor vigilance impairment due to sleep loss. In *Proceedings of the 36th Annual Conference of the Cognitive Science Society*, pp. 877-882. Austin, TX.

A Model of Distraction using new Architectural Mechanisms to Manage Multiple Goals

Niels A. Taatgen (n.a.taatgen@rug.nl), Ioanna Katidioti (i.katidioti@rug.nl), Jelmer Borst (j.p.borst@rug.nl) & Marieke van Vugt (m.k.van.vugt@rug.nl)
Institute of Artificial Intelligence, University of Groningen,
Nijenborgh 9, 9747 AG Groningen, Netherlands

Abstract

Cognitive models assume a one-to-one correspondence between task and goals. We argue that modeling a task by combining multiple goals has several advantages: a task can be constructed from components that are reused from other tasks, and it enables modeling thought processes that compete with or support regular task performance. To achieve this, we updated the PRIMs architecture (a derivative of ACT-R) with the capacity for parallel goals that have different activation levels. We use this extension to model visual distraction in two experiments. The model provides explanations for the finding that distraction increases with task difficulty in a memory task, but decreases with task difficulty in a visual search task.

Keywords: Cognitive Control, Task representation, PRIMs, Distraction, Multitasking

Introduction

Whenever we are faced with something new to learn or to do, we can rely on a vast array of skills and knowledge. Given what we usually call a *task*, we need to recruit the right procedural and declarative knowledge and find the best way to piece this knowledge together, and, if necessary expand it with the missing pieces. One might think this challenge should be a centerpiece in the cognitive modeling and cognitive architecture research tradition, but unfortunately it isn't.

In almost all flavors of cognitive modeling, whether symbolic, hybrid or connectionist, it is tacitly assumed that there is a one-to-one relationship between tasks and goals. In this context, we consider a *goal* to be an internal representation that is used to recruit the appropriate knowledge to achieve that goal. Most models model just use a single task and a single goal, and all the knowledge incorporated in or acquired by the model is just for that task.

The one-task-one-goal approach puts many restrictions on what can be achieved by cognitive modeling. It ignores the question how goals are set, prioritized and abandoned. It cannot answers questions about what *other* things people are thinking about when they carry out a task, which might affect the task either positively or negatively. For instance, metacognitive planning ahead may have a positive effect on performance, whereas distraction may have a negative effect. But distraction may be a positive influence if it prevents us from pursuing a hopeless goal.

An alternative for the one-task-one-goal approach is to have several active goals to support a single task. The traditional method of doing this is through subgoaling. In particular the Soar cognitive architecture has pursued the idea that a new task or goal can use several subgoals that have already been learned as part of other tasks (Laird, Rosenbloom, & Newell, 1986). Unfortunately, in most Soar models subgoals were specifically designed for a specific main goal. Moreover, the goal stack is now considered by many as a too rigid representation, because typically only a single goal in the hierarchy is active (Anderson & Douglass, 2001), while in reality goals typically compete with each other (i.e. calling while driving).

An alternative for a goal hierarchy is to have several goals active at the same time, as for example implemented in the threaded cognition extension to ACT-R (Salvucci & Taatgen, 2008). Threaded cognition, however, has mainly been used to model multi-tasking, so although a model would have multiple tasks and multiple goals, there was still a one-to-one mapping between tasks and goals (with some exceptions, e.g., Taatgen, Juvina, Schipper, Borst, & Martens, 2009).

Another effort to break the monolithic goal structure is the PRIMs (Primitive information processing elements) theory, another extension to ACT-R (Taatgen, 2013). PRIMS allows us to go beyond the original tasks by breaking down task-specific rules into combinations of primitive information processing elements that in themselves are task-general. Learning a new task involves combining those primitive elements into task-specific rules, but the byproduct of the learning trajectory is that the model also learns task-general rules that it can use for other purposes. This means that PRIMs can model knowledge transfer from one task to another. A limitation of PRIMs is that task knowledge is still specified in terms of task-specific operators that are linked to a single goal.

Altmann and Gray (2008) explore a different aspect of goals: the current goal is not set by production actions, as is the case in most models, but the goal with the highest activation determines the actions. Their goal representations are susceptible to decay, and therefore the reaction times for a particular goal gradually increase as subjects continue doing the same task. Rehearsal processes can influence goal activations, which is the primary process to control what the current goal is. Still, at any moment a single goal is active for a single task.

In this paper we will combine these three approaches as part of a new version of PRIMs (PRIMs 2.0), in which a single task is implemented by multiple goals. These goals are specified in such a way that they are can be reused for

other tasks, and have associated activation levels that determine which goal is most influential at a certain moment. We will then use this to build a model of distraction. Distraction, or self-interruption, is a major problem in our information society, because regular work progress is threatened by email-checking and Facebook updating. It is therefore of importance what factors influence self-interruption, which may enable us to find ways to mitigate or control it. First, we will explain the new version of PRIMs in detail.

The PRIMs 2.0 goal representation

As an example to illustrate the goal representation we will use part of a task that we will use later on: solving simple equations. The task of solving an equation like $5x + 2 = 12$ is represented by four parallel goals: reading the equation into working memory, transforming the equation, doing arithmetic, and giving the answer (Figure 1). The four goals are not carried out in parallel, of course, but their representations are all active. Active goals spread activation to *operators* in memory that can carry out that goal. Operators are the declarative counterpart of production rules, so they have conditions that are tested, and actions that are carried out when the conditions are satisfied (see Taatgen, 2013, for details). When solving an equation, first operators are retrieved that are associated with the reading goal, because there is no mental representation of the equation yet. Once there is a representation, operators associated with the transformation and arithmetic goals alternate in solving the equation, until the answer goal can key in the answer.

In this example all four goals are active throughout problem solving, and the conditions of the operators ensure that they are carried out in the right order. This is not always possible, so sometimes goals have to be added or removed. However, this requires a particular control strategy, which makes learning harder.

Ultimately, goals only influence the activation of operators. This means that a goal doesn't guarantee a matching operator is selected, it only makes it more likely. Other factors can influence the retrieval of operators, though, for example external stimuli or the content of particular declarative retrievals. It is therefore possible that an operator is retrieved that has nothing to do with the current goals, leading to distractions. As we will see later on, it is in between operators for different goals that distractions have an opportunity to intervene.

Key Principles

The key principles of the PRIMs 2.0 goal representation are as follows:

Goals are carried out by operators that are associated with that goal. This is of course true for almost any symbolic architecture, but uniquely in PRIMs multiple goals can be active, and operators can also be associated with multiple goals. Operators do not refer directly to goals or vice versa, there are connected through strength of association only.

Goals have an activation value that is susceptible to all ACT-R memory processes. As a consequence, not all goals are equal, and the goal with the highest activation has a higher probability of recruiting operators it needs.

The activation value of a goal determines how much activation it spreads. All active goals are stored in slots in ACT-R's goal buffer. This means that they are sources of spreading activation. However, instead of the standard spreading activation of W/n, as is in regular ACT-R, the W_j of a goal is equal to its activation. This means that operators that are associated with a certain goal receive spreading activation proportional to the activation of that goal.

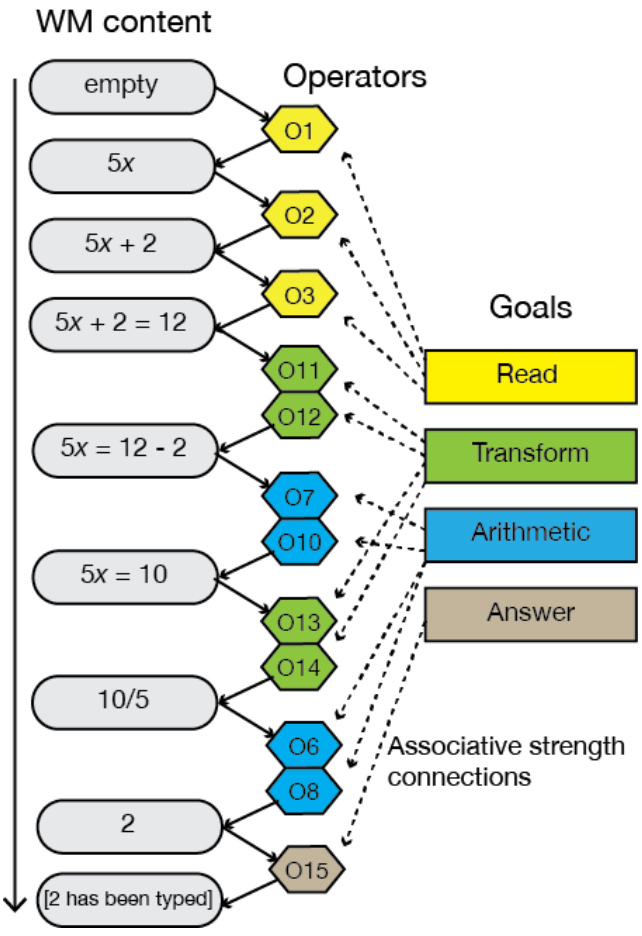


Figure 1. Illustration of how operators associated with different goals together solve an equation

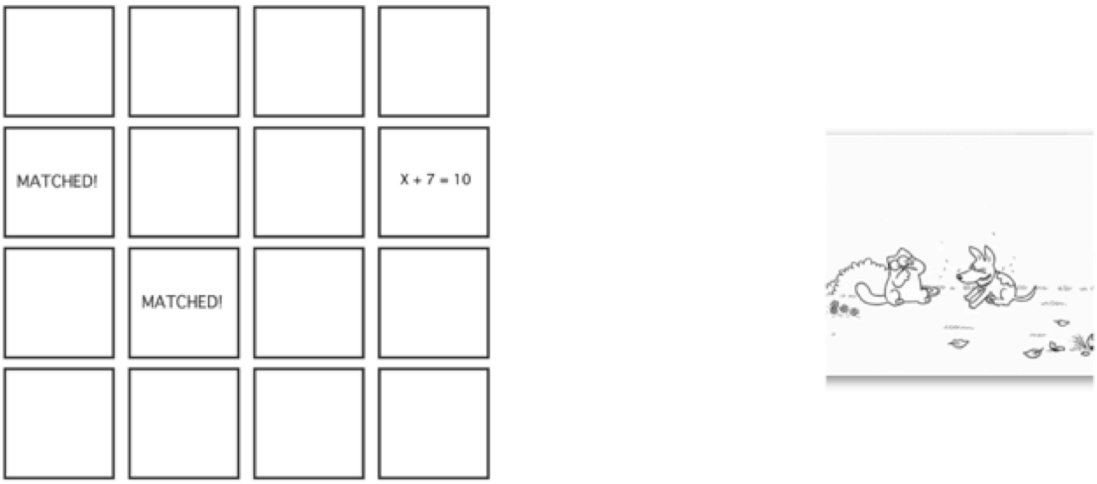


Figure 2. Example of the memory game on the left, and the movie on the right. The memory game is in the medium condition, and two of the cards have already been matched.

The most active operator *whose conditions are satisfied* is carried out. This is not necessarily an operator for the most active goal, because it may have no operators that currently match. Moreover, other influences can add activation to operators to influence the selection, in particular spreading activation from other sources (perception, working memory, memory retrieval, etc.).

Activation of a goal can be increased by explicitly retrieving it (possibly repeatedly). Retrieval is a deliberate strategy to influence the priority of goals. Increased influence can be achieved by multiple retrievals (rehearsal, cf. Altmann & Trafton, 2002).

Activation of a goal decays over time. This means that if goals aren't maintained, or reinforced in any other way, they decay and disappear.

Operators associated with the same goal are also associated with each other. This makes it more likely that an operator for the same goal as the previous operator is chosen.

To demonstrate the power of this approach, we will use it to model a distraction experiment.

Experiment

The main idea of the experiment is that subjects had to carry out different tasks of varying difficulty level. While they carried out the task, a video played at the other side of the screen. The video was unrelated to the task. The extent to which subjects in the experiment were distracted by the video was measured with an eye tracker.

The experiment involved two different tasks, one focusing on mental operations, and the other on visual operations. Each had three different levels of difficulty. In the *memory game*, subjects played the game of *Memory* or *Concentration* with cards with equations instead of pictures. Sixteen cards were displayed on the screen. Subjects had to click on the cards, which would reveal the equation on the back. Clicking on two consecutive cards with the same solution to the equation would remove them, with the eventual goal of removing all sixteen cards (Katidioti, Borst, & Taatgen, 2014). There were three levels of difficulty: the easiest level had equations of the form $4 + 2 = x$, basically simple arithmetic. Medium level equations were of the form $x + 4 = 16$, requiring a transformation followed by arithmetic, and hard question had the form $5x + 2 = 12$, as illustrated in Fig. 1, requiring several operations to solve. Figure 2 shows a screen-shot of the experiment.

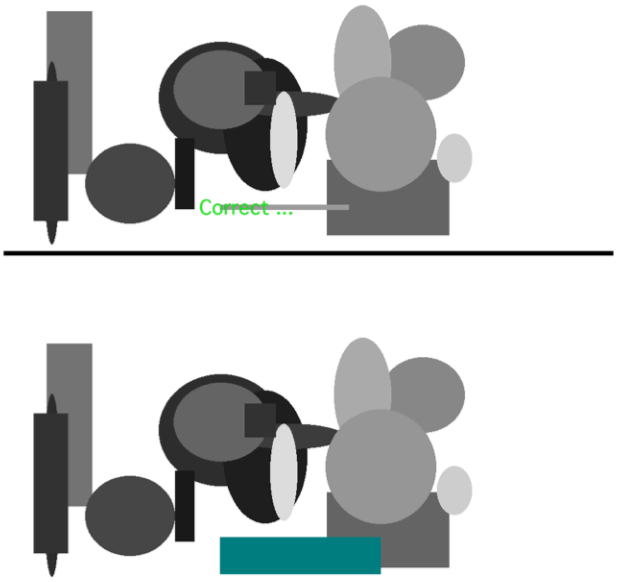


Figure 3. Example of the find-the-differences game in the medium condition, just after feedback.

In *find-the-differences game*, subjects were presented with two pictures on the top and bottom of the screen that each consisted of a number of random shapes (colored ovals and rectangles). Both pictures were identical except for one

small difference in one of the shapes (Figure 3). The task was to find the difference, and click on it. In the easy condition, each picture consisted of 2-4 shapes, in the medium condition 15-17 shapes, and in the hard condition 40-42 shapes. Like in the memory game, a movie that was unrelated to the task also played at the other side of the screen.

Subjects in the experiment either did the find-the-differences game or the memory game (25 participants per task). They performed one of the tasks for 15 minutes at each level of difficulty, for a total of 45 minutes.

Our theory about choice in multitasking is that people tend to switch to another task if that task needs resources that are not currently used by the present task. In the experiment, the distraction requires use of the visual resource, so the prediction is that if the main task needs fewer visual resources, the frequency of distractions by the video will be higher. We therefore predict that the frequency of distraction increases with difficulty for the memory game, because solving an equation temporarily requires fewer visual resources, which are then free to move to the video. In contrast, distraction decreases with difficulty for the find-the-differences game, because visual resources are more occupied by the more difficult games.

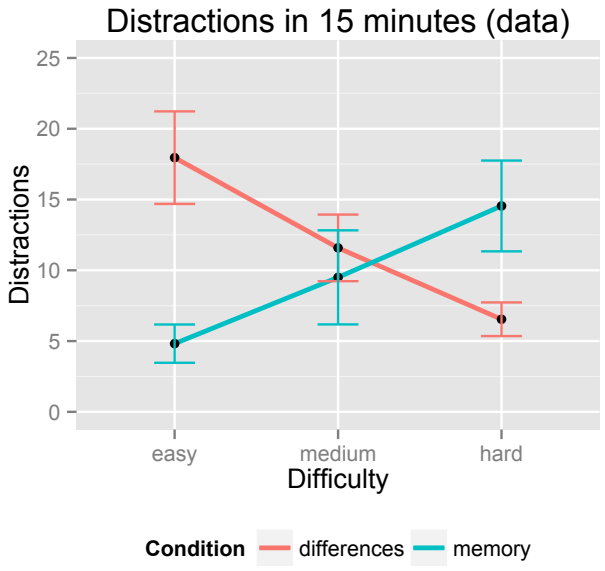


Figure 4. Results of the experiment. Error bars represent 1 standard error.

Figure 4 shows the mean number of distractions (i.e., eye-movements to the video) per subject in each of the 15 minutes blocks of playing either game (distractions between games were removed). The effect of the level of difficulty on the number of distractions is highly significant: when we fit generalized linear mixed effect models based on a Poisson distribution to the data, all differences between different levels of difficulty are significant with p values less than 0.001.

To conclude, the experimental results support the theory that distraction increases if the resources that are available match the resources that the distraction requires (in this case visual resources). We have similar, although weaker, evidence that this is also the case for working memory, where subjects tended to switch to a secondary task involving memory more often at moments that the memory requirements of the main task had just decreased (Katidioti et al., 2014). The next challenge is to construct a model that reproduces this behavior.

A Model of Distraction

In the new PRIMs 2.0 representation, tasks are represented by multiple goals. However, these goals only spread activation to applicable operators, so they do not enforce that only task-relevant operators are chosen. This means that at any point an operator can be retrieved that is not related to the task, assuming that operator has a high enough activation. In our case, the video is a perceptual input that is associated with operators that propose to attend the video. A condition of these operators is that the visual resource is available.

A model of the memory game

For simplicity, we have not constructed a model that plays the whole game, but a model that just solves equations of varying difficulty. We think that this partial model captures the essential characteristics of the task as a whole. Figure 1 already gave a clear representation of the structure of the model: the task is represented by four goals, read, transform, arithmetic and answer. Each of these goals has a small set of associated operators that implement that goal. The hard equations require the sequence as it is shown in Figure 1, involving 12 operators. The medium and easy equations use the same operators, but omit some of them: the medium equations skip the second transform and arithmetic sequence, and therefore only require 8 operators. The easy equations require no transformations, just the final arithmetic, and one fewer read operator, so a total of 5 operators.

In addition to the task-related operators, the model has two operators that respond to the distraction. The first of these has as a condition that the visual resource is not used, and moves attention to the video. The second operator activates at the moment that the video is attended, and disengages immediately. This reflects the empirical fact that in the experiment, subjects typically attended the video for only 200-400ms.

Most of the time, the distraction operators do not have much of a chance to intervene in the equation solving process. When reading operators are engaged they have no chance at all, because the visual resource is in use, whereas a condition of distraction is that the visual resource is free. Whenever the model transitions between an operator for a transform step and one for an arithmetic step (or the other

way around), however, distraction has a small probability of winning the competition due to activation noise.

The model can explain the data, because the distraction operator only competes when the model is not reading the equation (which takes proportionally more time as the equation is easier), and because in the harder conditions the model switches more often between transformation and arithmetic, providing more opportunities for distraction.

A model of the find-the-differences game

The model of find-the-differences is relatively simple. It consists of three goals: a *search* goal that attends an unattended feature in the top picture, a *compare* goal that, given an attended feature in the top picture, finds the corresponding location in the bottom picture, and then compares the two. If they are the same, search continues, and if they are different, the *click* goal then clicks on the location where the difference was found.

Figure 5 illustrates the process: it first attends an arbitrary unattended feature, in this case an oval feature in the top figure at location (10,10). It represents this as "Oval4", which in this simplified representation stands for an oval of a particular shape and size. The compare goal then takes over, and finds the matching location in the bottom picture (O2), and concludes that that location also contains an Oval4. It therefore clears the visual buffer, allowing the search goal to find a new feature. In the second attempt, the feature in location (20,15) turns out to be different, which allows the click goal to retrieve the operator that clicks the location. The model keeps track of object it has checked, so will no revisit those.

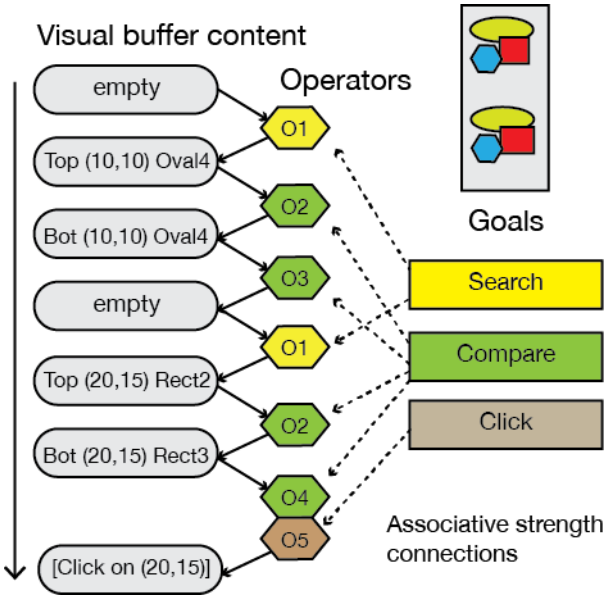


Figure 5. Illustration how the find-the-differences model operates on the picture in the top-left of the Figure.

Distraction is modeled in exactly the same way as in the memory game: whenever the visual resource is unused (i.e., when it is empty, so after each comparison), the distraction operator can direct visual attention to the movie. However, the model makes an additional assumption about the strength of this distraction, namely that it is proportional to the number of yet unattended visual features on the screen. If there are many unattended features on the screen, the video spreads less activation to the distraction operator than when that number is low. Nyamsuren and Taatgen (2013) have extensively modeled this spreading activation from perception to declarative memory: the spreading in this model is based on that work.

The model can explain the data because in the easier version of the task there are fewer visual objects on the screen, causing the movie to spread more activation to the distraction operator, and therefore increasing the probability that it is selected.

Results

Figure 6 shows the results of the simulation. Although the quantitative fit with the data is very good, it required fitting of the parameter that determines how much activation the distraction operator receives. The main quality of the model is therefore the ability to match the qualitative nature of the data.

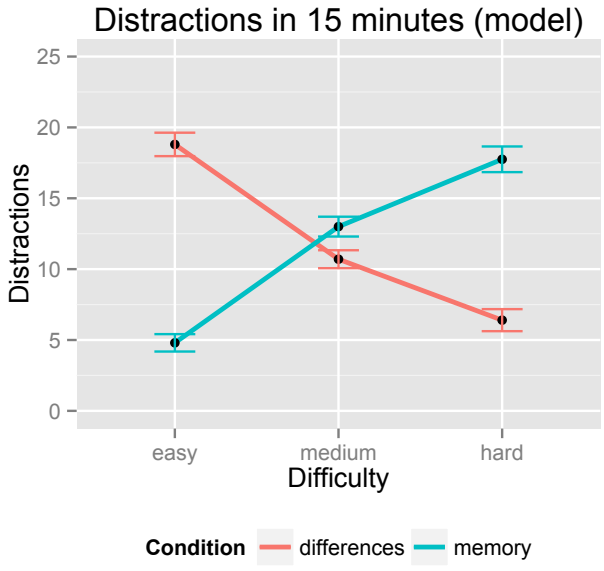


Figure 6. Model fit of the experimental data (from Fig. 4) as a function of task and difficulty level. Error bars indicate 1 standard error.

Although we cannot compare the performance on the memory task directly with the data, because we only partially modeled that task, we can compare performance on the find-the-differences task. Figure 7 shows the results.

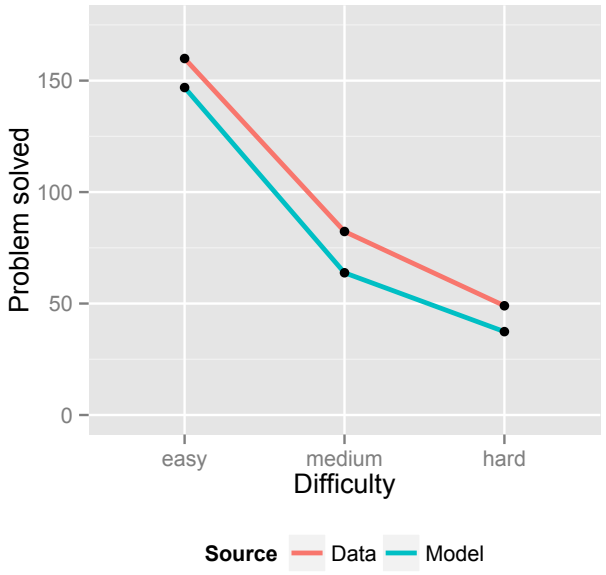


Figure 7. Comparison of number of problems solved between model and data in the find-the-differences game

Discussion and Conclusions

An important problem not addressed by previous models is how task- and non-task goals interact in producing behavior. This makes it hard, or impossible, to model why people suddenly change or give up on tasks. The example of distraction shows that the extension of PRIMs offers new options of modeling phenomena that would be hard to model in existing architectures. More monolithical approaches would probably have a hard time explaining why in some cases distraction increases with workload, and in other cases decrease with workload. In the case of a task that is heavy on reasoning, like the memory game, distraction can slip in at moments where one goal takes over processing from another goal, explaining why problems that require more of such switching are more susceptible to distraction. However, distraction can only intervene if it can latch on to resources that are currently unused. In the experiment, the distraction is visual, and can therefore only succeed if the visual resource is unused.

There is, of course, a danger in weakening the role of the task goal, because we don't want it to be derailed all the time by distraction or irrelevant other skills. This will require a more robust approach to modeling. However, other skills are not necessarily always distractions, but can be beneficial for the actual task, and thereby increase robustness.

In this experiment, distractions were relatively neutral: it didn't help nor hurt performance on the main task. However, in general other thought processes may be added as goals with slightly lower priority than the main task goals. For example, planning ahead is useful to do at moments that resources are available for such planning. A more "neutral" form of parallel processing may involve mind-wandering, which may be harmless, or may eventually turn into a self-induced distraction, and explain why people sometimes suddenly decide to check their email while doing something that is mentally taxing. PRIMs 2.0 opens possibilities for investigating mental processes that we all believe are taking place during experiments, but that we never model.

Another feature of PRIMs 2.0 is the option to build a model of a task by selecting certain existing goals that are connected to appropriate operators. That means that learning a new task involves the selection of goals, filling in certain specific values for those goals, and specifying a control strategy. A control strategy can be very simple: in our example here all goals were active in parallel. But in other cases it may be necessary to actively reinforce goals, in the same manner as Altmann & Gray (2008) did in their model of task switching.

Acknowledgements

This research was supported by ERC StG 283597 MULTITASK from the European Research Council.

References

Altmann, E. M., & Gray, W. D. (2008). An Integrated Model of Cognitive Control in Task Switching. *Psychological Review*, 602-639.

Altmann, E. M., & Trafton, J. G. (2002). Memory for goals: An activation-based model. Cognitive Science, 26, 39–83.

Anderson, J. R., & Douglass, S. (2001). Tower of Hanoi: Evidence for the Cost of Goal Retrieval. *Cognition*, *27*(6), 1331–1346. doi:10.1037//0278-7393.27.6.1331

Katidioti, I., Borst, J. P., & Taatgen, N. A. (2014). What happens when we switch tasks: pupil dilation in multitasking. *Journal of Experimental Psychology: Applied, 20*(4), 380–396.

Laird, J. E., Rosenbloom, P. S., & Newell, A. (1986). Chunking in Soar: The anatomy of a general learning mechanism. *Machine Learning*, (1), 11–46.

Nyamsuren, E., & Taatgen, N. A. (2013). Pre-attentive and attentive vision module. *Cognitive Systems Research*, 62-71.

Salvucci, D. D., & Taatgen, N. A. (2008). Threaded cognition: An integrated theory of concurrent multitasking. *Psychological Review*, *115*(1), 101–130.

Taatgen, N. A. (2013). The nature and transfer of cognitive skills. *Psychological Review*, *120*(3), 439–471.

Taatgen, N. A., Juvina, I., Schipper, M., Borst, J. P., & Martens, S. (2009). Too much control can hurt: A threaded cognition model of the attentional blink. *Cognitive Psychology*, *59*(1), 1–29.

Author Index

Author Index

Author Index