MALTA: Enhancing ACT-R with a Holographic Persistent Knowledge Store

Matthew F. Rutledge-Taylor (mrtaylo2@connect.carleton.ca)
Institute of Cognitive Science, Carleton University, 1125 Colonel By Drive
Ottawa, Ontario, K1S 5B6 Canada

Robert L. West (robert_west@carleton.ca)
Institute of Cognitive Science, Department of Psychology, Carleton University, 1125 Colonel By Drive
Ottawa, Ontario, K1S 5B6 Canada

Abstract

There is a concern that ACT-R (Anderson & Lebiere, 1998), and other cognitive architectures do not adequately account for long-term memory (Schultheis, Barkowsky, & Bertel, 2006). These architectures, while ideal for modeling single tasks, do not address the issue of preserving a task neutral knowledge store, which persists between tasks (Rutledge-Taylor, 2005). A dynamical holographic associative memory system, DSHM, based on Jones and Mewhort's model of the lexicon, BEAGLE (2007), is presented. A suggestion for a new cognitive architecture, MALTA, which combines elements of ACT-R and DSHM is described. Arguments for why this new architecture addresses some concerns with ACT-R's declarative memory system are offered.

Keywords: cognitive modeling; cognitive science; ACT-R; knowledge representation; holographic associative memory.

Introduction

Modeling memory is not new; detailed descriptive models of memory have existed since the 1960s. Atkinson and Shiffrin (1968), Craik and Lockhart (1972), Tulving (1972), and, Baddeley and Hitch (1974), among others, are credited with providing the first early influential models of memory. A more recent phenomenon is the formalization of memory models using computers. Since memory does not exist in isolation from other components of the human mind, it is natural to embed memory within a larger scheme for modeling all of cognition. The concept of the cognitive architecture arose from this rationale.

A cognitive architecture is a means for formalizing the principles that underlie one's theory about how the mind works (Newell, 1990). It specifies the structure of the mind, including how information is gathered, processed, and used to perform tasks. There are several advantages to implementing a cognitive model. Using the architecture, models can be built which can then be used to simulate humans performing tasks. The resulting simulation data can be compared to human experimental data, which allows for both the evaluation of the particular model, and the formalizations specified by the architecture itself.

Three of the most popular cognitive architectures are ACT-R (Anderson & Lebiere, 1998), SOAR (Laird, Newell & Rosenbloom, 1987; Newell, 1990), and EPIC (Kieras & Meyer, 1997). Of these, ACT-R has the best developed and well tested model of the human declarative memory system and is also highly consistent with mainstream theories about declarative memory from cognitive psychology.

The declarative memory system of ACT-R (DM) arose out of a need to account for the results of experiments on human memory generated by experimental psychologists. DM is also a necessary component of accounts of most experiments on cognition, due to the need to account for participants' memory of experimental instructions, and relevant general background knowledge. In developing an ACT-R model, a knowledge base relevant to the experimental task is usually provided to the model in advance of a simulation to represent knowledge held by the subject before beginning the experiment. During the experiment the model typically learns by creating new chunks and adjusting the relative activation of existing chunks (Anderson & Lebiere, 1998). This general scheme, and variations on it, has been very successful for modeling the sorts of tasks given to participants in psychological experiments related to memory and learning (Anderson & Lebiere, 1998).

However, a unified theory of cognition ultimately needs to go beyond modeling individual tasks done in isolation (Newell, 1990). It must also account for how it is that we, as intelligent agents, accumulate a vast amount of world knowledge that we apply in a context specific manner to the various tasks that we perform throughout our lives. In order to make this distinction clear we will refer to the knowledge that we carry around outside of a particular context as *persistent knowledge*, and the specific knowledge needed for a particular task as *task knowledge*. Task knowledge is a local manifestation of persistent knowledge that has been retrieved to make it available for the task. ACT-R is a very good system for modeling how people use and manipulate task knowledge, but there is very little work focusing on the management of persistent knowledge. One approach that has been used for modeling how people process the enormous database that makes up persistent knowledge is the use of holographic memory models.

The principle purpose of this paper is to present our Dynamically Structured Holographic Memory System (DSHM) as a means for modeling persistent knowledge and to suggest how it could be used to augment ACT-R. The format for the remainder of this paper is as follows: First, DSHM, based on the lexical representation system BEAGLE (Jones & Mewhort, 2007) is presented; second, a hybrid system, MALTA, which combines DSHM and ACT-

R is described; and third, how MALTA can, in principle, address the limitations of the ACT-R DM system is discussed.

Holographic Associative Memory

BEAGLE

DSHM is based on Jones and Mewhort's BEAGLE (Bound Encoding of the Aggregate Language Environment) model (Jones & Mewhort, 2007). The BEAGLE model learns a lexicon by creating holographic representations of words. The resulting memory system is capable of accounting for the results of various experiments on the lexicon, e.g., typicality judgments (Rosch, 1975).

The BEAGLE system takes as input, sentences, one at a time. Every word is represented internally, by the system as an item composed of two large vectors of numbers. One, called the environmental vector, uniquely codes the word and never changes. The other, called the memory vector, encodes information about which other words co-occur with the given word in the sentences imported into the system. This memory vector is modified in two ways. For every word in a sentence imported into the system, the sum of the environmental vectors of every other word in the sentence is added to the word's memory vector. This imbues the each word with traces of every word it has co-occurred with, but in a manner that is insensitive to the relative order of the words in the given word's environment. In order to preserve this order information a subsequent procedure is performed. The details of this rather complicated algorithm will not be described in full here. However, a key feature is that Jones and Mewhort make use of Tony Plate's holographic technique for binding vector representations of words together via circular convolution (Plate, 1995). Binding two vectors in this manner results in a vector with the same length as each of the original vectors. This allows for the recursive binding of lists of words.

Based on these two methods of encoding information about a word's neighbourhood, BEAGLE is able to develop a rich lexicon if provided a large enough corpus. After training, the memory vectors of words in the system can be analysed. The Euclidean distance between any two words is correlated with the difference in meaning of the two words. With sufficient training synonymous words will eventually converge to having identical memory vectors because the words appear in all of the same language contexts. Any difference in the memory vectors of near synonyms is indicative of idiosyncrasies in the contexts in which each word does and does not appear. For example, 'big' and 'large' appear in many of the same contexts and are often interchangeable. However, 'large' cannot be substituted for 'big' in "big sister". Thus, the two words are only near synonyms, and this would be reflected in BEAGLE by slight differences in their memory vectors.

In their published results, Jones and Mewhort (2007) used pairs of 2048 element vectors to represent words, and provided BEAGLE with a corpus of text approximately equal to what an average undergraduate student will have read in his or her lifetime. Words learned by the system distributed themselves over the state-space defined by their memory vectors in such a way that categorically similar words grouped together in a hierarchical manner. Thus, BEAGLE effectively solves a constraint satisfaction problem, where all of the words in a lexicon must be distributed in a finite multi-dimensional state space where the semantic relationships between the words are preserved in the distances between words in the space.

BEAGLE also functions as a pattern recognition system. The grammatical features, and more generally, the lexical features of English that it learns are discovered only by exploiting statistical regularities in the text imported into the system. As such, if presented with a sentence (or, any other string of words) with missing words, BEAGLE can be used to complete it. For example, if "he ____ to the highway" is presented, where '____' indicates a missing word, the system will nominate words like 'went' and 'drove' as those most suitable for filling in the blank. However, if "he ____ to the audience" is presented, 'spoke' would be the top candidate generated by BEAGLE. A key aspect of this ability of BEAGLE is that it is very sensitive to context. The two sentences above differ only by a single word. However, it is the presence of 'highway' in the first and 'audience' in the second that allows BEAGLE to hone in on a unique, small set of substitution candidates in each example.

The lexicon that BEAGLE develops is referred to as holographic due to the fact that it exhibits some high-level features of holograms. Holograms compress a three-dimensional scene onto a two-dimensional film; and, every part of the holographic film stores information about the entire scene. Similarly, BEAGLE takes some subset of natural language, English in this case, which has no defined dimensionality and is perhaps best understood as infinitely dimensioned, and represents it in 2048 dimensions (i.e., the length of the vectors used to represent words). Additionally, every word in the BEAGLE lexicon stores information about the entire language. Specifically, the lexical information stored in the memory vector of each word is the sum of all the contexts, properly encoded, in which it has appeared. Interestingly, the memory vectors of ubiquitous grammatical words such as the article 'the' store very little information due to the fact that the diversity of lexical contexts in which they appear blurs the contributions of each individual context. Words that appear in very specific contexts, such as 'Pharaoh' tend to be very informative about the contexts in which they appear. This is intuitively correct. The reader may recognize that the word 'the' does not bring to mind very many specific related concepts, whereas 'Pharaoh' immediately brings to mind ideas such as the great pyramids, and ancient Egypt.

It should be noted that the neuroscientist, Karl Pribram, was the first researcher of human cognition to recognize that the brain seems to have some of these features in common with holograms (Pribram, 1971; 1991).

DSHM (Dynamically Structured Holographic Memory System)

Jones and Mewhort's BEAGLE is a very important first step in understanding human memory and human cognition in general. BEAGLE is a powerful model of the lexicon. It demonstrates that knowledge of the referents of words is not necessary for many features of the lexicon to develop during the learning of a language. These features are extracted from only the statistical features of the (written) language.

Our claim is that the mechanisms employed by BEAGLE can be generalized such that they apply to information processing generally, and not to the processing of language in particular. BEAGLE takes as input sentences, which can be interpreted as ordered sets of words. DSHM operates on objects, which like the items in BEAGLE consist of an environmental vector and a memory vector. However, objects also consist of a set of components, which are either an ordered set of objects, an unordered set of objects, or the empty set (in the case of the atomic objects of the system). An ordered relationship between objects *x* and *y* can be variously interpreted as: *x* is to the left of *y*; *x* is superordinate to *y*; or, *y* is a token of the type *x*, etc. Thus, objects have recursive compositional structure over other objects.

ACT-R chunks are hierarchically structured. Each ACT-R chunk type consists of one or more slots that act like variable names. Each instance of a chunk consists in the pairing of each slot with a value, which can be either an atomic object of the system or another chunk. An ACT-R chunk can thus be interpreted as an unordered set of ordered pairs of objects, where each pair consists of a slot and a value. The set of pairs is unordered, because the order in which the slots of an ACT-R chunk are defined is not relevant to the representational content of the chunk. The pairs themselves are ordered, because for each slot/value pair, which object is the slot and which is the value is relevant. It should be noted that in DSHM there is no privileged distinction between a slot and a value as kinds. That which can appear as a slot can also appear as a value, and vice-versa. Just as BEAGLE is able to discover the syntactic categories of English, DSHM should similarly discover any regular type/token distinctions etc. in the data to which it is exposed.

DSHM is very flexible with respect to how to encode information. For example, mathematical knowledge in DSHM can be encoded similarly to the classic ACT-R example: [isa:addition-fact addend1:three addend2:four sum:seven]. Here, a colon indicates that the items to the left and right bear an ordered relationship to one another, and a space indicates that the items to the left and right bear an unordered relationship to one another. By default, spaces delimit ordered sets. However, DSHM affords the possibility of defining this chunk in a variety of ways, for example: [isa:addition-fact [3 4]:7]. In this case, rather than using slots and values for all of the content of the object, only the type of chunk is defined in this way. The remainder, [3 4]:7, can be interpreted as, "the addends 3 and 4, bear a relation, sum, to the number 7". Internally, the object [isa:addition-fact [3 4]:7] is composed as follows: the objects 'isa' and 'addition-fact' are bound together to form the complex ordered object [isa:addition-fact]; the objects '3' and '4' are bound together to form the complex unordered object [3 4]; the objects '7' and [3 4] are bound together for form [3 4]:7; and finally, the objects [isa:addition-fact] and [3 4]:7 are bound together. A feature of this representation is that, because [3 4] is unordered, it is identical to [4 3], and so, the system automatically knows that '4+3=7', after learning '3+4=7'. Thus, a redundant chunk representing the same fact, but with the addends reversed is not needed. However, in the case of subtraction, this would not be the case. The minuend and the subtrahend would have to bear an ordered relationship with one another; e.g., [isa:subtraction-fact [7:4]:3]. Other ways of representing this mathematical knowledge can be devised. As with ACT-R, comparisons to human data on math competence (e.g., from Mauro, LeFevre & Morris, 2003), would help determine what representational scheme best models human knowledge of mathematical facts.

Retrieval From Memory The retrieval of information from DSHM can be accomplished in two ways. First, the environmental vector of an object can be presented to the DSHM, and the objects with memory vectors that resonate with (are similar to) that vector will be retrieved. This is effectively asking the system for objects that have co-occurred with the given object. Second, an incomplete complex object can be presented to the DSHM. The system uses a rather complex algorithm to decode which objects best fill the missing parts of the object. For example, if presented with [isa:addition-fact [4 ?]:7], where '?' indicates a missing object, the system will first nominate completions of [4 ?]. A list of pairs consisting of '4' and one other object will be generated. These pairs will be based, roughly, on asking '4' who occurs most with it. Second, this list of pairs is presented to '7', and '7' is asked which of those object pairs often precedes it. This will drastically, reduce the number of objects remaining in the list by discarding pairs like [4 o'clock], and [connect 4] etc. Next, the candidates for the completion of [[4 ?]:7] are presented to the complex object [isa:addition-fact], where [[4 3]:7] will resonate best.

This object structure provides DSHM with flexibility not available to BEAGLE. DSHM is able to operate on sentences, just as BEAGLE does. To model sentence processing using DSHM, each sentence would be represented as an object with an ordered set of parts consisting of atomic objects representing each word. However, it is not necessary that objects representing words be atomic objects. Each word could have as parts an ordered list of atomic (objects representing) phonemes. The ability to create objects with hierarchical structure allows the study of top-down versus bottom-up processing. For example, DSHM can be used to investigate phoneme restoration effects by manipulating whether words with

deleted phonemes are presented alone or in the context of a sentence (Samuel, 1981). For example, one measure could be whether the phoneme missing in the word "b_g", where '_' indicates where a phoneme has been deleted, will be restored differently in the contexts, "b_g" alone, "the boy gave her a b_g for her groceries", and "the boy gave her a b_g for her gecko". If trained on a corpus where the words "big", "bag", and "bug" occur, with "big" appearing more frequently than the other two words, DSHM ought to predict that the "big" would be the top candidate in the first case. However, in the cases where the incomplete word occurs in the context of a sentence, the system subsequently evaluates how well the candidate words "big", "bag", and "bug", match the context of the rest of the sentence. If "bag" is much more likely to occur in the context of "groceries" than is "big", then "bag" could get a sufficient boost to move it ahead of "big" in the subsequent re-ranking of the candidate words, in the second case listed above. However, if "big" occurs much more frequently than "bag", the sentential context may not be enough to move "bag" ahead of "big". Frequent words are identified with more efficiency than infrequent words (Monsell, 1991), and so a preference for high frequency words should be a feature of DSHM.

MALTA

As stated above, ACT-R is a system for modeling human performance of specific tasks. The chunk represents roughly what can be entertained in working memory at one time. The declarative memory system (DM) of ACT-R, in our opinion, is a good model of a task specific short term store. That is, the collection of memories and newly discovered information relevant to the particular task at hand. What is needed to make ACT-R capable of modeling human performance across multiple tasks is a longer term store of persistent knowledge. It is our belief that DSHM can be integrated with ACT-R, so as to provide this persistent knowledge store. There are many possible ways to do this. Two, MALTA1 and MALTA2, will be described here.

The Multiple and Long-term Task Architecture (MALTA), is the name used to reference both the theory underlying MALTA1 and MALTA2, and the implementational features they share in common. MALTA is designed to model the manner in which human beings: transition from completed tasks to new tasks; switch back and forth between concurrent tasks; and, apply knowledge in a context specific manner to each task. MALTA1 is the most basic implementation of MALTA. It makes minimal changes to how ACT-R operates, and is best suited for accounting for how humans transition from completed tasks to new tasks. MALTA2 is more dynamic and involves a more drastic departure from orthodox ACT-R. It is suitable for accounting for how humans switch back and forth between concurrent tasks. In the descriptions of MALTA below, some details are omitted for clarity. All information should be assumed to be represented as DSHM objects, unless otherwise stated.

MALTA consists primarily of a DSHM, a dynamical holographic store of all persistent knowledge available to the system; a set of ACT-R model templates, each with an associated set of *template keys*, represented as DSHM objects; a sensory system (which can be implemented as ACT-R perceptual buffers); and, an executive responsible for integrating the DSHM and the ACT-R portion of the system. Each model template is identical to a standard ACT-R model except that its DM contains only a minimal set of predefined task specific chunks. However, the template may also define additional chunk types that can be used by the executive to properly format objects retrieved by the DSHM, for use by the model, as explained below.

MALTA1 The most basic way to integrate DSHM and ACT-R is to have them operate as independently as possible. MALTA1 is based on this premise. When MALTA1 is first started, and after it completes a task, the executive samples the sensory system and binds this to an optional trace left behind by the previously completed task (if any), to create a representation of the state of the system. This *state cue* is used to determine which of the ACT-R templates, contained within the given MALTA model, best matches the current context of the system's environment. This is accomplished by determining which set of template keys best match the state cue. The winning ACT-R template is used to invoke an ACT-R model suitable for performing a task relevant to the current environment. The default task specific chunks provided by the template are supplemented by the DSHM. This is accomplished, as follows. Each template specifies some number of chunk definitions. These definitions are bound to the state cue to create a set of context specific incomplete chunks, which are then presented to the DSHM. The DSHM completes these chunks, which are then imported in to the DM of the current ACT-R model.

The ACT-R model created using the template then runs without further interference from the rest of the system. However, as the model runs, all new and modified chunks are tagged with an updated state cue, and fed back into the DSHM. This way, all information acquired while performing a task is made available in the future to the system. When the task is completed, the executive resamples the environment and prompts the application of a new template as described above. Thus, MALTA1 operates as continuous series of ACT-R models, where each makes use of information in the persistent knowledge store (the DSHM) relevant to each task, while committing new information to memory.

MALTA2 MALTA2 operates in the same manner as MALTA1, but with a few differences. At any given moment in time, MALTA2 is both running an ACT-R model based on one of its templates, and predicting the future state of the system's environment. If that future state is inconsistent with the current task, it abandons the currently running model, and invokes a new one more

relevant to what the system anticipates. The executive accomplishes this by first encoding the current state of the environment every time a production fires in the current ACT-R model. This state cue is then bound to representations of the previous 2 states of the system to form a representation of the recent history of the system. The result of this calculation, the *history cue*, [current-state:previous-state:two-states-ago], is first committed to the DSHM. Next, for each chunk in DM, the chunk is converted to a DSHM object, bound to both the current state cue and the current template keys, and then committed to the DSHM. Once this is done, the current state cue is used to predict the next state of the system. This accomplished by creating an incomplete history cue, using the current state and the previous state as cues: [?:current-state:previous-state]. This history cue is completed by the DSHM. The *predicted state cue* used to complete the incomplete history cue is a state cue previously committed to the DSHM, and which in the past typically followed states similar to the current state cue and the previous state cue. See West and Lebiere (2001), for a discussion of this predictive technique, and how it is used to discover sequential dependencies in the evolution of dynamical systems.

The predicted state cue is then compared to the current template keys. If they fail to agree, the current task is judged to be inappropriate for what the system anticipates to be the relevant context of the environment. The current ACT-R model is abandoned, and the predicted state cue is used to invoke a new ACT-R model in the manner that MALTA1 does after the completion of a task. This scheme of tightly binding perceptions of the environment to cognition is inspired, in part by Lawrence Barsalou's situated simulation theory (2003).

A feature of MALTA is its ability to halt a current task, begin a new one, and then return to its previous task. A task can be resumed by the executive, if the state of the system causes it to invoke a model based on the same template as had previously been in operation. Since the state of the system determines what chunks will be retrieved from DSHM, a previous task can be considered to be continued if it just so happens that the DM system is restored to the state it was in prior to being interrupted. This would be the case if the state of the system returns to (at least nearly) the exact state it was in before the interruption (i.e., the same model template and perception of environment). For example, imagine that the system has invoked a mathematical problem solving model. The system progresses by detecting unsolved problems via the perceptual system, and going about taking the steps to solve them. Suddenly, a spider is detected in the environment of the model. This is inconsistent with doing math, and the executive abandons the mathematical problem solving model, in favour of the 'kill bug' model. Once the spider has been disposed of, the perceptual system returns to detecting unsolved math problems, and the mathematical problem solving model in reinvoked. Chunks pertaining to partially completed problems will be restored if the system detects a problem that it had been previously working on.

Resolving the issues

The authors believe that the features of MALTA described above will resolve some of the limitations of ACT-R, with respect to retaining task neutral persistent knowledge. The reason for this is that DSHM operates differently than DM. ACT-R creates, modifies, and stores explicit representations of chunks in DM, in the context of a specific task. However, it is not clear that the manner in which a chunk is structured during one task will make it suitable for retrieval during a different task. A set of special productions could potentially be designed to convert one set of chunks to another. However, this seems to be a redundant process at a minimum, and most likely a very difficult one to implement efficiently in a methodologically sound manner.

DSHM does not store an explicit representation of an object when it is imported into memory, as is the case with chunks and DM. Rather, DSHM stores, implicitly in the memory vectors of each component object of a complex object, the relations that exist between those components. As a result, every object in the DSHM stores information about every context in which it has existed, due to the binding mechanisms described above. Thus, each object can be associated with potentially thousands of other objects without any of these associations being explicitly encoded anywhere in the system. These associations can reflect very complex relations pertaining to the individual contribution of each object to the structures of other objects. That is, every object 'knows' about the kinds of structures to which it contributes, and what other objects were part of those structures. The authors believe these features to be and advantage over ACT-R models of long-term memory that rely on explicitly encoded connections between entities in memory (e.g., Schultheis, Barkowsky, & Bertel, 2006).

A central feature of MALTA is that it is not limited to retrieving only exact reproductions of previously stored chunks from DSHM. Just as synonyms group together in BEAGLE, so do both structurally similar complex objects and conceptually similar atomic objects group in DSHM. Each stored chunk can be interpreted as making more salient an attractor in the dynamical structural/conceptual state-space formed by DSHM. A retrieval cue consists of two parts: the structure of an incomplete object, and the information contained in the complete components of that object. The influence of the complete components is roughly to pick out a point in the state-space of the DSHM. The objects located around this point are evaluated for their suitability as candidates for filling in the missing components of the incomplete object. Thus, novel chunks can be generated when the converging contributions of the supplied retrieval cues pick out objects from DSHM that have not previously been brought together according to the constraints placed on that retrieval.

This feature of DSHM allows for chunks, specific to a particular context, to be generated without that exact context

having been presented to the given MALTA model previously.

Comparisons to other models

MALTA is a prototype for a cognitive architecture that the authors hope will provide a solution to the problem of representing and applying knowledge appropriately in a context specific manner. Although MATLA2 is itself a model of multi-tasking, it is born out of a different paradigm than are models of multi-tasking that make a less dramatic departure from conventional ACT-R, such as Salvucci's multi-tasking general executive (Salvucci, 2005).

Salvucci's general executive is a mechanism for managing the performance of concurrent tasks within the ACT-R framework. This is accomplished by implementing a new goal module which ranks a set of current goals associated with multiple tasks using a *queue*. Special productions add and remove goals as they are created and completed, respectively. The relative priority of tasks can be adjusted by flagging new goals with a temporal delay. Additionally, heuristics for interleaving incomplete tasks, and blocking low-priority tasks exist.

In contrast to traditional goal driven systems, MALTA does not make use of explicitly represented goals, at all. Rather, the authors are committed to the dynamical systems paradigm, whereby future states of the system depend primarily on the current state of the system (including the system's knowledge). Thus, there is no need to explicitly represent delays, priorities, and heuristics for task switching. All of these features naturally fall out of the system learning which states of the system precede, and follow, other states associated with current tasks and which are associated with invoking new tasks.

References

Anderson, J. R., & Lebiere, C. (Eds.) (1998). *The atomic components of thought*. Mahwah, NJ: Lawrence Erlbaum.

Atkinson, R. C., & Shiffrin, R. M. (1968). Human memory: A proposed system and its control processes. In K. W. Spence and J. T. Spence (Eds.), *The Psychology of Learning and Motivation: Vol 2. Advances in Research and Theory*, (pp. 89-195). New York: Academic Press.

Baddeley, A. D. & Hitch, G. (1974). Working memory. In G.H. Bower (ed.) *Recent Advances in Learning and Motivation: Vol. 8.* (pp. 47-90). New York: Academic Press.

Barsalou, L. W. (2003). Situated simulation in the human conceptual system. *Language and Cognitive Processes, 18*, 513–562.

Craik, F. I. M. & Lockhart, R. S. (1972). Levels of processing: A framework for memory research. *Journal of Verbal Learning and Verbal Behavior, 11*, 671-684.

Jones, M. N., & Mewhort, D. J. K. (2007). Representing word meaning and Order information in a composite holographic lexicon. *Psychological Review, 114*, 1-37.

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*, 391-438.

Laird, J., Newell, A., & Rosenbloom, P. (1987). Soar: An architecture for general intelligence. *Artificial Intelligence, 33*, 1-64.

Mauro, D. G., LeFevre, J., & Morris, J. (2003). Effects of problem format on division and multiplication performance: Division facts are mediated via multiplication-based representations. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 29*, 163–170.

Monsell, S. (1991). The nature and locus of word frequency effects in reading. In D. Besner & G. W. Humphreys (Eds.), *Basic processes in reading: Visual word recognition*. (pp. 148–197). Hillsdale, NJ: Lawrence Erlbaum.

Newell, A. (1990). *Unified theories of cognition*. Cambridge, MA: Harvard University Press.

Plate, T. A. (1995). Holographic reduced representations. *IEEE Transactions on Neural Networks*, 6, 623-641.

Pribram, K. H. (1971). *Languages of the brain: Experimental paradoxes and principles in neuropychology*. Englewood Cliffs, NJ: Prentice-Hall.

Pribram, K. H. (1991). *Brain and perception: Holonomy and structure in figural processing*. Hillsdale, NJ: Lawrence Erlbaum.

Rosch, E. H. (1975). Cognitive representations of semantic categories. *Journal of Experimental Psychology: General, 104*, 192-233.

Rutledge-Taylor, M. F. (2005). Can ACT-R realize "Newell's Dream"?. *Proceedings of the XXVII Annual Conference of the Cognitive Science Society*, 1895-1900. Stresa, Italy: University and Polytechnic of Turin.

Salvucci, D. D. (2005). A Multitasking General Executive for Compound Continuous Tasks. *Cognitive Science, 29*, 457-492.

Samuel, A. G. (1981). Phonemic restoration: Insights from a new methodology. *Journal of Experimental Psychology: General, 110*, 474-494.

Schultheis, H., Barkowsky, T., & Bertel, S. (2006). LTM^C — An improved long-term memory for cognitive architectures. *Proceedings of the Seventh International Conference on Cognitive Modeling*, 274-279. Trieste, Italy

Tulving, E. (1972). Episodic and semantic memory. In E. Tulving and W. Donaldson (eds.) *Organisation of memory*. London: Academic Press.

West, R. L., & Lebiere, C. (2001). Simple games as dynamic, coupled systems: Randomness and other emergent properties. *Cognitive Systems Research, 1(4)*, 221-239.