Ritter, F. E., & Young, R. M. (Eds.). (1998). Proceedings of the Second European Conference on Cognitive Modelling. Thrumpton (UK): Nottingham University Press. ISBN 1-897676-67-0

Invited Talks

Ritter, F. E., & Young, R. M. (Eds.). (1998). Proceedings of the Second European Conference on Cognitive Modelling. Thrumpton (UK): Nottingham University Press. ISBN 1-897676-67-0

Ritter, F. E., & Young, R. M. (Eds.). (1998). Proceedings of the Second European Conference on Cognitive Modelling. Thrumpton (UK): Nottingham University Press. ISBN 1-897676-67-0

Modeling Neural Function and High Level Cognition

Marcel Just
Center for Cognitive Brain Imaging
Department of Psychology
Carnegie Mellon University
Pittsburgh, PA 15213, USA
+1 (412) 268-2791
just+@cmu.edu

ABSTRACT

Recent brain imaging findings suggest several new assumptions concerning the architectural properties of the neural systems that underlie high level cognition, such as language, comprehension, visual cognition, and problem solving. Some of these assumptions have to do with

1. resource-constrained processing and task assignment;
2. dynamic configuration and resource recruitment;
3. functional embedding, self-similarity, and interaction among the components of the cognitive system;
4. a preference ordering for the types of processing that each cognitive component can perform (graded specialization).

The 4CAPS computational modeling system implements these assumptions, with the goal of accounting not only for processing times and error probabilities, but also for the amount of brain activation observed in each of the activated component neural systems. 4CAPS consists of several component processing modules, each of which is a parallel production system with some connectionist properties, and each of which is intended to correspond to the function of an underlying large-scale neural network. The component production systems are highly interactive with each other, operate in parallel, and have a task allocation regimen based on graded specialization and resource availability.

Ritter, F. E., & Young, R. M. (Eds.). (1998). Proceedings of the Second European Conference on Cognitive Modelling. Thrumpton (UK): Nottingham University Press. ISBN 1-897676-67-0

Mechanisms and Implications of Pervasive Episodic Memory

Erik M. Altmann
Psychology Department and Krasnow Institute
George Mason University
Fairfax, VA 22030 USA
+1 703-993-1326
altmann@gmu.edu

ABSTRACT

This paper investigates the memory phenomena underlying directed access to hidden objects. A computational cognitive model is described that encodes long-term episodic traces automatically whenever it attends to an object in its environment. Later, if an object of interest is hidden from view, the model can try to remember seeing it. This involves generating appropriate cues from memory to try to trigger episodic traces encoded while attending to that object. The underlying cognitive architecture (Soar) constrains the nature of these cues and the processes required to generate them. These constraints lead to a theory of episodic indexing, which is that people store simple information about attention events in large amounts, but make use of it only to the extent that they are able to generate appropriate images from memory. Episodic indexing helps characterize the cognitive cost of a cluttered interface.

Keywords

Cognitive simulation, episodic memory, human-computer interaction, Soar

INTRODUCTION

Our surroundings are filled with information. Most of this is hidden to us at any given time, being out of our field of view, yet we manage to gain access to it when we need to. For example, we might recall seeing a figure in a book, or a key phrase, and then return to that area in the book to refresh our memory, or to examine the context more carefully.

This paper investigates the memory phenomena underlying such access to external information. What do people encode about something they see, such that they can remember later that it exists? We would like to know both what information is stored, and under what circumstances. Second, what causes the retrieval of these memories? People typically navigate their environment for a purpose rather than haphazardly, implying some knowledge of a target to be visited. We would like to understand the role of domain knowledge in mediating access to what we know exists in our environment.

Our approach to these questions is to represent the phenomena explicitly using a cognitive architecture, Soar (Newell, 1990; Rosenbloom, Laird, & Newell, 1992). Soar includes mechanisms grounded in psychological theory and data that impose constraints on the representation of behavior. Applied to hidden-object access, these constraints imply that people store large amounts of information about their environment, but retrieve it only occasionally and with requisite knowledge and cognitive effort.

The paper is organized as follows. We first characterize the kind of task that requires the fine-grain episodic memory for efficient performance, and introduce the model using simple hypothetical examples to illustrate its encoding and retrieval processes. We then offer an accounting of the memory bandwidth implied by pervasive episodic encoding. Finally, we examine the theory for consistency with other findings on episodic memory, and for design of interfaces to extensive information environments.

THE MODEL

The kind of hidden-information access we are interested in studying is illustrated by the following scenario. A computer user is working with an application that generates much more information than fits on the screen at once. Most of this information is hidden, scrolled out of the way by the application to make room for the new information that it generates continually. This old information remains accessible, and the user occasionally scrolls some of it back into view. Thus, the user appears to have a memory that functions as an index to the environment. Much as the index in a book supports looking up a term of interest, the episodic index stored in memory supports "looking up" objects of interest in the environment. We are interested in how this index is created in memory, and how it is later accessed. In the following, we use examples from a hypothetical database programming task. The real task simulated by the model is described elsewhere (Altmann & John, in press; Altmann, 1996; Altmann, Larkin, & John, 1995).

The model's main mode of performance is a kind of comprehension in which it tries to gather information about objects in its environment. This is a generalized and simplified representation of interaction with an information-rich environment. In particular, it is simplified in that the model does not construct the complex mental structures generally associated with comprehension of text (e.g., Lewis, 1993; Kintsch, 1998).

The model selects goals to comprehend objects and issues commands to change the display. Some commands generate new information, and some scroll to old information. The model uses this external information as it tries to comprehend objects.

To comprehend a particular object, the model selects subgoals that retrieve information about that object. Information can come either from the display (an external source) or from LTM (an internal source). For example, suppose the model is comprehending a data structure that represents a student record. The student record contains a

Ritter, F. E., & Young, R. M. (Eds.). (1998). Proceedings of the Second European Conference on Cognitive Modelling. Thrumpton (UK): Nottingham University Press. ISBN 1-897676-67-0

field for the student's Social Security Number (SSN), which is displayed on the screen. To retrieve information about this field, the model selects an attend subgoal. Suppose (for simplicity) that the model attends only to the field and not to the actual number stored there. This act of attention would add the following attribute-value pair to WM.

```
(^field ssn)          From attending to SSN field.
```

Alternatively, if this information is not available externally but the model has the appropriate domain knowledge, the same information can be recalled from LTM. To do this, the model selects a probe subgoal. For example, the model might probe with the SSN field, perhaps to see if this activates any other information relevant to the student record. Probing and attention are symmetrical in that a probe can look exactly like the output of attention.

```
(^field ssn)          From probing with SSN field.
```

Under episodic indexing, attention and probing process another kind of element, one which represents the actual event of attending to an object. Attention automatically adds this element to WM as a side effect of attending to an object. Thus the full outcome of attending to an SSN field would be the following.

```
(^field ssn)          From attending to SSN field.
(^attended-to ssn)    From attending to SSN field.
```

The same representation could also be produced by a probe, consistent with the attention-probing symmetry noted above. The probe below represents the model asking itself, "What do I know about the event of attending to an SSN field?"

```
(^field ssn)          From probing with SSN field.
(^attended-to ssn)    From probing with SSN field.
```

We refer to an attribute-value pair like attended-to ssn, when generated by a probe, as an image of attending to an object. The term image is meant to suggest a code like that produced by attention, namely more like a percept than an abstraction or a concept. Beyond this, we do not attempt to interpret the model's images phenomenologically, or psychologically in terms other than how they function in the model. For example, their symbolic nature reflects Soar's representation language and is not intended as a statement in the debate over propositional vs. analog spatial codes. In general, LTM contains many kinds of codes (Bower, 1975), and in particular expert programmers often use vivid imagery to understand programs, including color, sound, and dancing symbols (Petre & Blackwell, 1997). Amidst this diversity it seems reasonable to posit a code representing the event of attending to an object.

Thus the model can imagine attending to an object, providing it has the knowledge to do so. Such imagining, and hence the requisite store of images, is the basis of the retrieval processes of episodic indexing.

Learning in Soar

Encoding information about the environment is a form of learning, and requires that the model modify its long-term knowledge representation. In Soar, all long-term knowledge is represented productions. These are condition-action rules like the one below. If the condition part (above the arrow) matches a structure in WM, then the action part (below the arrow) adds new elements to WM. The production below acts as a declarative memory, because it associates an object (a student record) with facts about that object (that it has an SSN field). In general, all the model's operations, like attending to objects, generating probes, and recalling facts, depend on knowledge represented as productions.

```
    (^structure student-record)     Condition:
                                    Student record in WM.
-->
    (^field ssn)                    Action: Put SSN field in
                                    WM.
```

The model learns by acquiring new productions. The learning mechanism is part of Soar. It is unified with Soar's knowledge-representation language (productions) and control structure (goals) in that Soar acquires new productions in response to achieving goals (Laird, Rosenbloom, & Newell, 1986). A new production, or chunk, represents an inference that may have taken several steps to make. The chunk is added to LTM, making the inference available in a single step from then on.

For example, suppose the model's goal were to find the sum of two numbers (4 and 7) and that although it could not retrieve the sum directly from memory, it knew a procedure for adding by counting up from one of the addends. The goal to find the sum would be implemented by subgoals that might involve initializing a running sum to the value of one addend, invoking the counting procedure, and recognizing when the count equaled the other addend. The result (11) would represent achieving the goal, and Soar would encode a chunk associating the relevant inputs to the counting procedure with the new result. In the future, this chunk will compute 4 + 7 = 11 without subgoals, bypassing the counting procedure.

In general terms, a chunk encodes an association between an inferred result (e.g., the sum) and the WM elements on which the inference is based, which we refer to here as premises (e.g., the addends). The premises have either already contributed to achieving the current goal or were in WM when the goal was selected. The result is inferred from the premises through a sequence of intermediate production firings. A chunk will fire immediately in the future if WM contains the same premises.

The chunking process does very little induction or generalization. The result essentially becomes the chunk's action and the premises become the chunk's conditions, though there is some variabilization (Laird, Rosenbloom, & Newell, 1986). This makes a chunk specific to its encoding context, consistent with the encoding specificity principle (Tulving, 1983). This specificity acts as a hard constraint on the nature of the process for retrieving learned knowledge (Howes & Young, 1997).

Encoding the Episodic Index

The model contains two key assumptions about the process of attending to an object. Both assumptions are related to the event of attending. The first assumption is that the event itself is worth representing in WM, apart from the object of attention. The second assumption is that all attention events are goal-directed. This assumption says that the model is always looking for new information about the object it is trying to comprehend,

Ritter, F. E., & Young, R. M. (Eds.). (1998). Proceedings of the Second European Conference on Cognitive Modelling. Thrumpton (UK): Nottingham University Press. ISBN 1-897676-67-0

and therefore automatically takes any attention event to contribute to the current goal. The two assumptions together operationalize what we might think of informally as "paying attention to" or "concentrating on" what we are doing. The important implication is that if the model "pays attention" to an event, this enables remembering the event because it causes chunks to be acquired.

The first assumption (that attention events are noteworthy) is implemented as follows. When the model attends to an object, it records the event using its internal clock. That is, it associates the WM code for the attention event with the current value of an internal variable that is updated periodically. For example, when the model attends to the SSN field of a student record, the complete representation created in WM is something like the following.

```
(^attended-to ssn)          From attending to SSN field.
(^event ssn ^time t42)      From attending to SSN field.
```

The model's internal clock ticks when it selects a new object to comprehend (meaning that the model's sense of time is keyed to its train of thought). All objects attended while comprehending that object are encoded in LTM with the current time symbol.

The second assumption (that episodic processing contributes to the current goal) is implemented by associating the time symbol with the current goal in WM. This causes Soar to build a chunk, as described in the previous section. The premise of the chunk is the attribute-value pair representing attention to the SSN field, and the result is the time symbol. The two are linked by the inference that the SSN field was attended now. The chunk is shown below (named attended-ssn for reference later).

```
chunk: attended-ssn     Chunk for an attention event.
 (^attended-to ssn)
 -->
 (^event ssn ^time t42)
```

Attended-ssn represents an attention event. This makes it an episodic trace, as distinct from a semantic trace with no temporal content (Tulving, 1983). It functions as one entry in an index of objects encountered in the environment. In the future, if no SSN field is visible, the model can look up the SSN field in this index by attempting to cause this chunk to fire. If the lookup is successful, then the model can infer that it attended to an SSN field in the past, even though no such field is currently visible. The lookup and inference processes are described in the next section.

Retrieval from the Episodic Index

The episodic index consists of a set of chunks, each of which associates an attention event with a time symbol. Suppose that a particular attention event occurred long enough in the past that it is no longer active in WM and that the corresponding object is no longer in view. The model can use its episodic index to see if the object exists somewhere in the environment. This requires two steps. The first is to generate the cue necessary to get an episodic chunk to fire. We can think of this as "looking up" the object. The second is to make the appropriate inferences based on any recalled time symbols. We can think of this as acting on the information retrieved from the lookup.

To look up an object, the model must add to WM an image of attending to that object, as a cue for triggering episodic chunks. As discussed previously, an image can appear in WM either through attention, which generates the image from an external stimulus, or through probing, which generates the image from memory. In either case, an image appearing in WM will activate all episodic chunks acquired whenever the corresponding object was attended in the past. Production imagine-ssn, below, generates the necessary probe for the SSN field.

```
production: imagine-ssn
   Conditions testing that it's relevant to know that
   an SSN field was seen.
 -->
 (^attended-to ssn)          A1
 (^imagined ssn)             A2
```

Imagine-ssn will fire in a situation in which it would be useful to remember seeing an SSN field. For instance, suppose (as we did previously) that the model were asked whether a given database record contained confidential information. The model might try to recall seeing an SSN field by firing imagine-ssn. When imagine-ssn fires, A1 adds to WM an image of attending to the SSN field, providing an opportunity for a chunk like attended-ssn to fire. A2 tags this image as generated from memory rather than from a stimulus. In general, there could be many situations in which it might be useful to imagine an SSN field. Each would be represented in a production like imagine-ssn (with different conditions).

If we suppose that attended-ssn fires in response to imagine-ssn, then WM will contain the following elements.

```
(^attended-to ssn)          From imagine-ssn.
(^imagined ssn)             From imagine-ssn.
(^event ssn ^time t42)      From attended-ssn.
```

From these elements the model can infer that an SSN field exists in the environment. The production that makes this inference is recall-seeing-object, below. This production belongs to the set of generic mechanisms that form part of the model's static knowledge.

```
production: recall-seeing-object
 (^attended-to <o>)                 C1
 (^imagined <o>)                    C2
 (^event <o> ^time <then>)          C3
 (^time <now> != <then>)            C4
 -->
 (^recall-seeing <o>)
```

Recall-seeing-object's conditions, numbered on the right, are as follows. Conditions C1 and C2 test that there is an image in WM that was generated internally rather than from an external stimulus.[1] C3 and C4 test that the image was attended in the past. The single action summarizes

[1] Angle brackets around a letter (e.g., "<o>") indicate a variable. If the same variable occurs in multiple conditions, it must have the same value in each condition for the production to fire. Thus, for example, C1 and C2 test that the object bound to <o> is both attended-to and imagined.

Ritter, F. E., & Young, R. M. (Eds.). (1998). Proceedings of the Second European Conference on Cognitive Modelling. Thrumpton (UK): Nottingham University Press. ISBN 1-897676-67-0

what is expressed by the conditions. It adds to WM the recollection of having seen the object.

The identity comparison in C4 is the only operation afforded by time symbols. Thus time is categorical, rather than ordinal or interval, and the only categories are present (the current comprehension goal) and past (any previous goal). The model cannot compute, for example, the interval between two events. This information-leanness is consistent with qualitative aspects of the rapid decay of unelaborated temporal codes in people (Underwood, 1977).

The nature and use of the episodic index is shaped by Soar's constraints on learning. Because Soar makes a chunk specific to its encoding context, attended-ssn's conditions are tied to the object code that appeared in WM during the attending event. This specificity implies that recalling the existence of an object must be preceded by imagery involving the object.

Summary of Assumptions

There are four theoretical assumptions that shape how the model acquires and retrieves memories for attention events. The first assumption is that the attention event itself is worth symbolizing in WM, in addition to the attended object. The second assumption is that attention is an integral part of comprehension and thus contributes to every comprehension goal. These two assumptions are hypotheses that we have embodied in the model.

The third and fourth assumptions come with Soar. The third is that all knowledge that contributes to achieving a goal is stored permanently in chunks. The fourth is that chunks are specific to their encoding context.

Together, these assumptions imply that chunk acquisition in the model will be pervasive and automatic, and that retrieval will be effortful. Learning will be pervasive because the model will encode a new episodic chunk for every object it attends to. This learning is automatic, in that the model exercises no control over whether or not to learn, and in that learning is a side effect of attentional processing rather than an end in itself. Retrieval will be effortful because learning involves little induction. To get chunks to fire, cues describing the original encoding context will have to be generated from memory.

Knowledge Distinguished by Dperation

Episodic indexing encodes information about dynamic information arising during task performance. It also allows us to make distinctions among the different operations facilitated by the domain knowledge that one brings to a task. Domain knowledge is involved in three operations:

- Attention. During the acquisition episode, the model must know what to attend to in the first place. Thus the model must be able to identify objects and understand them to be relevant to the task at hand.
- Retrieval. During the retrieval episode, the model must (a) be able to generate an image, and (b) do this when the results of a successful probe on that image would be useful. Thus retrieval depends on both visual familiarity and semantic understanding specific to the particular domain.
- Action. The decision to revisit a hidden object is distinct from recalling that it exists. There might be other means for acquiring the information that the object could provide, and there might be no reason to act on the recollection.

Thus the model points to several operations by which relatively static domain knowledge helps us gain access to the relatively dynamic information around us.

PRODUCTIONS ENCODED AND FIRED

The model simulates 10.5 continuous minutes of problem solving, spanning the encoding and retrieval episodes of a number of scrolling events (Altmann, 1996). This extended lifetime served as a form of methodological control during our analysis. A sufficiently close examination of the data to construct the model was the best way to avoid missing events between acquisition and retrieval that might have recoded or otherwise affected the nature of the participant's episodic index. This extended lifetime also serves to illustrate the implications of pervasive episodic encoding for the bandwidth of the model's memory system in terms of the number of chunks acquired and fired.

Figure 1 tabulates productions and firing counts according to four categories of knowledge (arrayed horizontally). The top bar indicates the number of productions in each category when the model stops, including all preloaded productions and all chunks acquired as the model runs. The bottom bar indicates the total number of production firings in each category during the model's run.

Ritter, F. E., & Young, R. M. (Eds.). (1998). Proceedings of the Second European Conference on Cognitive Modelling. Thrumpton (UK): Nottingham University Press. ISBN 1-897676-67-0

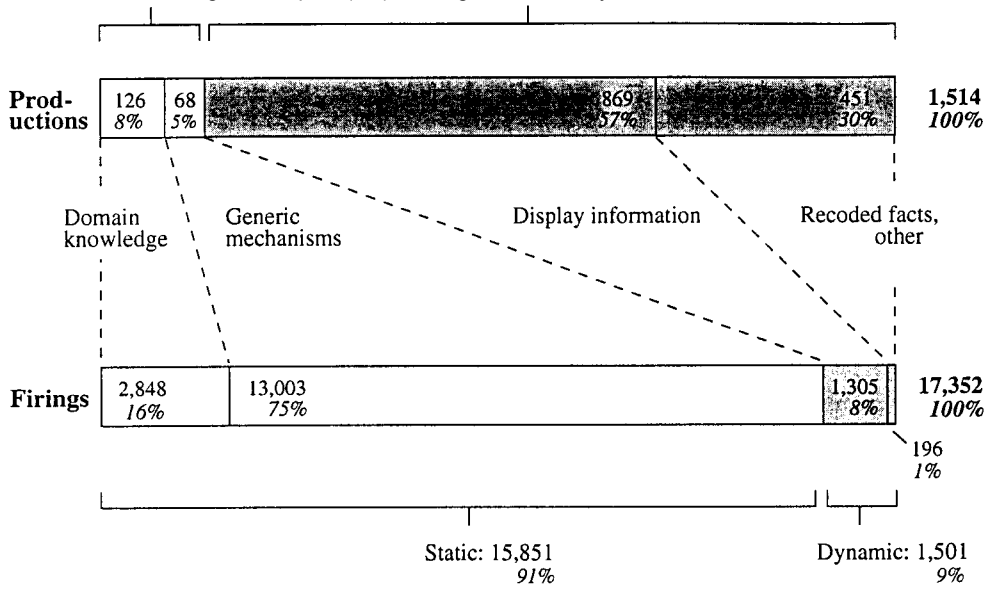

Figure 1: Production and firing counts.

The right half of the picture (shaded) shows that most of the model's productions are acquired by learning, but fire seldom because they are specific to their encoding context. The large number of chunks at the end of the run (1320) indicates the extent to which learning is pervasive. In terms of real time, the model is encoding roughly two chunks per second.

Few of these chunks fire, but some do. In particular, chunks encoding some aspect of the display account for 8% of firings. Thus the model's behavior depends in part on a memory for specific external situations that arise during task performance.

The left half of the picture (unshaded) shows that the model begins with a small number of preloaded productions that account for most of its processing. Preloaded productions number 194 (13%), of which 126 (8%) represent domain knowledge that we attribute to the programmer. This knowledge lets the model attend to external objects, generate cues, and recall facts about objects. It also tells the model what commands it can issue and what objects are important to comprehend and therefore select as comprehension goals.

Expertise should be flexible, in that it should guide behavior under a variety of appropriate circumstances. In our model, a large number of static domain-knowledge productions (93 out of the 126 indicated in Figure 1) represent either comprehension goals or attend or probe subgoals. These 93 productions account for all 499 goal and subgoal selections that occur as the model runs, for a mean of 5.4 goals per production. They also account for 2,518 out of the 2,848 firings of domain-knowledge productions. These measures indicate that to a large extent the model's goal and subgoal productions transfer among situations rather than being hardwired to a particular one.

The category of preloaded productions labeled generic mechanisms accounts for 75% of total production firings, despite being only 5% of the total number of productions. These are domain-independent productions like recall-seeing-object, discussed earlier, which infers the existence of a hidden object from an episodic trace. The high firing rate of mechanistic productions is consistent with their being the most general productions in the model and potentially general across many domains.

The production and firing counts over the model's lifetime illustrate the implications of pervasive episodic learning. In a few minutes of simulated time, the model acquires a great deal of dynamic information about its environment and stores it permanently in LTM. Some of these chunks transfer in the near-term, firing seconds to minutes (of simulated time) after being created. The fast rate of learning -- 1,320 productions over 10.5 minutes -- suggests that Brooks's (1977) estimate of tens or hundreds of thousands of rules making up a programmer's static domain knowledge may account for only part of what generates expert performance. There may in addition be a vast and constantly growing store of rules capturing dynamic knowledge.

DISCUSSION

Below we discuss the relationship of episodic indexing to previous conceptions of episodic memory in Soar and to a related theory advanced to account for expanded working memory for domain experts. We then speculate on episodic indexing and the cognitive cost of clutter.

Episodic Memory in Soar

Episodic memory is a natural construct to study in Soar. Learning is closely integrated with performance, meaning that events are easy to capture and store in LTM. Moreover, chunk conditions are determined by a process that gives chunks an inherently episodic quality. The chunking mechanism traces from a result back to the WM elements from which the result was generated, encoding an association between the result and important elements of context in which it was encoded. Thus the simple existence of a chunk represents some episodic information. A model can gain access to this information by generating cues that would cause the chunk to fire if it

Ritter, F. E., & Young, R. M. (Eds.). (1998). Proceedings of the Second European Conference on Cognitive Modelling. Thrumpton (UK): Nottingham University Press. ISBN 1-897676-67-0

existed, then by monitoring WM for the appearance of the chunk's result. Several Soar models have addressed episodic memory in these terms (e.g., Rieman, Young & Howes, 1996; Rosenbloom, Newell, & Laird 1991).

However, episodic indexing requires richer information to decide whether an object was actually attended at some time in the past. Below we examine the constraints met by the model's time symbols, and how these constraints arise from the interaction of pervasive episodic encoding (an assumption in our model) with encoding specificity (an architectural constraint inherited from Soar).

Episodic encoding extends to probe events as well as attention events -- that is, the model encodes episodic chunks for both. This follows from assuming that attention is integral to comprehension and thus contributes to every comprehension goal. Probing contributes equally to comprehension, and thus with respect to episodic learning the model treats probing and attention symmetrically.

This symmetry could lead to confusion should the model probe repeatedly with the same image. A particular probe will trigger episodic chunks from all previous probes, potentially leading the model to mistake these past probes as attention events. A kind of reality monitoring (Johnson & Raye, 1981) is necessary to avoid this mistake (and hence to avoid scrolling to imaginary objects). To support this reality monitoring, episodic chunks must contain enough information about the source of a memory (attention vs. probing) to let the model discriminate past attention events from past probe events.

Identifying past attention events must be done indirectly because source information cannot be represented explicitly in episodic chunks, when the source is the environment. This seems a surprising constraint, but it follows from encoding specificity. When building a chunk, Soar traces from the result back to premises existing before the result was generated, and encodes these premises as conditions. Therefore, if source information is a result, it also becomes a condition. Thus if a chunk has an action identifying an object as real, its conditions can never be met by an image alone. However, by the same logic, a chunk can have an action identifying an object as imagined and still be triggered by an image. Thus the model includes an probe tag with each chunk built during a probe event (see Appendix).

At retrieval time, these probe tags provide part of the information necessary to decide if the object of interest was ever attended. To make this decision, the model must identify all episodic chunks triggered by the current probe but built during past probes, and subtract them from the total set of episodic chunks triggered by the current probe. If the resulting set is non-empty, then the object was attended in the past. In terms of predicate calculus, the model tests an existential quantifier ("Did I recall an attention event?") by testing a negated universal quantifier ("Did I recall any event that was not a probe?"). This requires that each episodic chunk be uniquely identifiable. Because chunks are identifiable only by their results, this in turn requires that each episodic chunk have a uniquely identifiable result. This requirement cannot be met by a fixed set of symbols because at most one instance of any particular symbol can be represented in WM at any given time whereas the number of distinct events to represent is effectively unbounded. The requirement is met by the model's time symbols (as illustrated in the Appendix), because each is unique and they are generated anew at regular intervals.

Thus episodic indexing contrasts with previous Soar formulations of episodic memory in which multiple chunks may have the same result (e.g., Rieman, Young & Howes, 1996; Rosenbloom, Newell & Laird 1990). The episodic representation in our model is implied by theoretical assumptions interacting with task requirements in a way that does not constrain these other models. Our assumptions specify an indiscriminate encoding of episodic chunks, and the task requires that chunks from attention events transfer to probe events. However, this transfer requirement combined with encoding specificity restricts the source information that episodic chunks can represent. To compensate they are made discriminable by their results, allowing the model to partition past events into probe events and all the rest. This shaping of a representation by a complex interaction of constraints illustrates the benefit of taking a comprehensive and integrated approach to modeling cognitive phenomena (Newell, 1973).

A Form of Long-Term Working Memory

Episodic indexing posits that access to dynamic information depends on static information that one brings to the task. In this it is congruent with long-term working memory (LT-WM; Ericsson & Kintsch, 1995), of which a central claim is that long-term knowledge (as opposed to inherent WM capacity; e.g., Just, Carpenter & Hemphill, 1996) accounts for functionally expanded WM in domains in which one has expertise. Episodic indexing and LT-WM both propose that people store information rapidly in LTM, using domain knowledge to organize it and gain access to it later.

Episodic indexing extends LT-WM in the direction of leaner and more ubiquitous memory structures acquired at encoding time. The most routine application of LT-WM reviewed by Ericsson and Kintsch (1995) is text comprehension, but even this involves online encoding of memory structures that represent potentially intricate semantic mappings. For example, in referent resolution the comprehender must represent the connection between a pronoun and what it stands for, which is a semantic association that is not always straightforward to establish. By contrast, the episodic index is a one-way mapping from semantic to episodic codes which lacks the network structure that typically characterizes semantic memory.

The Cost of Clutter

Episodic indexing suggests that clutter has a cognitive cost, due to the paucity of information encoded with episodic traces and the effect this has at retrieval time. An episodic retrieval indicates the existence of an object of interest but not its whereabouts. This is consistent with the difficulty that even experienced users have in recalling features of interfaces (Mayes, Draper, McGregor, & Oatley, 1988; Payne, 1991), and with findings that spatial and location knowledge is not automatically encoded in real-world task environments (Lansdale, 1991). It is also consistent with the generally reconstructive nature of memory for the source of an item (Johnson, Hashtroudi, & Lindsay, 1993).

One possible strategy for dealing with clutter might be to add spatial information to the episodic information encoded during attention. However, encoding specificity as implemented in Soar predicts that any such information would place a heavy burden on the retrieval process. Location information encoded in the actions of a chunk would also be present in the conditions, thus requiring that location cues be generated at retrieval time. This would not completely defeat the purpose, because the model could use the same kind process it now uses to generate and recognize images at retrieval time. However, more cues would have to be generated, requiring both more cognitive effort and more knowledge from which to generate them.

This shifts the emphasis to alternative strategies. One alternative might be to infer location from the nature of the target item. For example, applications often deposit different kinds of output into different windows. In such environments a reliable and easily-retained mapping from content to location should reduce the cost of clutter. Another strategy might be search, implying that reliable, easily-retained, and flexible searching tools also reduce the cost of clutter. More generally, the implication of episodic indexing is that access to hidden objects requires a reconstructive memory process that becomes more costly the more source information is stored with the target item. Thus users are likely to mitigate clutter by inferring location as needed, implying that interfaces to extensive information environments should support such inferences with direct, structured and learnable item-location mappings.

CONCLUSIONS

We propose that people store simple dynamic information in long-term memory as a matter of course, and use this information to index their environment. Our theory of episodic indexing makes two main claims:

- Pervasive and automatic encoding. People acquire large amounts of recognitional, episodic information about attention events, as a side effect of attention.
- Semantic, image-based retrieval. People retrieve this episodic information as a function of pre-existing knowledge that generates image cues when semantically appropriate.

The generality of these claims rest on the generality of their theoretical underpinnings. Soar's chunking mechanism (which predicts goal-based learning and encoding specificity) has been offered as a universal account of learning (Laird, Rosenbloom, & Newell, 1986; Newell, 1990), and our additional assumptions about the integration of episodic processing, attention, and comprehension are domain-independent. Thus episodic indexing may operate whenever people pay attention to what they are doing, and know the domain well enough to generate the right cues at the right time.

ACKNOWLEDGMENTS

This work was supported in part by a Krasnow Postdoctoral Fellowship, in part by the Wright Laboratory, Aeronautical Systems Center, Air Force Materiel Command, USAF, and ARPA under grant number F33615-93-1-1330, in part by the Office of Naval Research, Cognitive Science Program, Contract Number N00014-89-J-1975N158, and in part by the Advanced Research Projects Agency, DoD, monitored by the Office of Naval Research under contract N00014-93-1-0934. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies or endorsements, either expressed or implied, of Wright Laboratory, the Advanced Research Projects Agency, the Office of Naval Research, or the U.S. Government..

REFERENCES

Altmann, E.M. (1996). *Episodic Memory for External Information.* Doctoral dissertation, School of Computer Science, Carnegie Mellon University, Pittsburgh.

Altmann, E.M. & John, B.E. (in press). Modeling episodic indexing of external information. *Cognitive Science.*

Altmann, E.M., Larkin, J.H. & John, B.E. (1995). Display navigation by an expert programmer: A preliminary model of memory. *CHI 95 Conference Proceedings* (pp. 3-10). New York: ACM Press.

Anderson, J.R. (1990). *The Adaptive Character of Thought.* Hillsdale, NJ: Lawrence Erlbaum.

Bower, G.H. (1975). Cognitive psychology: An introduction. In W. Estes (Ed.), *Handbook of Learning and Cognitive Processes* (Volume 1). Hillsdale, NJ: Lawrence Erlbaum.

Brooks, R.E. (1977). Towards a theory of the cognitive processes in computer programming. *International Journal of Man-Machine Studies*, 9, 737-751.

Ericsson, K.A. & Kintsch, W. (1995). Long-term working memory. *Psychological Review,* 102, 211-245.

Howes, A. & Young, R.M. (1997). The role of cognitive architecture in modeling the user: Soar's learning mechanism. *Human-Computer Interaction*, 12, 311-343.

Johnson, M.K. & Raye, C.L. (1981). Reality monitoring. *Psychological Review*, 88, 67-85.

Johnson, M.K., Hashtroudi, S. & Lindsay, D.S. (1993). Source monitoring. *Psychological Bulletin*, 114, 3-28.

Just, M.A. & Carpenter, P.A. (1992). A capacity theory of comprehension: Individual differences in working memory. *Psychological Review*, 99, 122-149.

Kintsch, W. (1998) *Comprehension: A paradigm for cognition.* New York: Cambridge University Press.

Laird, J.E. Rosenbloom, P.S. & Newell, A. (1986). Chunking in Soar: The anatomy of a general learning mechanism. *Machine Learning*, 1, 11-46.

Lansdale, M.W. (1991). Remembering about documents: Memory for appearance, format, and location. *Ergonomics*, 34, 1161-1178.

Lewis, R.L. (1993). *An Architecturally-Based Theory of Human Sentence Comprehension.* Doctoral dissertation, School of Computer Science, Carnegie Mellon University, Pittsburgh.

Newell, A. (1990). *Unified Theories of Cognition.* Cambridge: Harvard University Press.

Ritter, F. E., & Young, R. M. (Eds.). (1998). Proceedings of the Second European Conference on Cognitive Modelling. Thrumpton (UK): Nottingham University Press. ISBN 1-897676-67-0

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

Petre, M. & Blackwell, A.F. (1997). A glimpse of expert programmers' mental imagery. *Empirical Studies of Programmers: 7th workshop*. New York: ACM Press.

Rieman, J., Young, R.M. & Howes, A. (1996). A dual-space model of iteratively deepening exploratory learning. *International Journal of Human-Computer Studies*, 44, 743-775.

Rosenbloom, P.S., Laird, J.E., & Newell, A., Eds. (1992). *The Soar Papers: Research on integrated intelligence*. Cambridge: MIT Press.

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*. Hillsdale, NJ: Lawrence Erlbaum.

Tulving, E. (1983). *Elements of Episodic Memory*. New York: Oxford University.

Underwood, B.J. (1977). *Temporal Codes for Memories: Issues and problems*. Hillsdale, NJ: Lawrence Erlbaum.

APPENDIX

Below we present a complete picture of the processing that occurs when the model probes with an image and retrieves episodic chunks. (This elaborates on the process described in the section, Encoding the Episodic Index.) In the general case, the episodic chunks retrieved by a probe will be of two kinds: those encoded during attention events and those encoded during (past) probe events. Only those acquired during probe events will contain source information in their actions (as discussed in the section, Episodic Memory in Soar). To determine whether the object of interest was actually attended in the past, the model computes the difference between the total set of episodic chunks retrieved and those representing probe events. The scenario below supposes that the model first probes for information about the SSN field, then actually attends to the field, then probes again.

At time t42, the model probes by placing an image in WM (A1) together with source information identifying the image as an image (A2).

```
production: imagine-ssn
   Conditions testing that it's relevant to know that
   an SSN field was seen.
  -->
  (^attended-to ssn)          A1
  (^imagined ssn)             A2
```

The model encodes an episodic chunk during the probe event, under the assumption of pervasive episodic encoding. Source information is included as a chunk action (A2) and hence also as a chunk condition (C2).

```
chunk: imagined-ssn     Chunk capturing a probe
event.
  (^attended-to ssn)          C1
  (^imagined ssn)             C2
  -->
  (^event ssn ^time t42)      A1
```

At time t43, the model actually attends to the SSN object, resulting in another episodic chunk.

```
chunk: attended-ssn     Chunk capturing attention
event.
  (^attended-to ssn)
  -->
  (^event ssn ^time t43)
```

Finally, at time t44, the model probes a second time (by firing imagine-ssn). This triggers the two episodic chunks described above, causing the following elements to enter WM.

```
(^attended-to ssn)          From imagine-ssn.
(^imagined ssn)             From imagine-ssn.
(^event ssn ^time t42)      From imagined-ssn.
(^probe t42)                From imagined-ssn.
(^event ssn ^time t43)      From attended-ssn.
```

From these elements the model can infer that an SSN field exists in the environment. The production that makes this inference is recall-seeing-object, below. Condition C5 (not reported in the section, Using the Episodic Index) effectively subtracts the set of probe events (containing t42) from the set of probe plus attention events (containing t42 and t43). The leading minus sign ("-") negates the subsequent condition, meaning that WM cannot contain an element matching that condition. In our scenario, this negated condition holds for at least one past event (t43). Thus the production matches, inferring that SSN was attended at some point in the past (A1).

```
production: recall-seeing-object
  (^attended-to <o>)          C1, <o> = ssn
  (^imagined <o>)             C2
  (^event <o> ^time <then>)   C3, <then>=t43
  (^time <now> != <then>)     C4, <now>=t44
- (^probe <then>)             C5
  -->
  (^recall-seeing <o>)        A1
```

Ritter, F. E., & Young, R. M. (Eds.). (1998). Proceedings of the Second European Conference on Cognitive Modelling. Thrumpton (UK): Nottingham University Press. ISBN 1-897676-67-0

Why Operators' Cognitive Models are Hard to Incorporate into Design: The Case of Human Reliability Models

René Amalberti
(Institute of Aerospace Medicine, Department of Defence, France)
11 Boulevard Hotel de Ville
93600 Aulnay-sous-bois, France
+33 (0)1 48 66 85 52
rene-a@mail.imaginet.fr

ABSTRACT

Operators' models, or equivalent end-user models, have became a standard prerequisite for most man-machine system design. Nowadays, the designer can chose among a great variety of models: behavioral models of performance, running competence models, and cognitive models are available in a large range of granularity from quasi-neuropsychological models of memory to framework models of dynamic cognition. However, despite -- or maybe because of -- that variety, modelling the operator is still an area of uncertainty within the industry, with multiple forms and meanings, and with a persistent feeling that these models, whereas they should be useful, are hard to incorporate into the design process.

This paper focuses on the development and use of cognitive models of human reliability for the design of complex systems, and tries to understand biases and limitations of their use within the industry. In that sense, the paper is more industry-oriented than research oriented. It is divided into three sections. The first section details the range of existing cognitive models of human reliability and proposes a classification of these models into four main categories: error production models, error detection and recovery models, systemic models, and integrated safety ecological models. The example of the Aviation Industry shows how difficult it has been in the recent past to incorporate the most advanced of these models into design, whereas the same Industry had long complained about the lack of availabilily of cognitive operators' models.

The second section tries to explain the reason for the relative failure. It shows the inter-dependency existing between the category of cognitive model, the safety paradigm, and the strategy for design. Severe drawbacks may occur each time a model is used with the wrong safety paradigm or the wrong strategy for design. It also shows that the more cognitively-based the model is, the less it is incorporated into design. The lack of education in psychology of designers, as well as the lack of a clear procedure for incorporating such models into design, are among the most important factors explaining this lack of success.

The third and last section points to new directions in cognitive modelling to improve the fit between operator modelling and design requirements.