Memory for Goals: An Architectural Perspective

Erik M. Altmann (altmann@gmu.edu)
Human Factors & Applied Cognition
George Mason University
Fairfax, VA 22030

J. Gregory Trafton (trafton@itd.nrl.navy.mil)
Naval Research Laboratory, Code 5513
4555 Overlook Ave., SW
Washington, DC 20375-5337

Abstract

The notion that memory for goals is organized as a stack is central in cognitive theory in that stacks are core constructs leading cognitive architectures. However, the stack over-predicts the strength of goal memory and the precision of goal selection order, while under-predicting the maintenance cost of both. A better way to study memory for goals is to treat them like any other kind of memory element. This approach makes accurate and well-constrained predictions and reveals the nature of goal encoding and retrieval processes. The approach is demonstrated in an ACT-R model of human performance on a canonical goal-based task, the Tower of Hanoi. The model and other considerations suggest that cognitive architectures should enforce a two-element limit on the depth of the stack to deter its use for storing task goals while preserving its use for attention and learning.

Introduction

The ability to decompose a complex problem into subgoals is to complex cognition roughly as the opposable thumb is to complex action. However, despite a generation of research into cognitive goal-based processing strategies like means-ends analysis, and into goal-based processing in artificial intelligence, we lack an adequate theory of how cognition manages the fine-grain goals of everyday tasks. The most common description we have is the stack — that is, a data type taken from computer science and interpreted as a description of cognitive processing.

The stack is an appealing descriptive tool because it elegantly captures the structure of everyday tasks that have to be decomposed to become achievable (Miller, Galanter, & Pribram, 1960). For example, traveling to a conference may entail taking an airplane, and whereas getting to the conference is in some sense the top-level goal, getting on the airplane has to happen first. Thus the order in which steps are planned (conference, then airplane) is opposite the order in which steps are executed (airplane, then conference). This is precisely the kind of reversal supplied by a stack, if goals are pushed in the order they are planned and popped in the order they are achieved (Figure 1). The theoretical inference has been that, because tasks are often decomposable in this way, the human cognitive architecture incorporates a stack to support such processing.

However, from a theoretical and empirical perspective, the stack is an idealized model of memory for goals that raises more questions than it answers. For example, the stack implemented in the ACT-R architecture invites questions about reactivity, dual-tasking, and memory for old goals (Anderson & Lebiere, 1998). The stack implemented in Soar (Newell, 1990) invites similar questions about similar issues (John, 1996; Rosenbloom, Laird, & Newell, 1988; Young & Lewis, in press).

We offer a critical analysis of the stack as cognitive theory. The supply of such theory is limited essentially to the ACT-R and Soar cognitive architectures, but in both cases the stack has been used to model a broad range of goal-based behavior. The role of the stack is typically to store goals associated with the task — *task goals* — like taking a flight or attending a conference. In both architectures, stacked task goals are immediately available for error-free retrieval with their order preserved regardless of when they were placed on the stack and regardless of how much or how little they were used since. These properties make the stack as poor a cognitive theory as it is useful a programming construct. As a theory it masks strategic adaptation and variation in goal management across tasks and individuals, and its usefulness as a programming tool simply adds to the problem by tempting the analyst. A better way to represent goals is to treat them like any other kind of memory element and then ask what (if any) supporting processes are necessary to account for how people remember them.

We start by comparing predictions of the task-goal stack to evidence from the literature. We then present an ACT-R model of the Tower of Hanoi, a task often taken to reveal the cognitive reality of the task-goal stack (Anderson & Lebiere, 1998; Anderson, Kushmerick, & Lebiere, 1993; Egan & Greeno, 1973). The model conforms to our proposal to treat task goals like any other kind of memory element, and shows that stacking them is not necessary to account for human performance. The model uses ACT-R's stack in prin-

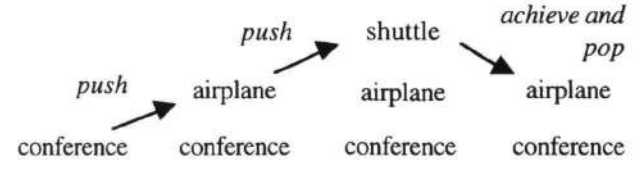

Figure 1: Pushing and popping task goals on a stack. Getting to the conference means flying, which in turn means shuttling to the airport.

cipled ways to support encoding and retrieval processes, and goes beyond a previous ACT-R model to predict errors as well as response-time data. We analyze this and other uses of the stack in ACT-R and Soar to propose a limit of two elements as an architectural constraint on stack depth.

The Stack: Predictions and Contradictions

A stack is an ordered set of elements, with the element at one end designated the top. The only operations defined on a stack affect this top element. A new element can be pushed onto the stack, covering the top element and itself becoming the top, or the top element can be popped off the stack, uncovering the element underneath it as the new top element (Figure 1).[1] Thus stack elements are ordered by age, with the newest element always on top.

One property of the stack is that once an element is pushed, it remains on the stack until it is popped. If the element is a task goal being pushed onto a stack as part of a planning process, then that goal remains on the stack as long as it takes to accomplish all task subgoals pushed on top of it. This is how task goals are stored on the ACT-R stack. The top goal controls the system's behavior, and when that goal is popped, the next older one takes over, no matter how old it is. Thus, ACT-R's stack predicts that memory for old goals is perfect.

Empirically, however, memory for pending goals (often referred to as prospective memory) is generally variable and strategic. Subjectively it seems quite common to embark on a simple errand or chore and midway through forget the purpose. In one study of intentions, actions to be carried out directly were remembered better than actions to be verified as carried out by someone else (Goschke & Kuhl, 1993). A stack-based account, in which memory for goals is perfect, predicts that memory should be at ceiling in either case.

A second property of the stack is that its elements are ordered last-in first-out, or LIFO. If these elements are again task goals pushed as they arise during planning, then the stack preserves that order perfectly, in addition to the elements themselves.

LIFO goal selection is not as pervasive as an architectural stack might predict. For example, goal selection in arithmetic can be highly variable both within and between subjects, and the variance is better explained by idiosyncratic goal selection strategies than by uniform LIFO ordering (VanLehn, Ball, & Kowalski, 1989). Similarly, goal selection in mundane tasks like VCR programming is guided by background knowledge and by simple difference-reduction strategies that make extensive use of perceptual information (Gray, in press). Finally, a phenomenon that defies LIFO goal selection is post-completion error, for example leaving the originals in the photocopier after taking the copies (Byrne & Bovair, 1997). Making copies is the top level goal and hence should be the first goal pushed and the last goal popped, with no stragglers. Thus if cognition had a task-goal stack, post-completion error would not be the common procedural error that it is.

A third property of the stack, at least as implemented in Soar and ACT-R, is that it depletes no resources directly affecting declarative memory. An element on the stack requires neither activation nor active maintenance to stay available and maintain its rank. Thus, memory for goals is not only perfect but free.

Several studies show that maintaining goals in memory does involve cognitive opportunity cost. For example, goal management strategies that reduce memory load have been linked to performance on Raven's Progressive Matrices in that better strategies allow more activation to be focussed on the underlying inference task (Carpenter, Just, & Shell, 1990). Similarly, working memory capacity has been linked to goal-selection errors in the Tower of Hanoi (Just, Carpenter, & Hemphill, 1996). Finally, fewer post-completion errors occurred when completed goals were allowed to decay, suggesting that activation is a limited resource that can shift among goals (Byrne & Bovair, 1997). The traditional task-goal stack, for example as found in ACT-R and Soar, simply cannot account for these results.

In sum, a variety of evidence suggests that memory for pending goals is strategic and effortful rather than perfect and automatic, contradicting the stack's fundamental predictions.

Storing Task Goals in Memory

If the analyst must do without the stack as a goal store, then other storage mechanisms must provide whatever critical functionality is lost. The natural stores to consider as substitutes are those that hold other kinds of declarative information, namely memory and the environment. These stores, together with supporting cognitive processes, would have to provide memory for goals and guidance for goal selection, and do so at reasonable cognitive cost.

To examine this possibility, we modeled human performance on a canonical goal-based task, the Tower of Hanoi. The structure of this task is such that the overall goal (of relocating a tower of disks) has to be recursively decomposed into subgoals (of moving individual disks). The task has been widely studied and modeled, generally on the assumption that people manage this decomposition using a stack.

Our starting point was a model by Anderson and Lebiere (1998) that uses ACT-R's stack in traditional fashion to store task goals. Their model (Traditional Goal Stack, or *TGS*) fits their data set (Anderson et al., 1993) remarkably well, with $R^2 = .99$ for response times on 4-disk trials and $R^2 = .95$ for 5-disk trials.

Our model (Memory as Goal Store, or *MAGS*) fits the same data set equally well, with $R^2 = .99$ for 4-disk trials and $R^2 = .95$ for 5-disk trials.

MAGS also goes beyond TGS to predict errors. TGS performs every trial perfectly in the fewest possible moves, which is 15 moves for 4-disk trials and 31 moves for 5-disk trials. In contrast, MAGS veers off the optimal path due to noisy retrieval of task goals from memory. In Monte Carlo simulations MAGS predicted a mean of 18.0 moves per 4-disk trial (3.0 errors) and 50.4 moves per 5-disk trial (19.4 errors). In the Anderson et al. (1993) data, empirical means

[1] The pop operation in Soar is generalized to allow popping any stack element along with all newer elements. This lets Soar react to events that, for example, achieve an older task goal and thus make all its subgoals on the stack moot. However, this more flexible pop operation does not materially alter the predictions of perfect, zero-cost memory for goals and goal order.

are 19.2 moves per 4-disk trial and 53.5 moves per 5-disk trial, both within 10% of our predictions.

Instead of a task-goal stack, MAGS incorporates commonly-recognized limitations on memory like decay and noise. Task goals are ordinary memory elements that have to be encoded and retrieved in ways that overcome these limitations. The processes that accomplish this goal management, adopted from an independently-constructed memory model (Altmann & Gray, 1999), account for response-time patterns in the data set. Moreover, instead of goal order being retained internally, goal selection is guided by cues from the environment. Finally, in addition to functioning without the stack, MAGS uses fewer free parameters than TGS.[2]

Figure 2 shows response-time data from 4-disk trials solved in the fewest possible moves. The empirical data (solid ink) are from Anderson et al. (1993), and the simulation data (dashed ink) are from MAGS. There are several patterns in the data that both MAGS and TGS account for, but we focus on how MAGS accounts for the large latency peaks at moves 1, 9, and 13, the latency valleys at even-numbered moves, and the small peaks at remaining moves.

Task Goals in the Tower of Hanoi

The algorithm used by the Anderson et al. (1993) participants combines the goal-recursive and perceptual strategies described by Simon (1975). The algorithm, shown in Figure 3, starts by focusing on the largest out-of-place disk, which we refer to as the *LOOP* disk. The LOOP disk is initially disk 4 and rests on peg A with target peg C. The algorithm first checks whether disk 3 blocks disk 4. (One disk *blocks* another if it is smaller than the other and rests on the other's

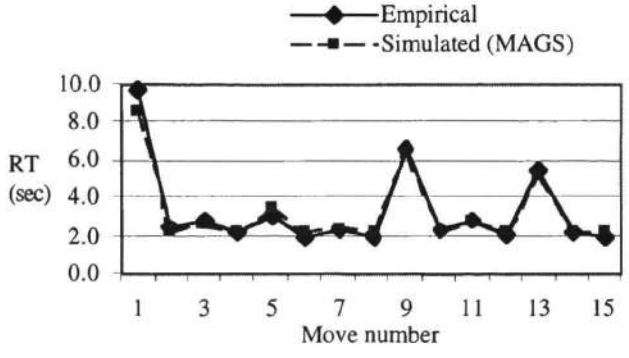

Figure 2: Tower of Hanoi response times (RTs) for 4-disk problems. Empirical RTs are from Anderson and Lebiere (1998) and simulated RTs are from the MAGS model.

[2] Default values are used for the ACT-R parameters of goal activation (W = 1.0), base-level learning (d = 0.5), and latency factor (F = 1.0). Transient activation noise (s = 0.3) and retrieval threshold (τ = 4.0) are taken from a model of a different but related goal-management task (Altmann & Gray, 1999), along with the mechanisms that depend on them. Encoding time (185 msec) is taken from ACT-R models of menu scanning and the Sperling task (Anderson, Matessa, & Lebiere, 1998). The one free parameter is move time, which we set to the same value as the Anderson and Lebiere model (2.15 sec). The model code is available for downloading at hfac.gmu.edu/people/altmann/toh.

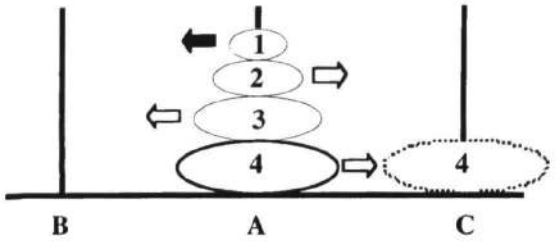

Figure 3: Planning a move in the Tower of Hanoi. Disk 4 must move to peg C (4:C), which entails 3:B, which entails 2:C, which entails 1:B as the move.

source or target pegs.) Disk 3 blocks disk 4 so it must be moved out of the way to peg B. This move is blocked by disk 2, so disk 2 must be moved to peg C. This move is blocked by disk 1, which must be moved to peg B (the filled arrow in Figure 3). Thus, moving disk 4 to peg C (which we designate *4:C*) entails a plan whose first step is 1:B.

A task goal in this context is an association between a disk and a target peg. The first task goal produced by the algorithm is 4:C, the second 3:B, and the third 2:C. The first of these (4:C) is readily inferred from the display by comparing the current state of the puzzle to the end state. However, the recursive task goals formulated in service of 4:C are not as easily available. These must be re-inferred after every move using the algorithm or they must be stored in memory and retrieved at the right time. Thus, for example, once 1:B has been made, the next move could be inferred anew by focussing on disk 4 again, then on disk 3, and then on disk 2. Alternatively, if a task goal for 2:C had been stored in memory during the first pass of the algorithm, and if that task goal could be retrieved now, then a second pass could be spared; memory would indicate 2:C.

The Anderson and Lebiere TGS model stores task goals on the ACT-R stack. For example, the model pushes 4:C, 3:B, and 2:C as it infers 1:B. After making 1:B the model examines the stack to see what move is on top. The top move is 2:C, which can now be made. If the top move had been blocked instead, the model would have pushed a new goal to move the blocking disk. For example, once 2:C is made, 3:B (underneath 2:C on the stack) is blocked because disk 1 is at peg B, so the model would push a goal for 1:A.[3]

The MAGS model, in contrast, treats a task goal like any other memory element. This poses two functional challenges. First, declarative memory elements in ACT-R, or *chunks*, decay over time. Thus an old goal, like 3:B in the example above, could be difficult to retrieve. Second, when one goal is achieved another must be selected. LIFO ordering is optimal for the Tower of Hanoi, so without the stack a useful selection order must come from some other source.

One of the patterns in the data in Figure 2 is that the peaks at moves 1, 9, and 13 decrease in size. These peaks correspond to the planning phase in the MAGS model, and the decrease is caused by successive shortening of the plan-

[3] Note that the retention interval is longer for 3:B than it is for 2:C, at both ends: 3:B is both encoded earlier and retrieved later. Despite this, both goals are equally available at retrieval time because the stack preserves them perfectly.

ning phase. For example, planning move 1 means starting with disk 4, whereas planning move 9 means starting with disk 3, because disk 4 is in place and can be ignored for the rest of the trial. With one fewer task goal to encode, planning move 9 is faster than planning move 1.

Encoding and Retrieving Task Goals

The first challenge in storing task goals in memory is to encode them so as to resist decay. In MAGS the process that accomplishes this is interleaved with the planning algorithm. Whenever the model recurses to focus on a blocking disk, it encodes a task goal for where it wanted to move the blocked disk. The encoding process strengthens a goal using cumulative mental rehearsal to a criterion that anticipates the retention interval (Altmann & Gray, 1999). This process accounts for the large latency peaks at moves 1, 9, and 13 (Figure 2). On these moves the model focuses on the LOOP disk and encodes a task goal for that disk, then focuses on the blocking disk and encodes a task goal for it, and so on, until it reaches a disk it can move. These task goals represent a plan that was time-consuming to encode in memory.

The second challenge to storing task goals in memory is to retrieve them in a useful order. In the Tower of Hanoi, perceptual cues afford the same ordering information as the stack does, and the MAGS model makes use of this information. For example, when the model uncovers a disk it asks itself, Where did I want to move the disk I just uncovered? The nature of the task is such that simple heuristics like this suffice to generate stack-like behavior.

Three simple heuristics produce optimal retrieval cues from perceptual information.[4] The first heuristic, *retrieve-uncovered*, applies when a move uncovers a disk on the source peg, and says simply to use the uncovered disk as a cue (as described above). The second heuristic, *retrieve-larger*, applies when a move empties a peg instead of uncovering a disk. It says to use the next larger disk as a cue, wherever that disk is now.

The third rule, *retrieve-next*, is needed because task goals can become "stale". Some disks move more than once while the LOOP disk is being unblocked, so an old task goal may indicate an obsolete target. For example, when the LOOP disk is 4, disks 4 and 3 each move once, disk 2 moves twice, and disk 1 moves four times before disk 4 is unblocked. The model encodes no task goals for disk 1, because disk 1 can always be moved. The first time disk 2 is used as a cue, its task goal indicates the correct target. The second time, however, its task goal incorrectly indicates the peg to which disk 2 was moved before. Retrieve-next recognizes this situation and tries to retrieve a task goal for the next larger disk.

When the model retrieves a task goal, it makes that move if it can. If instead the move is blocked, the model invokes the planning algorithm and focuses recursively on the blocking disk. However, the algorithm takes less time than it would have otherwise, because the cue disk is a better starting point than the LOOP disk for inferring the next move.

Some moves can be selected efficiently without retrieving goals from memory. An admissible heuristic in the Tower of Hanoi is *don't-undo*, which produces a single candidate for all even-numbered moves. A second heuristic, *one-follows-two*, is that when disk 2 moves, disk 1 should always be moved on top of it. In one closely-studied Tower of Hanoi session, the participant used both heuristics throughout despite being a novice (VanLehn, 1991). On these grounds we assume that the Anderson et al. (1993) subjects also used both heuristics, and incorporating them in our model improved its fit to the data. Don't-undo is responsible for the latency valleys at even-numbered moves, and one-follows-two reduces the small peaks at moves 3, 7, and 11 to less than the small peak at move 5 (Figure 2). Thus many moves in the Tower of Hanoi can be governed by simple perceptual information, reducing the burden on memory for goals and the need to posit a mechanism like the stack.

Roles for the stack

Our analysis suggests that a stack is not necessary to account for human performance in a canonical goal-based task. This undermines the traditional rationale for the architectural stack in theories like Soar and ACT-R, raising more general questions about the stack's role in cognitive theory. We suggest that the stack has several roles to play, but argue that it can and should be constrained to grow no more than two elements high at run time. The argument is based on a closer examination of our model and of learning mechanisms in both architectures.

Focusing Attention

MAGS uses the stack to focus attention on task goals. In ACT-R, the construct of attention is represented in part as activation, which affects the availabilty of chunks in memory. The more active a chunk, the greater the speed and likelihood of its retrieval (Anderson & Lebiere, 1998).

Activation in ACT-R has two components, source activation and base-level activation. *Source activation* captures the effect of context. Metaphorically, source activation acts as a spotlight on one chunk in memory. This focussed-on chunk, always the top chunk on the ACT-R stack, then radiates source activation to other, related chunks in memory. The link through which source activation spreads, which we refer to as a *cue*, is itself a chunk contained in the focussed-on and related chunks. Should the related chunk be the target of a retrieval, source activation is one component of its total activation. The other component is the target's *base-level activation*, which captures the effect of history. Base-level activation increases (strengthens) with use and decreases (de-

[4] Pseudo-productions for the three heuristics are:

retrieve-uncovered:
if the last move uncovered a disk,
then retrieve a task goal for the uncovered disk.

retrieve-larger:
if the last move emptied the source peg,
and there is a next larger disk,
then retrieve a task goal for the larger disk.

retrieve-next:
if a task goal was retrieved for a disk,
and it says to move that disk to a peg,
and the disk is already on that peg,
and there is a next larger disk,
then retrieve a task goal for the larger disk.

cays) without. Noise in memory is represented as transient fluctuation in total activation.

The need to focus attention arises because the amount of source activation is fixed and divided equally among the cues in the focussed-on chunk. Thus the more cues there are, the less effective is any one of them in activating a given target. The stack enables strategic control over attention by allowing the system to push a subgoal containing cues selected so that all available source activation reaches the target. When retrieval succeeds (or the attempt is abandoned), the subgoal pops and the architecture restores the greater context by refocussing on the older goal. The hypothesis is that with preparation (creating and pushing a subgoal), cognition can maintain context through intervals of focussed attention.

In MAGS, the need to focus attention arises because the model keeps a variety of state information in the focussed-on chunk. This information includes the source and target pegs of the previous move and the disks that were covered and uncovered. This information interferes with task-goal retrieval by drawing source activation away from the cue disk. To overcome this interference, the model pushes a subgoal containing only the cue disk, which then directs all available source activation to the target task goal. After retrieving the task goal the model pops the retrieval subgoal, causing ACT-R to refocus on the chunk below it on the stack. This older chunk contains the same state information it did before, now augmented with the target peg from the task goal.

The encoding process also needs to focus attention, because it needs to simulate the retrieval context to know when to stop. The encoding process functions by strengthening the task goal's base-level activation. To determine when this activation is high enough, the encoding process pushes a subgoal containing only the retrieval cue and attempts a retrieval.[5] If this retrieval succeeds, the encoding process exits because it has verified that the task goal's retrieval demands are likely to be met.

In sum, ACT-R's stack plays a key role in MAGS, not directly by storing task goals but indirectly by supporting the processes that encode and retrieve them. Thus the stack remains a core construct, but at a finer grain of analysis than its traditional interpretation as a theory of memory for goals.

Learning

The stack plays other roles in both ACT-R and Soar, one of which is learning of productions. Productions specify a conditional transformation on the current focus of attention, or *state*. Half the production (the condition) specifies the input state to the transformation, and the other half (the action) specifies the output state. Learning a new production therefore entails retaining a memory of the input state while the output state is generated. The stack is a natural way to implement this learning. For example, Soar models often learn by pushing a duplicate of the input state onto the stack and transforming the duplicate into the output state. When the transformation is done, both states are available as blueprints for the new production, whose conditions are patterned on the input state and whose actions are patterned on the output state. The new production caches in one step the sequence of steps that transformed the state

Several learning mechanisms in ACT-R also make use of the stack. First, in declarative learning a chunk enters long-term memory when popped off the stack, on the premise that any information deliberately focussed on leaves a trace in memory. Second, production learning is also deliberate in that it requires pushing a special kind of subgoal to create a new production. In both cases, the stack maintains a previous context while the system focuses temporarily on learning, much as the stack maintains an older context while MAGS focuses attention. Third, in learning the utility of productions ACT-R credits a production pending the outcome of the goal created by that production. This credit assignment requires that the old goal stay available over time. Fourth, the stack has been used to control the associations learned between cues and facts (Anderson & Lebiere, 1998, Ch. 9) in a manner also related to attention focussing.

A two-high stack suffices for the architectural learning processes outlined above. Two states support symbolic learning, and support subsymbolic learning for one unit of procedural knowledge or one declarative association at a time. A two-high stack is also sufficient for modeling complex, goal-based behavior in a range of domains, including programming (Altmann & John, in press), exploratory learning (Howes & Young, 1996; Rieman, Young, & Howes, 1996), and serial attention (Altmann & Gray, 1999), as well as the Tower of Hanoi. Indeed, representation decisions concerning goals and goal selection are simpler when there is no open-ended choice for the analyst to make about what stack depth best accounts for the participant's state of mind at a given instant.

Conclusion

The stack captures the means-ends structure of many tasks, but, we argue, is a poor explanation of cognitive performance on those tasks. It over-predicts memory for goals and goal order and under-predicts the cost of encoding and retrieval. We made this case in part with reference to data on prospective memory, goal selection order, and the relationship between goals and working memory.

We also made the case with an analysis of the Tower of Hanoi. This task is traditionally assumed to induce cognitive goal stacking, and the TGS model of Anderson and Lebiere (1998) incorporates this premise. However, our MAGS model improves on TGS in several ways. First, it dispenses with the task-goal stack, which we argue is cognitively implausible, while achieving an equally good fit to response-time data. Second, in place of the task-goal stack MAGS incorporates independently-constrained memory mechanisms (Altmann & Gray, 1999). Third, MAGS goes beyond TGS to predict errors as well as response times. Fourth, MAGS uses fewer free parameters at the subsymbolic level. MAGS thus offers an accurate, well-constrained, and broad account of empirical data, increasing our confidence that performance in the Tower of Hanoi reflects the reality of memory and attention rather than the reality of the task-goal stack.

The use of perceptual cues in MAGS is congruent with routine use of such cues in behavior that appears plan-like to

[5] This test retrieval is scaled to the size of the disk, to model our assumption that practiced subjects will adapt to the fact that task goals for larger disks have longer retention intervals.

the observer (Suchman, 1987; Vera, in press). It also predicts that tasks will be much more difficult when goal order is relevant but perceptual cues to order are not available. In such situations goal order must be explicitly encoded in memory, requiring more time and possibly the knowledge and techniques underlying skilled memory (Ericsson & Kintsch, 1995). If selection order is not relevant, then selection should be guided by factors like the recency and frequency with which pending goals were checked or rehearsed.

Whereas the stack is a poor theory of memory for task goals, a limited stack is a natural basis for sub-theories of attention and learning. Moreover, a limit of two elements has proven not only workable but parsimonious in several models of goal-based behavior, and thus may be appropriate to enforce in cognitive architectures like ACT-R and Soar.

Cognition may of course still provide specialized, stack-like support for means-ends processing under circumstances that we have not anticipated. In favor of this view, one could argue that memory has likely adapted to the means-ends structure of the environment as it has adapted to the structure of the environment in other ways (Anderson, 1990). It remains for future research to identify control processes or memory subsystems that function like a stack under appropriate circumstances, or to add to the evidence that pending goals are stored in memory like any other information.

Acknowledgments

The first author is supported by Air Force Office of Scientific Research grant F49620-97-1-0353, and the second author by funds from the Office of Naval Research. We thank B. D. Ehret, W. T. Fu, W. D. Gray, I. R. Katz, A. W. Lipps, S. L. Miller, L. D. Saner, M. J. Schoelles, C. D. Schunn, S. B. Trickett, S. Varma, and an anonymous reviewer for their comments.

References

Altmann, E. M. & John, B. E. (in press). Episodic indexing: A model of memory for attention events. *Cognitive Science*.

Altmann, E. M. & Gray, W. D. (1999). Serial attention as strategic memory. *Proceedings of the twenty first annual conference of the Cognitive Science Society*, to appear.

Anderson, J. R. (1990). *The adaptive character of thought.* Hillsdale, NJ: Erlbaum.

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

Anderson, J. R., Kushmerick, N., & Lebiere, C. (1993). The Tower of Hanoi and goal structures. In J. R. Anderson (Ed.), *Rules of the mind.* Hillsdale, NJ: L. Erlbaum. 121-142.

Anderson, J. R., Matessa, M., & Lebiere, C. (1998). ACT-R: A theory of higher-level cognition and its relation to visual attention. *Human-Computer Interaction,* **12**, 439-462.

Byrne, M. D. & Bovair, S. (1997). A working memory model of a common procedural error. *Cognitive Science*, **21**, 31-61.

Carpenter, P. A., Just, M. A., & Shell, P. (1990). What one intelligence test measures: A theoretical account of the processing in the Raven Progressive Matrices test. *Psychological Review*, **97**, 404-431.

Just, M. A., Carpenter, P. A., & Hemphill, D. D. (1996). Constraints on processing capacity: Architectural or implementational? In D. M. Steier & T. M. Mitchell (Eds.), *Mind matters: A tribute to Allen Newell.* Hillsdale, NJ: L. Erlbaum. 141-178.

Egan, D. S. & Greeno, J. G. (1973). Theory of rule induction: Knowledge acquired in concept learning, serial pattern learning, and problem solving. In L. W. Gregg (Ed.), *Knowledge and cognition.* Hillsdale, NJ: Erlbaum. 43-103.

Goschke, T. & Kuhl, J. (1993). Representation of intentions: Persisting activation in memory. *Journal of Experimental Psychology: Learning, memory, and cognition,* **19**, 1211-1226.

Gray, W. D. (in press). The nature and processing of errors in interactive behavior. *Cognitive Science.*

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

Howes, A. & Young, R. M. (1996). Learning consistent, interactive and meaningful task-action mappings: A computational model. *Cognitive Science,* **20**, 301-356.

John, B. E. (1996). Task matters. In D. M. Steier & T. M. Mitchell (Eds.), *Mind matters: A tribute to Allen Newell.* Hillsdale, NJ: L. Erlbaum. 313-324.

Miller, G. A., Galanter, E., & Pribram, K. H. (1960). *Plans and the structure of behavior.* New York: Holt, Rinehart, & Winston.

Newell, A. (1990). *Unified theories of cognition.* New York: Harvard.

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. (1988). Meta-levels in Soar. In P. Maes & D. Nardi (Eds.), *Meta-level architectures and reflection.* Amsterdam, The Netherlands: Elsevier Science Publishers B.V. 227-240.

Simon, H. A. (1975). The functional equivalence of problem solving skills. *Cognitive Psychology*, **7**, 268-288.

Suchman, L. A. (1987). *Plans and situated action: The problem of human-machine communication.* New York: Cambridge University Press.

VanLehn, K. (1991). Rule acquisition events in the discovery of problem solving strategies. *Cognitive Science*, **15**, 1-47.

VanLehn, K., Ball, W., & Kowalski, B. (1989). Non-LIFO execution of cognitive procedures. *Cognitive Science,* **13**, 415-465.

Vera, A. H. (in press). By the seat of our pants: The evolution of research on cognition and action - Review of *Plans and situated action. Journal of the Learning Sciences.*

Young, R. M. & Lewis, R. L. (in press). The Soar cognitive architecture and human working memory. In A. Miyake & P. Shah (Eds.), *Models of working memory: Mechanisms of active maintenance and executive control.* New York: Cambridge University Press.