Using Diverse Cognitive Mechanisms for Action Modeling

John E. Laird (laird@umich.edu)
Joseph Z. Xu (jzxu@umich.edu)
Samuel Wintermute (swinterm@umich.edu)
University of Michigan, 2260 Hayward Street
Ann Arbor, MI 48109-2121 USA

Abstract

Predicting the results of one's own actions is a powerful cognitive capability that can aid in determining which action to take in a given situation. In this paper, we describe a task-independent framework based on the Soar cognitive architecture in which rules, episodic memory, semantic memory, mental imagery, and task decomposition are available for predicting an action's consequences. We include results from two domains and make predictions for human behavior based on these results.

Keywords: Action modeling; prediction; cognitive architecture

Introduction

When faced with a decision between alternative actions, an intelligent agent may have sufficient knowledge to immediately determine which choice is best. However, in situations where directly available knowledge is insufficient or in conflict, an agent can often use predictions of how its actions will change the environment to make its decision. We call the knowledge used to make such a prediction an *action model.* Using this approach to make a decision typically involves the following steps:

1. Choose one of the alternative actions to evaluate.
2. Create an internal representation of the situation.
3. Apply the action model to the internal representation to generate a prediction.
4. Repeat for all other actions.
5. Choose the action that leads to the best predicted state.

This approach to decision making is ubiquitous in humans (de Groot, 1965; Newell & Simon, 1972) and has been used throughout artificial intelligence (AI) systems, where the agent internally simulates multiple steps into the future. A critical ingredient in this process is the action model: the means by which the results of actions are predicted. Action modeling is important because it allows an agent to move beyond reactive behavior – an agent can plan and deliberate about the implications of its actions before choosing one.

Historically, AI systems have used rule-like structures as action models, such as STRIPS operators (Fikes & Nilsson, 1972). Cognitive science research has addressed action modeling, but it has typically been isolated within specific cognitive processes, such as mental imagery (Johnson, 2000; Wintermute & Laird, 2009) or episodic memory (Atance & O'Neill 2005, Schacter & Addis 2007).

Rather than focus on one particular approach to action modeling, we investigate the problem in general. We propose that different combinations of memory and processing systems can be used for action modeling, and that domain characteristics and the agent's knowledge determine which mechanisms are used for a specific task. The mechanisms we propose include rule-based procedural knowledge, episodic knowledge, semantic knowledge, mental imagery, action decomposition, and arbitrary combinations thereof. These mechanisms vary along many dimensions including generality, reportability, learnability, computational expense, and the types of problems where they are appropriate. Forbus & Gentner (1997) have previously posited a similar diversity of processing to support mental models, although they did not focus on detailed architectural mechanisms as we do here.

Included in our work is task-independent knowledge that dynamically combines these mechanisms, implemented within Soar (Laird, 2008). Soar has the requisite representational capabilities to support the diverse forms of memories, processing units and knowledge required for action modeling. In the next section, we give an overview of Soar and our approach to using action models in support of decision making. This is followed by descriptions of the different forms of action modeling, with demonstration of them on a simple blocks world task. We then demonstrate them together on a simple board game, and analyze their relationship to human behavior.

Framework for Action Modeling in Soar

Figure 1 shows the structure of Soar, including its long-term and short-term memories and processing components. Working memory is a shared, symbolic memory that maintains the agent's primary representation of the current situation. Long-term symbolic memories hold procedural, semantic, and episodic knowledge, which are retrieved based on either the total contents of working memory (for

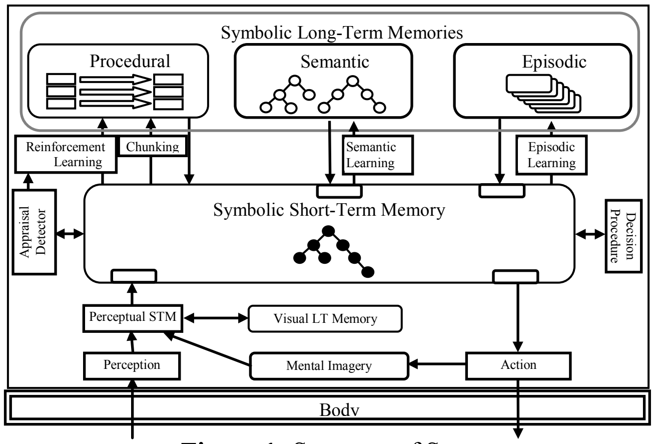

Figure 1: Structure of Soar

procedural) or cue structures created in working memory (for episodic and semantic). Soar has a non-symbolic, spatially-based perceptual short-term memory (STM) from which symbolic information can be extracted into working memory. This memory is the medium of mental imagery.

Behavior in Soar is driven by rules stored in procedural memory. Rules that successfully match the contents of working memory fire in parallel. *Operators* are the locus of sequential behavior in Soar and only a single operator can be selected at a time.[1] Operators are implemented via rules that propose, evaluate, and apply them. Rules that propose and evaluate an operator create *preferences,* while rules that apply an operator modify elements in working memory when that operator is selected.

If there is insufficient knowledge to select or apply an operator, an *impasse* arises, and a substate is created. Within the substate, operators can be proposed, selected, and applied to resolve the impasse. A side effect of resolving an impasse in a substate is that Soar builds a rule that summarizes the processing in the substate. This process is called *chunking*. The learned rule fires in similar situations so that the same impasse is avoided in the future.

Conceptually, operators are either *external*, in that they initiate action in the environment, or *internal*, in that they change the internal state of an agent. Throughout this paper, we call external operators *actions,* so that an *action model* refers to an internal model of the changes that result from the application of an external operator.

Figure 2 shows how action modeling arises in Soar. When an agent is unable to make a decision using its directly available knowledge, it internally simulates the effects of proposed actions to aid in decision making. In this example, the agent is attempting to create a stack of blocks, with A on B, B on C, and C on the table. In the upper left corner of the figure, the agent's state is shown, with the lower half corresponding to a representation of the problem state as it might be in the agent's perceptual short-term memory. The top half of the state shows the symbolic relations that the agent extracts from perception, and it is these relations that

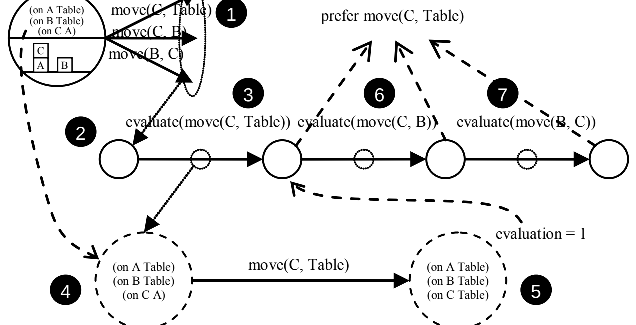

Figure 2: Soar processing using an action model.

[1] Operators in Soar correspond most closely to rules in ACT-R (Anderson, 2007); however, operators in Soar provide a richer representation for organizing action than do rules in ACT-R because of the independent representations of knowledge (as rules) for proposing, selecting, and executing the actions associated with an operator.

are available in working memory.

We assume the agent has sufficient knowledge to propose the three legal actions for this state: move B onto C, move C onto B, and move C onto the table. However, there are no rules to create preferences, so an impasse arises (1), and Soar automatically creates a substate (2).

To resolve this impasse, the agent tries out each proposed action on a copy of the state and then evaluates the quality of the result. Task-independent knowledge (TIK), encoded as rules, carries out this strategy. The *only* additional task-dependent knowledge required in this processing are action models and state evaluations, both of which can use the various forms of knowledge presented below.

As shown in Figure 2, following the impasse, operators are selected (at random) to evaluate the actions. In the example, move C to the table is evaluated first (3). In this case, the agent does not have rules to evaluate this action directly, and thus, another impasse arises. In the resulting substate (4), the TIK copies the contents of the original task state and uses a model of the action being evaluated to predict the resulting state. Once this state is computed (5), the agent must also have some knowledge (usually encoded as rules) for evaluating it. In this case, we use an evaluation that counts the number of blocks in their desired positions, which assigns the state an evaluation of 1. The creation of this evaluation terminates the evaluate operator, which is followed by the selection of operators to evaluate the remaining actions (6, 7). When all the evaluations are computed, preferences are created for the actions, leading to the selection of the action to move C to the table, and resolving the first impasse. The action is then performed. Chunking learns rules for evaluating each of the actions (from the substates where the action modeling occurs), and for creating the preferences based on those evaluations.

Different Forms of Action Modeling

In this section, we describe how action modeling can be implemented using different processing and memory systems, with the blocks world serving as an example.

Procedural Knowledge

The most direct way to encode an action model in Soar is as rules. These rules test features of the state, features of the selected action, and that the state is an internal copy of the task state. They modify the internal copy in the same way the external action would modify the real state. For complex actions, the model can be implemented with multiple rules that fire in parallel and/or in sequence.

Episodic Memory

Soar has an episodic memory that automatically stores "snapshots" of working memory over time (Nuxoll & Laird, 2007). Soar's episodic memory is an idealization of human episodic memory, and emphasizes basic functionality, such as efficient storage and associative retrieval of temporally organized episodes. For action modeling, episodic memory requires that the agent has a previous experience when the action being considered was applied in the environment.

The agent can then use its memory of that experience to make a prediction as to what will happen when the operator is applied to a similar situation (Xu & Laird, 2010).

When episodic memory is used, the behavior of the agent is as follows. The first time the agent gets to the point where the action is selected in Figure 2, an impasse would arise because there is no rule to apply the action. In the resulting substate (not shown in Figure 2), the TIK for using episodic memory selects an operator which creates a cue consisting of the task state with the action selected, in an attempt to retrieve a similar previous episode. Once the cue is created, the episodic memory system retrieves the most recent, best match to the cue and reconstructs it in working memory. If no match is found, then this approach to action modeling fails, and the agent must either try other methods, or assign a default evaluation value to the action being evaluated. Chunking does not create rules to summarize processing in substates where episodic memory retrieval failed.

If the retrieval is successful, the agent then retrieves the following episode. The agent continues retrieving subsequent episodes until it finds one where the action is no longer selected, which indicates the action has terminated. The agent then compares the task state in that episode to the current task state and modifies the internal copy of the task state to reflect any changes. Chunking creates a rule that summarizes the processing, so that in the future, the retrievals are not required.

Figures 3 and 4 compare results for using the rule-based versus the episode-based approaches to action modeling. Both figures show the progression of performance across four identical trials of the blocks world problem described above, and both use log scales for the y-axis. Figure 3 shows the number of external actions that the agent takes to solve the problem, while Figure 4 shows the number of decisions (processing cycles in Soar). These results are not intended to precisely model human behavior (for example, we are not including time for perception or motor actions); however the comparisons should be meaningful in predicting qualitative differences across methods and trials.

In Figure 3, the top line shows the average performance of an agent using episode-based action modeling where episodes are not learned, so that a random selection is always made. The next line shows the performance when episodes are being learned. Initially there are no relevant episodes, so the selections are random, but with experience, the episodes accumulate and the agent's performance improves as it is able to correctly predict future states and select the correct action, until finally it achieves optimal performance. Even the first trial gets some improvement from learned episodes. The bottom line shows the performance with the rule-based action model, which always makes the correct predictions.

Figure 4 shows the performance in terms of decisions, not just external actions. The top line corresponds to the steps required when episodes are not learned. The next line shows the performance as episodes are learned. The dashed line that starts at the same point for trial 1 shows that when chunking is used with episodic memory, it eliminates the need for episodic retrievals over time as the agent learns action models based on rules that replace those based on episodic memory. The agent eventually learns rules that choose actions directly, eliminating the need for action models. Thus, there is a combined gain with episodic memory improving solution quality, and chunking improving the efficiency of the problem solving process. Note that external actions take orders of magnitude more time to execute than internal reasoning steps, so the differences are more pronounced in real environments.

The next line shows the performance for the rule-based action model without chunking, which serves as the optimal base line for action modeling. The final line shows the impact of using chunking with the rule-based action model, where after one trial, rules are learned that eliminate the need for the action model. As these figures show, in only a few trials, the combination of episodic memory and chunking converts an agent with little task knowledge into one that solves the problem in few actions (due to episodic memory-based action modeling), while eliminating the need for purely internal decisions (due to chunking).

Semantic Knowledge

Whereas episodic memory is based on specific experiences, semantic memory consists of decontextualized facts – such as knowledge about objects and their structure, independent of when they were experienced. This makes semantic knowledge more difficult to learn than episodic knowledge,

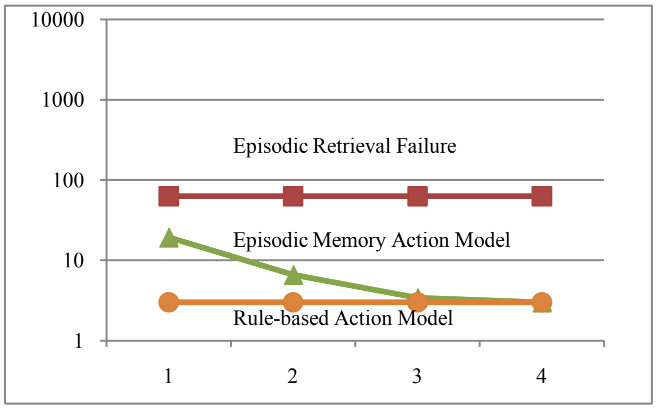

Figure 3: External actions taken across multiple trials.

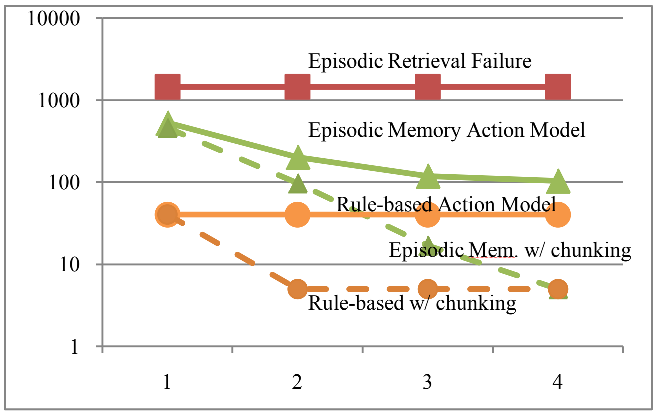

Figure 4: Total decisions taken across multiple trials.

but more useful across a variety of tasks. Soar as yet does not have a theory of how semantic memories are automatically learned, and instead Soar agents must deliberately store semantic data they encounter.

The use of semantic memory for action modeling is analogous to the use of episodic memory – when there is no action model encoded as rules, an impasse arises, and in the resulting substate, an operator is selected which queries semantic memory to retrieve knowledge that can aid in predicting the result of applying that action. Semantic memory covers a broad range of knowledge, and one can imagine many ways it can aid in action modeling. For example, the fact boiling kettles are hot can be useful when predicting the consequence of touching one. Here, we use declarative instructions that specify how to modify the internal task state to model the action.

To use semantic memory, the agent selects an internal operator that initiates a retrieval for instructions related to the action being evaluated. If the relevant instructions are retrieved, TIK selects the "interpret" operator, whose purpose is to apply the instructions to the copy of the task state. The interpret operator is not implemented directly in rules, but leads to a substate where operators are selected and applied for each of the instructions. The processing in the substate allows for arbitrarily complex implementations of instructions, and is similar in spirit to how declarative instructions are used in ACT-R (Anderson 2007; Best & Lebiere 2003); however, in those cases the instructions are interpreted to control the execution of a task, while here they are used to model the execution of an action.

The format of declarative instructions is like that of an imperative programming language or a recipe. We have developed task-independent declarative representations for common control flow instructions and state modifications. In the blocks world example, instructions specify additions and deletions of predicates. The rules that interpret those instructions assume a specific representation of predicates in working memory. Figure 5 shows the instructions for moving a block. When using semantic memory, the number of decisions decreases after one trial, as chunking creates action model and action selection rules.

Mental Imagery

Mental imagery involves the maintenance of a separate memory structure grounded in perception, which represents objects and their spatial properties. While the contents of the memory is mostly created bottom-up from perception, an agent can create new "imagined" structures and manipulate them by operations such as translation, rotation, and scaling, as well as simulate complex motions, such as the path of a car (Wintermute, 2009). The agent can extract spatial predicates from perceptual memory, such as the relative positions of objects and whether they collide. When applied to perceived structures, this can be used to create the initial symbolic representation of the problem. When applied to imagined structures, symbolic consequences of actions can be predicted. The use of mental imagery for action modeling is restricted to actions that involve spatial motion, or actions that can be mapped onto such motion.

```
Move-block(blk, dest):
  1. Del-predicate ontop(blk, x) ∀ x ≠ dest
  2. Add-predicate ontop(blk, dest)
```

Figure 5: Instructions encoded in semantic memory.

As in the use of episodic and semantic memory, mental imagery is employed when there are no rules for an action model, and an impasse arises. Mental imagery takes advantage of the spatial representation and maps the action to be modeled onto imagery operations. Making the connection between the action and mental imagery operations can involve accessing knowledge in semantic memory, or such knowledge can be encoded in rules. In our example, the agent knows that to move a block, it should imagine it centered on top of the destination block. Once the perceptual memory has changed, relevant predicates can be extracted, creating a symbolic description of the situation that serves as the resulting state.

Mental imagery involves processing that cannot be analyzed by chunking because the results of the processing are not uniquely determined by the symbolic structures available in working memory. Therefore, chunking does not create rules that summarize mental imagery processing. This is similar to ACT-R avoiding rule compilation for processing over external interactions (Anderson, 2007).

Although not as general as the other methods, mental imagery has wide applicability because of the ubiquity of spatial problems. Imagery-based action models are effective in a range of problems, from simple tasks in the blocks world (Wintermute & Laird, 2009) to complex tasks such as path planning for cars (Wintermute, 2009).

Action Decomposition

The final alternative approach is to model an action by decomposing it into simpler actions that can be modeled using any of the approaches described above. In Soar, hierarchical operator decomposition is ubiquitous, and arises when complex operators are selected, and then implemented in substates by simpler operators. In the blocks world example, when move-block is selected, it can be decomposed into pickup-block and put-down-block actions. When these actions are selected, any of the previous methods can be used as models for them, including further decomposition. One typical use of action decomposition is to take an action that involves complex spatial interactions and decompose it into simpler parts until those parts can be mapped onto imagery operations. Chunking will create rules for the action model of a complex operator as long as mental imagery was not used in any substate processing.

A Policy for Controlling Action Modeling Approaches

We have presented these action modeling approaches as alternatives, with no attention to when each would be used in an integrated agent. Inherent to Soar is that it uses rules for action modeling if they are available. That is the default behavior and it is not under control of the agent. When rules are not available, an impasse arises, and in the ensuing substate, operators are proposed for the alternative methods, as well as any operators that decompose the selected action. This structure introduces an extra level of deliberation, which adds flexibility at minimal cost to the agent (the results in Figure 4 are without this additional layer). Although it may be possible for an agent to learn when best to use each method, that could be a difficult learning problem and we leave it to future research. As an alternative, we encoded a simple ordering preference for these approaches in the TIK and use this method in the board game demonstration below.

Integrated Demonstration

To provide additional illustration of how these approaches work, both independently and in unison, we present an agent that plays a simple board game, shown in Figure 5. In this game, the agent must slide the hexagonal marker on the left along the directional paths to numbered nodes until it gets to the end (node 10). As the marker slides along a path, it may touch one of three different objects, labeled X, Y, and $. If the marker hits an object, the agent gets points. The agent has semantic knowledge that the $ is worth 20 points, but does not initially know the values of the other objects (X is worth 10 points and Y is worth 5). The goal is to get to the end with the highest possible score, which is achieved via path A, C, F, H, I, K. We assume that the agent can sense the marker position, the paths, and the objects, but it does not a priori know whether the marker will hit a nearby object as it slides along a path.

To perform the task, the marker starts at position 1, and the agent is faced with making a decision to take path A or B. To make this decision, the agent will attempt to predict the result of each move. At this point, the agent does not have any action model rules, nor does it have any episodes or relevant information in semantic memory. However, it can use mental imagery to imagine moving the marker along each of the paths. Mental imagery predicts that if it moves along A, it will intersect with object X, while for B, it will intersect with Y. In both cases, it does not know how encountering those objects will affect its score, so it chooses at random. We assume it picks path B. It executes that action, encountering Y and getting 5 points.

Once at 3, the agent picks path D to get to 4. Here, the decision is between going along path E or F. This time, after it uses mental imagery to detect that it will encounter object Y, it then uses episodic memory to recall that the last time it encountered object Y it received 5 points. When it considers path F, it uses imagery to predict it will encounter object $, and then semantic memory to predict that it will receive 20 points. Based on these evaluations, it chooses path F. It receives 20 points, moves to 6 and then 7. At this point, it uses a combination of mental imagery and episodic memory to predict the result of moving to 8 (10 points). In imagining moving to 9, imagery shows that it will not encounter Y, so it will get a score of 0. It selects moving to 8, and then finishing by moving to 10, getting a total score of 35.

The next time the agent plays the game, it uses episodic memory to predict the results of the paths it took the first time (B, F, I). Since it has no episodic memories of moving on paths A, E, and J, and cannot chunk over imagery action models, it must continue to use imagery for those paths.[2] Thus, in its second attempt, it will use imagery and episodic memory to predict a 10 score for A, while it will use only episodic to predict a score of 5 for B. Similar use of imagery and episodic memory will be used at nodes 4 and 7. As a result, the optimal path is taken, resulting in a score of 40.

Figure 7 shows the progression of how the agent's decisions are distributed across using imagery versus episodic memory over multiple trials. The highest line shows the total number of internal reasoning steps. The bottom two lines are the number of decisions that involve imagery and episodic memory operations. In the first trial, imagery dominates as the agent has no prior experiences it can draw on. In the second run, the agent must still use imagery for those cases where it has not taken a path, but it uses episodic memory for those cases where it had prior experiences. Although not evident in the graph, chunking replaces the use of semantic memory with a rule. For the third run, chunking decreases the total number of steps by eliminating the use of episodic memory. In the final trial, some imagery is still required for those paths the agent never actually tried, and episodic memory is no longer used as it has been replaced by rules learned through chunking.

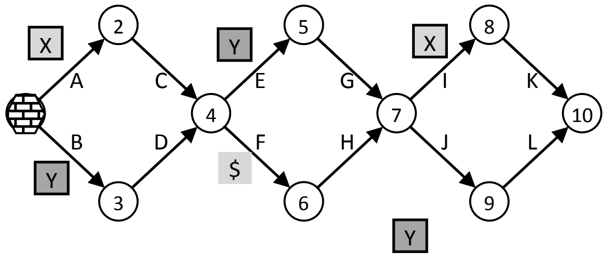

Figure 6: Board game task performed by agent.

Predictions

From these examples and an understanding of the approach, we can make some predictions about the behavior

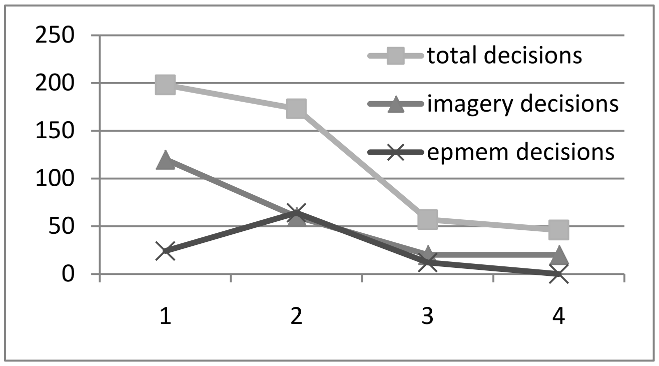

Figure 7: Agent performance over multiple trials.

[2] Soar's episodic memory does not capture subgoal processing, so the agent has no episodic memories of previous predictions. Otherwise, these steps could also be removed.

of an agent with the capabilities we described.

In a spatial environment, an agent initially relies on mental imagery for action modeling (and semantic knowledge if it is available). As the agent gains experience, it switches to using episodic memory when possible. With further experience, rules learned via chunking replace episodic memory, and eventually rules are learned that choose actions directly, eliminating action modeling.

Concurrent with learning, the agent's ability to report on its internal reasoning should change, as different structures become available in working memory (which is the basis for our predictions about reporting). Initially, for spatial problems, the agent can report imagining spatial situations, which then transitions to reports of using episodic memory (things it "remembers"). When using semantic memory, it can report on the instructions and facts it is using (things it "knows"). With practice, the agent loses the ability to report on its reasoning as intermediate structures are no longer generated in working memory and processing is done purely with rules. The rules produce behavior without the creation of a declarative trace that the agent can report.

As shown in Figure 7, our model predicts there are also changes over time in terms of which mechanisms are used in action modeling, and thus decision making. The obvious prediction is that in humans the brain areas used for action modeling, and thus decision making, will change based on characteristics of the task (whether it is spatial or symbolic) and a subject's experience (whether it has access to relevant semantic, episodic, or procedural knowledge).

Conclusions

The major claim of this paper is that intelligent agents, including humans, have a variety of available mechanisms that can be used to predict the results of their actions in service of decision making. A related claim is that internal prediction does not occur in any specific architectural module, but results from a combination of characteristics of the domain, the agent's background knowledge, prior experience, and the agent's available memories and processing elements. We have demonstrated two agents in two domains using rules, episodic memory, semantic memory, mental imagery, and action decomposition for action modeling. Although the domains are simple, the results predict significant changes in behavior as knowledge accumulates in episodic memory and is compiled into rules.

Central to achieving these results are the various memories and processing units in Soar as presented in Figure 1, as well as the task-independent knowledge that controls the use of these knowledge sources. A critical component of Soar's ability to support these methods is its employment of impasses when knowledge is incomplete. Impasses are critical for identifying when action modeling is necessary (a tie among competing actions) and for invoking alternative approaches when rule-based action modeling knowledge is missing. In addition, substates provide the representational structure needed to support retrieving and combining knowledge without disrupting the state of the problem being attempted. These components appear to be missing, or at least difficult to achieve, in other architectures, and it would be informative to attempt to duplicate the qualitative structure achieved here in other cognitive architectures.

Acknowledgment

The authors acknowledge the funding support of the Office of Naval Research under grant number N00014-08-1-0099.

References

Anderson, J. R. (2007). *How Can the Human Mind Occur in the Physical Universe?* Oxford University Press.

Atance, C. M., and O'Neill, D. K. (2005). The emergence of episodic future thinking in humans. *Learning and Motivation* 36(2): 126-144.

Best, B. J. and Lebiere, C. (2003). Teamwork, Communication, and Planning in ACT-R Agents Engaging in Urban Combat in Virtual Environments, *International Joint Conference on Artificial Intelligence*.

de Groot, A. D. (1965). *Thought and choice in chess*. The Hague: Mouton Publishers.

Fikes, R., and Nilsson, N. (1971). STRIPS: A new approach in the application of theorem proving to problem solving. *Artificial Intelligence* 2, 189-208.

Forbus, K. and Gentner, D. (1997). Qualitative mental models: Simulations or memories? *Proceedings of the Eleventh International Workshop on Qualitative Reasoning.* Cortona, Italy.

Johnson, S. H. (2000). Thinking ahead: the case for motor imagery in prospective judgements of prehension. *Cognition*, *74*(1), 33-70.

Laird, J. E. (2008). Extending the Soar Cognitive Architecture. *Proceedings of the First Conference on Artificial General Intelligence*.

Newell, A., and Simon, H. A. (1972*). Human problem solving*. Englewood Cliffs, NJ: Prentice-Hall.

Nuxoll, A. M. and Laird, J. E. (2007). Extending Cognitive Architecture with Episodic Memory. *Proceedings of the 22nd National Conference on Artificial Intelligence.*

Schacter, D. L, and Addis, D. R. (2007). The cognitive neuroscience of constructive memory: remembering the past and imagining the future. *Philosophical Transactions of the Royal Society of London*. Series B, Biological Sciences 362, no. 1481 (May 29): 773-786.

Wintermute, S. (2009). Integrating Reasoning and Action through Simulation. *Proceedings of the Second Conference on Artificial General Intelligence*.

Wintermute, S., and Laird, J. E. (2009). Imagery as Compensation for an Imperfect Abstract Problem Representatio*n. Proceedings of the 31st Annual Conference of the Cognitive Science Society*.

Xu, J. Z. & Laird, J. E. (2010). Instance-Based Online Learning of Deterministic Relational Action Models, *Proceedings of the Twenty-Fifth Conference on Artificial Intelligence.*