Psychologically Inspired Symbolic Cognitive Architectures

Jeremy Ludwig

2/13/2005

A position paper for partial completion of the Oral Comprehensive requirement

Committee Members:
Art Farley (Chairperson)
Anthony Hornof
Dejing Dou

Abstract

This paper provides an overview of the area of psychological inspired symbolic cognitive architectures. It does this by first defining the terms that describe this research area and by selecting three architectures to examine: ACT-R, EPIC, and Soar. For each of these architectures, the motivations, assumptions, and features are evaluated. The similarities and differences between the architectures are discussed as issues are raised and the results summarized. A few example cognitive models are described to demonstrate the different architectures. Finally, some current and future research areas in the field of cognitive architectures are examined as a springboard for future dissertation related work.

Table of Contents

Introduction

Research into the area of cognitive architectures has two main benefits. One of these is the advancement of psychological theories of cognition. This is done by examining the start of the art in psychological theory and realizing it as a computer program, a cognitive architecture, which can be empirically tested. The other main benefit involves using cognitive architectures to create more accurate simulations of human behavior. These models are useful in a wide variety of tasks, from modeling a user of a computer program in studies of human-computer interaction to modeling a military jet pilot in a large distributed simulation.

This paper covers the breadth of modern psychologically inspired symbolic cognitive architectures. The reading list (see appendices for the reading list and annotated bibliography) on which this paper is based is designed to expose a broad range of topics and ideas for three cognitive architectures: ACT-R, EPIC, and Soar[1]. These readings are divided into four categories: foundation, ACT-R, EPIC, and Soar. For each of the areas, the readings support a general understanding of the architecture as well as examine a number of cognitive models. In addition to specific architectures and models, the reading list also covers a broad array of the important issues in cognitive architectures.

Area of Study

Some additional clarification is required with respect to the area of study: psychologically inspired symbolic cognitive architectures. The first clarification is based around what is meant by the terms "cognitive", "psychologically inspired" and "symbolic". The second is why ACT-R, EPIC, and Soar are chosen.

Cognitive

Newell (1990) specifies that cognitive behavior becomes evident at about one second in what he calls the *cognitive band*. This is based on a discussion of system levels where the time difference between levels is about an order of magnitude. Roughly, these levels are biological (up to 10ms), cognitive (100ms – 10 seconds), rational (minutes – hours), and social (days – months). He suggests that symbols are accessed around the 10ms level, which serves as the starting point of symbolic cognition. The level built on top of this, around 100ms, starts cognitive activity with deliberation (accessing remote knowledge and putting it to use). For example, retrieving a piece of knowledge from long term memory, say the name of your dog, and deciding that the answer is Baxter based on this fact would take around 100 ms. This doesn't include saying or writing the answer, just deciding that Baxter is the answer. Newell writes that composed operations, such as pushing a button, begin around the one second level.

Psychologically Inspired

The term psychologically inspired simply means that the architectures were created to model human cognitive activity, where cognitive activity is defined as those operations

[1] The acronyms stand for, respectively, Adaptive Control of Thought – Rational, Executive-Process Interactive Control, and Soar (not an acronym though it used to be SOAR – State Operator And Result)

that take between 10ms and 10s as describe above. An example of modeling that is not psychologically inspired are the finite state machines commonly used in the game industry to model intelligent behavior (Fu & Houlette, 2003), such as the bots in a first person shooter.

This paper takes psychologically inspired one step further, requiring included architectures to be psychologically validated as well. This means that both the theory behind an architecture, and the models created with it, have been validated to some extent. Validation is an important part of this paper and is discussed in greater detail in the *Cognitive Architecture* section of this paper. See Pew and Mavor (1998) for information on comparing the extent of validation for a large number of cognitive architectures.

Symbolic

Newell (1990) provides a description of symbol systems, stating that they contain *memory* (of tokens, the physical representation of a symbol), *symbols* (tokens that represent distal information), *operations* (that manipulate symbols), *interpretations* (of symbols to perform operations), and *capacities* (memory, composability, and interpretability). Symbol systems are often implemented as collections of if-then rules. Newell argues that symbol systems naturally represent human knowledge, making it easier to both encode knowledge and understand behavior of the architecture (see Clark, 2001 for an overview of some of the difficulties of this stance). An example of a subsymbolic system is a neural network. In such a network, behavior is represented as weighted connections between nodes rather than human interpretable symbols.

ACT-R/EPIC/Soar

A large number of symbolic cognitive architectures are described by Pew & Mavor (1998). Of these, only ACT-R, EPIC, and Soar are described as psychologically validated. Cognet was another candidate but is not covered in the reading list based on three reasons: it lacks psychological validation (Pew & Mavor, 1998), it is a proprietary system (whereas ACT-R, EPIC, and Soar are academic), and it has been cited as being too complex[2] (Kieras, 2003). The selection of ACT-R, EPIC, and Soar is supported by Byrne's (2003a) review of cognitive architectures.

Intention

The goal of this paper is to demonstrate knowledge of the basic principles and current directions in cognitive architectures. The reading list for this paper answers a number of questions (detailed answers can be found in the *Detailed Issues* appendix) such as:

- What are the basic issues surrounding symbolic cognitive architectures? How do the selected cognitive architectures differ from one another? How are they similar?
- How do the cognitive models themselves differ across architectures?
- What kinds of things can be done with a cognitive architecture? How does this differ across theories of cognition?

[2] Kieras writes that Cognet "… incorporates a multitude of ideas about human cognition and performance, so many that it appears to be rather difficult to understand how it works." The chosen cognitive architectures all use fewer and less complex mechanisms to account for cognitive behavior.

- In the architectures that support learning, how does it work? Is it useful?
- What are some of the current research topics in cognitive architectures and modeling?

The paper supports the aforementioned goals by providing a comprehensive review of the studied area. A general description of cognitive architectures comprises the next section. This foundation sheds light on some of the similarities of the chosen architectures. Following sections focus on the specific assumptions, motivations, and features of ACT-R, EPIC, and Soar. The final section describes both current and future research areas in cognitive architectures. Appendices supplement the paper, including a reading list, an annotated bibliography, and a list of detailed issues.

Cognitive Architecture Components

There are three major components essential in defining what a cognitive architecture is: theory, architecture, and model (Figure 1). First, a number of different ways of thinking about these components are examined. This is followed by a discussion of each component individually.

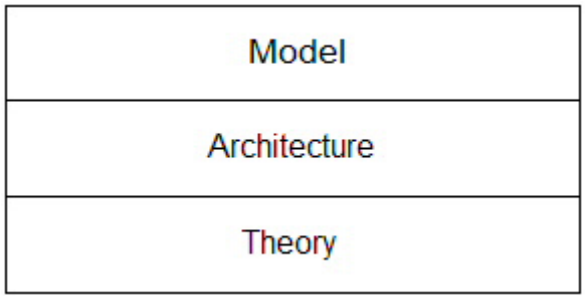


Figure 1. Cognitive Architecture Components

One way of visualizing the components is that the architecture lies between a theory and a model. A theory of cognition forms the core of the architecture. This theory describes how cognition occurs and what constraints there are on cognition. An architecture is built based on this theory. Creating an architecture requires complete specification of the relationships, constraints, and functions described in the theory. Finally, the architecture is used to build a cognitive model. A model specifies, in the language of the architecture, how to complete a given task.

There are other ways to visualize how these three components fit together. Figure 2 shows an explicitness-based visualization. The theory, based on the psychological literature, leaves a number of details unspecified. One example might be how procedural knowledge is stored. The architecture, going beyond the theory, adds constraints such as that procedural knowledge is stored as production rules. Finally, a model provides further constraints, dictating exactly what the production rules are. Progression up the pyramid requires that more details be made explicit.

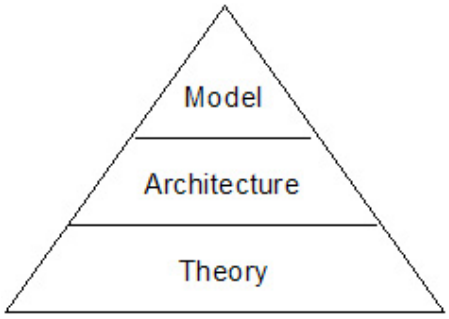


Figure 2. Cognitive Architecture Components (Second Version)

A third way of looking at these components, from a modeler's perspective, is shown in Figure 3. In this image, the architecture extends from the theory, as shown on the left hand side. This signifies that the architecture specifications extend beyond what is currently known in the literature. The right hand side illustrates how the architecture language is used to construct a model and how the model is fed into the interpreter. In this diagram, the model is something that the architecture uses, not an extension of the architecture.

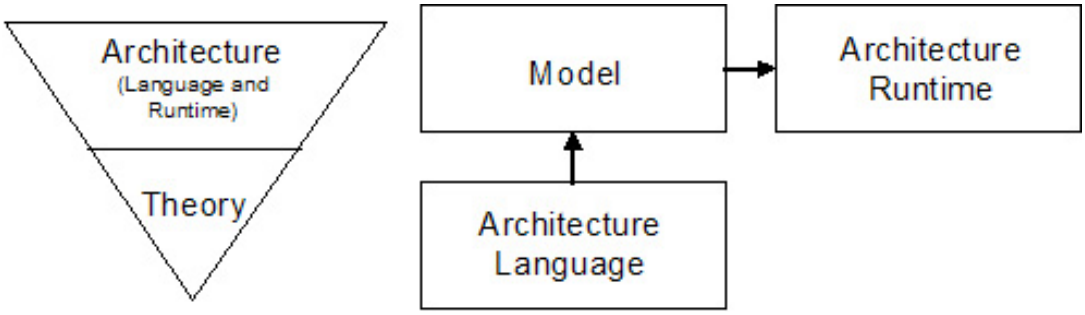


Figure 3. Cognitive Architecture Components (Third Version)

The java programming language can also be thought of along these lines. The java language, i.e. the basic operators and classes of java, is based on computer language theory. The java virtual machine (architecture interpreter) interprets a program (model) built using the java language. From this perspective, a cognitive architecture is essentially a specialty computer programming language.

A fourth way of relating these components, again from the perspective of a modeler, is that all of the pieces combine to form a cognitive model. A cognitive model, using this definition, contains the theory implemented in the architecture as well as the productions and parameter settings written in the modeling language and the simulated device the productions interact with.

Theory

The theory behind a cognitive architecture, the first component in all of these diagrams, describes the various processes involved in cognition, how these processes are related to and communicate with one another, and the types of information that these processes work with. First, the Model Human Processor (Card, Moran, & Newell, 1983) is described as a specific example of a theory of cognition. Second, the concept of a particular type of theory, a unified theory of cognition (Newell, 1990), is discussed. A unified theory of cognition is one that covers a substantial portion of human cognitive behavior.

Model Human Processor

One example of early work in cognitive theories is the Model Human Processor (MHP) (Card, Moran, & Newell, 1983). The overall design of the system can be seen in Figure 4.

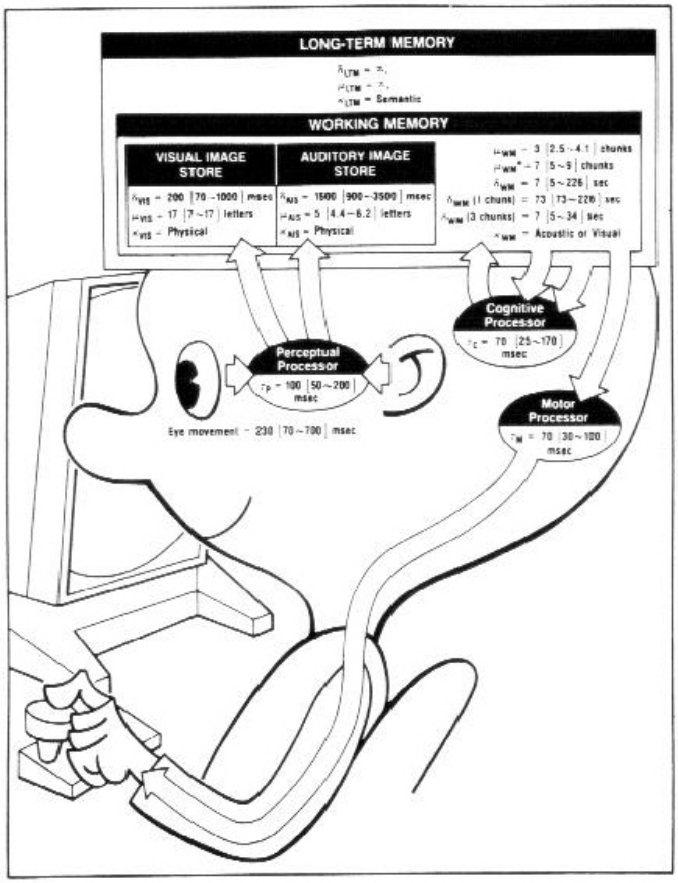


Figure 4. The Model Human Processor - Memories and Processors (Card, Moran, & Newell, 1983)

The MHP is an example of a theory of cognition, designed especially for the study of computer interfaces. It consists of a set of connected processes (perceptual, cognitive, and motor) combined with memories (working and long term) and a set of operational principles that guide how the processors and memories work together. Each processor can perform a number of basic tasks (e.g., the motor processor can tell the finger to press

a button) where the average time to complete a task is gathered from the relevant psychological literature. A few of the ten principles of operation (the rules for running the MHP) are the:

- Recognize-act cycle of the cognitive processor
- Fitt's law
- Power law of practice
- Rationality principle
- Problem space principle.

In addition to being an important part of the MHP, these principles can also be found in ACT-R, EPIC, and Soar. In the recognize-act cycle, the perceptual processors (visual, aural, tactile) send information to the working memory. The updated contents of working memory, possibly with additional information from the long term memory, determine an action to perform. This action may be internal, such as retrieving something from LTM, or external (e.g., sending a command to the motor or verbal processors). A simplification of the MHP recognize-act loop looks something like: Perception -> Recognition -> Decision -> Action.

Fitt's law and the power law of practice are two formulas that describe behaviors in human-computer interaction (HCI). Fitt's law is given as $T_{pos} = I_M \log_2 (D/S + .5)$, where T_{pos} is the time it takes to move a hand some distance, D, to a target of some size, S. I_M is a constant, generally 100 ms/bit. The power law of practice is given as $T_n = T_1 n^{-alpha}$, T_n is the time to perform on the n^{th} trial, T_1 is the time to perform on the first trial, and alpha is a constant generally equal to .4. These two laws are examples of information found in the psychological literature that can be directly applied to constrain behavior in cognitive architectures.

The rationality principle states that humans act in such a way as to achieve their goals in an efficient manner, given their knowledge of the current situation and possible actions. Card, Moran, and Newell (1983) give the following formula: Goals + Task + Operators + Inputs + Knowledge + Processing-limits = Behavior. This principle is a driving force behind the operation of ACT-R, EPIC, and Soar.

Finally, the problem space principle makes assumptions about how to represent rational behavior. The basic tenets are that knowledge can be represented as a set of states, operators change one state into another, operators can be constrained such that they only apply to certain states, and some mechanism must decide which of the applicable operators to apply. While this principle is much more explicit in Soar, it can also be seen in the workings of ACT-R and EPIC[3].

To validate a theory such as the MHP, justification must be given for each of the processors, relationships, and principle of operations defined in the theory. Card, Moran,

[3] Soar has explicit states but EPIC and ACT-R do not. However, in practice EPIC and ACT-R are maneuvering through a state space. For example, in ACT-R the current contents of the working memory buffers (a state, combined with a certain amount of noise) is used to determine the next production to use.

and Newell (1983) devote a significant portion of their work to reviewing psychological studies and describing the impact of the results in the context of their theory.

Unified Theory of Cognition

A unified theory of cognition is a designation for a particular type of cognitive theory, one that captures a substantial portion of human cognitive behavior. In his book, *Unified Theories of Cognition,* Newell (1990) writes about his view of the state of cognitive science and provides an overarching goal for research in the area of cognitive architectures – developing a unified theory of cognition. The book is a coalescing of ideas and previous research, resulting in a focused research direction, a candidate unified theory of cognition, supported by the refined cognitive science platform he describes.

Newell defines a unified theory of cognition as "a single set of mechanisms for all cognitive behavior." The importance of a single set of mechanisms is in developing a single, unified theory rather than stringing together a number of possibly disparate theories that each tackles a small part of cognition. One difference between these two approaches is that a single theory must take into account all of the functionality and constraints found in the human mind, which in turn influences the complete theory as it is developed. Figure 5, taken from his book, describes what he means by cognitive behavior.

Problem solving, decision making, routine action
Memory, learning, skill
Perception, motor behavior
Language
Motivation, emotion
Imagining, dreaming, daydreaming

Figure 5. Areas to be Covered by a Unified Theory of Cognition (Figure and caption from Newell, 1990)

Newell (1990) writes that a candidate unified theory of cognition does not necessarily have to do all of these things, but that it should be making progress down the list of behaviors. This definition also leaves open the likely possibility that there can be more than one unified theory of cognition. Both ACT-R (Anderson & Lebiere, 1998) and Soar (Newell, 1990) claim to be candidates for a unified theory of cognition. The basic theme of a single theory that covers much of cognition is found repeatedly in the cognitive architecture literature.

Architecture

Byrne (2003a) writes that "a cognitive architecture is a broad theory of human cognition based on a wide selection of human experimental data and implemented as a running computer simulation program." Lehman, Laird, and Rosenbloom (1998) provide a slightly different definition: "a cognitive architecture is really two things at once. First, it is a fixed set of mechanisms and structures that process content to produce behavior. At the same time, however, it is a theory, or point of view, about what cognitive behaviors have in common."

Another way of looking at this is: a cognitive architecture is a theory realized as a cognitive model programming language and runtime interpreter. The programming language dictates the representation of knowledge and provides a number of basic functional operators used in model construction. The runtime interpreter executes cognitive models according to the underlying theory, supplying input from the outside world (perception), performing cognitive processing as described by the model and constrained by the architecture (cognition), and producing actions (behavior). The EPIC architecture is used to illustrate the various elements of a cognitive architecture.

Figure 6 lists a few of the operations and knowledge representation capabilities available in the EPIC architecture (Kieras & Meyer, 1998). Operators such as these are the basic building blocks of cognitive models.

- Cognitive
  - Determine whether or not two variables are unique
  - Bind only the first value that matches to a given variable
  - Determine whether or not a variable has some property attached to it
  - Retrieve and item from memory
- Perceptual
  - Determine whether or not an object with some specific properties is currently visible
  - Determine whether or not a sound is currently audible
- Motor
  - Tells the left hand to move to a key and strike with the index finger
  - Tell the eyes to move to some x, y position
- Knowledge Representation
  - Add a clause to memory
  - Delete a clause from the memory

Figure 6. Example Operations Supported by the EPIC Architecture

These cognitive architecture operators provide predictive power to cognitive models in part by constraining what a model can do during its execution (Kieras, 2003). These constraints help provide psychological validity for a given model as the operators are carried out within the runtime portion of the architecture. The goal is to allow people that aren't cognitive psychologists to construct psychologically valid models using the

language of a cognitive architecture. As an example, the execution environment of the EPIC architecture is shown in Figure 7 (Kieras).

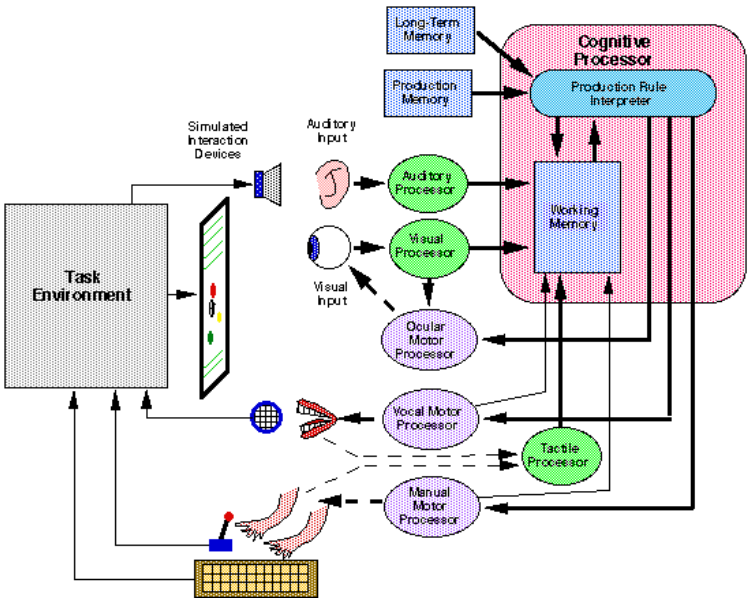


Figure 7. The EPIC Architecture (from Kieras, http://www.eecs.umich.edu/~kieras/epic.html)

Since the validity of models depends in part on the architecture, the validation of architectures is an important issue. The problem is how to determine if an architecture is correctly describing cognition. The evidence for this comes from both theory and models. To some extent, an architecture is based on the specific results from the psychological literature that were used to construct the underlying theory. New or conflicting research findings may provide support for, or detract from, an architecture. However, since architectures must be completely specified they tend to go beyond the research findings. The result is that the main method of validating architectures is empirical, i.e. based on the creation and validation of cognitive models for a diverse array of individual tasks.

Model

Byrne (2003a) defines a cognitive model as a cognitive architecture combined with the knowledge to perform a task. It is important to consider that acquiring and codifying a task requires knowledge-engineering and model-programming skills to 1) specify the task, 2) specify the task environment, and 3) create a set of instructions for a particular architecture that completes the task. Further, a model is a program that performs a task in a particular way while there often is more than one way to perform the task. Finally, the validity of a model has ramifications for both the model itself and the underlying architecture.

To make the idea of cognitive models more concrete, a cognitive model created during an EPIC seminar (Hornof, 2004, personal communication) is examined. The production rules from this model are shown in Figure 8.

```
(Top_point_A
IF
(
        (Step Point AtA)
        (Motor Manual Modality Free)
        (Motor Ocular Modality Free)
        (Visual ?object Text My_Point_A)
)
THEN
(
        (Send_to_motor Manual Perform Ply Cursor ?object Right)
        (Delete (Step Point AtA))
        (Add (Step Click AtA))
))

(Top_click_A
IF
(
        (Step Click AtA)
        (Motor Ocular Modality Free)
        (Motor Manual Modality Free)
)
        THEN
(
        (Add (Step Point AtB))
        (Send_to_motor Manual Perform Punch Z Left Index)
        (Send_to_motor Ocular Perform Move Point_B)
        (Delete (Step Click AtA))
))

(Top_point_B
IF
(
        (Step Point AtB)
        (Motor Manual Modality Free)
        //(Motor Ocular Modality Free)
        (Visual ?object Text My_Point_B)
)
THEN
(
        (Send_to_motor Manual Perform Ply Cursor ?object Right)
        (Delete (Step Point AtB))
        (Add (Step Click AtB))
))

(Top_click_B
IF
(
        (Step Click AtB)
        (Motor Ocular Modality Free)
        (Motor Manual Modality Free)
)
        THEN
(
        (Add (Step Point AtAAgain))
        (Send_to_motor Manual Perform Punch Z Left Index)
        (Send_to_motor Ocular Perform Move Point_A)
        (Delete (Step Click AtB))
))

(Top_point_A_again
IF
(
        (Step Point AtAAgain)
        (Motor Manual Modality Free)
        (Motor Ocular Modality Free)
        (Visual ?object Text My_Point_A)
)
THEN
(
        (Send_to_motor Manual Perform Ply Cursor ?object Right)
        (Delete (Step Point AtAAgain))
        (Add (Step Click AtAAgain))
))

(Top_click_A_again
IF
(
        (Step Click AtAAgain)
        (Motor Ocular Modality Free)
        (Motor Manual Modality Free)
)
        THEN
(
        (Send_to_motor Manual Perform Punch Z Left Index)
        (Delete (Step Click AtAAgain))
        (Add (Step Filler Not_Found))
))
```

Figure 8. Production Rules from a Simple EPIC Model

There are two points in the artifact: A and B. The task is to point to A with the right hand, assuming the user is already looking at A, and press a key (Z) with the left hand when it is reached. Then point from A to B with the right hand and press a key with the left hand. Finally point back to A again, and press a key again. The task completed by this model is basic, mainly demonstrating motor modeling in EPIC.

The model includes operations that check whether or not the manual or visual modalities are free (e.g., Motor Manual Modality Free) and whether or not an object is visible to the architecture based on what the eyes are currently looking at (Visual ?object Text My_Point_B), where ?object binds to any object that meets the description. The reaction time performance of this model is constrained by how long it takes the architecture to move the eyes, move the mouse, press buttons, etc. These production rules also show some of the working memory elements that are added to memory specifically to allow the task to be completed (such as Add(Step Click AtB)).

In this EPIC model, the model includes a simulation of the program interface. In this case, C++ code is used to build a simulated artifact that is a certain size with two letters, "A" and "B", on the screen. The EPIC runtime interpreter simulates eye movements within this specially created artifact. Additional functions of the artifact are performing measurements (such as reaction time) and task setup (e.g., the specific location of A and B on any given trial).

Creating models such as this provides an opportunity to validate the architecture by comparing model results to observed results. This means that the performance of cognitive models, built in a particular architecture, are compared with human performance. Newell (1990) provides a detailed discussion of agreement between human and model data, but for the purposes of this paper two types of agreement are important: quantitative (means and/or typical values) and qualitative (results are in the right directions but the scale does not match[4]). Either of these types of agreements can provide validation of a model and to some extent the underlying architecture as well.

The performance characteristics to be matched depend on the task being modeled, varying from something as specific as reaction time measured in milliseconds to as general as learning curves or something that "looks good" to a human observer. Kieras, Wood, and Meyer (1997) describe a model based on a well known study in human-computer interaction, called Project Ernestine. This model makes use of very small human-computer interaction measurements such as the time it takes to make a single keystroke. The authors build a cognitive model, based on a task analysis, which determines that it will take telephone operators longer to complete a task with a new interface than with the existing interface based on the duration, dependencies, and ordering of short sub-tasks. On the other end of the spectrum are personified interface agents (Yoshikawa, 2003), where the goal of the model is to seem human-like (e.g., social and affective behavior) to the user.

[4] This definition of qualitative results is not the same as that commonly used in psychology.

ACT-R / EPIC / Soar

Many of the differences between ACT-R, EPIC, and Soar stem from their varying motivations and assumptions. First the motivations of these three architectures are discussed and summarized in Table 1. Following this is an examination of the fundamental assumptions that the architectures make and how these assumptions push the architectures in different directions. Detailed summaries on how each of these architectures actually work can be found in Appendix C: List of Detailed Issues.

Motivations

The ACT (Adaptive Control of Thought) series of theories and programs have changed significantly since their original instantiation in 1976. Initially, the architecture has been used to study memory and problem solving with a long term goal of becoming a unified theory of cognition. In their book, *The Atomic Components of Thought,* Anderson & Lebiere (Anderson & Lebiere, 1998) make the case for ACT-R being ready for this title.

Kieras and Meyer (1997) list three major motivations for EPIC: examining embodied cognition, creating computational models of performance and attention as well as cognition, and studying the executive processor with a focus on multiple task performance.

Soar is specifically designed to be a candidate unified theory of cognition, focusing on human problem solving and learning (Lehman et al., 1998). To understand what is meant by problem solving and learning, the authors describe a number of features common to cognitive behaviors:

1. *It is goal oriented*
2. *It reflects a rich, complex, detailed environment*
3. *It requires a large amount of knowledge*
4. *It requires the use of symbols and abstractions*
5. *It is flexible, and a function of the environment*
6. *It requires learning from the environment and experience*

In attempting to model problem solving and learning, Soar needs to model these aspects common to cognitive behaviors.

The motivations are summarized in Table 1. These varying motivations can be seen to have directly influenced the feature set of the various architectures. Some examples of this are the detailed implementation of memory in ACT-R, perceptual-motor processes in EPIC, and problem spaces in Soar.

	ACT-R (Anderson et al., 2004)	**EPIC** (Kieras & Meyer, 1997)	**Soar** (Laird & Congden, 2004)
Motivations	Memory and problem solving	Embodied cognition and multiple task performance	Problem solving and learning

Table 1. Motivations for ACT-R, EPIC, and Soar

Assumptions

The assumptions chosen by the different theories also affect the resulting architectures. Of course, to some degree these three architectures are very similar and share a number of the same assumptions. Some examples of this are the principle of rationality, goal directed behavior, and the use of production rules (though there are a number of differences in the specifics of production rules and how they are processed). However, the architectures are still far apart on some of their assumptions as seen in Table 2.

Anderson & Lebiere (1998) outline the basic assumptions of ACT-R. A number of their assumptions differ from those of EPIC and/or Soar. Among these are the use of "chunks", each of which has up to three or so attributes with values, for representing declarative memory. The attributes of a chunk are the sources of activation for elements in declarative memory, with different attributes having the possibility of having differing strengths of activation. Procedural memory is represented using production rules, with the possibility of partial pattern matching of productions. Partial matching requires a way of selecting which production to fire since ACT-R only fires one production each cycle. This supports an assumption that productions are chosen using utility based conflict resolution. Learning occurs in a variety of places, from production utilities to production compilation to activation strengths. These assumptions are discussed in practice in Anderson et al. (2004), including embodied cognition, the defense of processing/buffer limitations (central bottleneck theory), the representation of memory in "chunks", the subsymbolic aspects of declarative and procedural memory, the learning involved in these subsymbolic aspects, and the learning of productions through production compilation.

Kieras & Meyer (1997) lay out the basic philosophy of EPIC: make the simplest assumptions first and refine later. One of the main assumptions made by the EPIC architecture, based on this philosophy, is that cognitive limitations are the result of perceptual constraints and motor constraints (including their respective working memories). This differs from the assumption that limitations are caused by bottlenecks in the central cognitive processor. The parallel rule-processing of the cognitive processor supports this assumption. It is important to note that performance is still limited by the rate of execution (50ms cycle), perceptual/motor processors (only one set of eyes) and the various working memories (see Meyer & Kieras, 1997, for a more detailed treatment). This basic assumption is propagated into the processors as well. For example, there are no capacity limitations in verbal working memory. Instead, the number of items is limited by the decay rate of items in memory and the rate of subvocalization (i.e. rehearsal).

Newell (1990) lists the main characteristics of central cognition in Soar:

1. "Problem spaces represent all tasks
2. Productions provide all long-term memory (symbols)
    a. Search control, operators, declarative knowledge
3. Attribute/value representation is the medium for all things.
4. Preference-based procedure used for all decisions
5. Goals (and goal stack) direct all behavior
6. Chunking of all goal-results (impasse resolutions) occurs continuously"

One additional principle, applied to all of the above characteristics, is *uniformity*. For example, there is one type of LTM (productions) to be used for episodic, semantic, and procedural knowledge. Another principle is implied by the *decision cycle* (described in more detail in the architecture description). This principle is that all relevant information should be considered before a decision is made.

	ACT-R (Anderson et al., 2004)	**EPIC** (Kieras & Meyer, 1997)	**Soar** (Laird & Congden, 2004)
Major Assumptions	Activation of chunks as declarative knowledge. Central processing limitation.	Embodied cognition. Making the simplest assumptions first and refining later.	Representing cognition in the terms of problem spaces. The principle of uniformity.

Table 2. Assumptions and Approaches of ACT-R, EPIC, and Soar

These different assumptions and approaches lead to significantly different architectures. Perhaps the best example of this is how the architectures make decisions. Each cycle, EPIC applies all production rules that match the current state. ACT-R, on the other hand, computes the utility of all partial matches and (stochastically) determines which single rule will fire in a given cycle. Soar, differing from both of these, tries to apply all relevant information through what they call a decision cycle. Here, Soar fires every rule that matches in continuous cycles until no new rules fire. From all of the operators suggested during these waves, preference operators choose which single action will actually be performed. Continuing this thread, the different feature sets of the three architectures is discussed.

Features

In order to study the different feature sets, each architecture is examined in turn. EPIC is described first because its perceptual/motor processors are the basis for ACT-R and because it is the easiest of the three architectures to understand. Following this, the features of ACT-R and Soar are given. Table 3 summarizes this information in an easy to read table format. For a number of features the table includes additional information not found in the text. This table is similar to, and contains information from, the tables compiled by Pew and Mavor (1998) and Bryne (2003a) as well as the attributions given in the table.

EPIC

EPIC consists of a series of interconnected processors (visual, auditory, cognitive, motor) based on the multiple-resource theory (Kieras & Meyer, 1997, 1998). Each processor contains its own working memory in addition to a partition in the general working memory with processor-specific functionality. A brief overview of the processors follows after a discussion on multiple-resource theory.

The multiple-resource theory is in opposition to the central limitation theory of cognition. The former places limits on cognition by the relationships and dependencies of multiple resources, while the latter places limits on cognition by positing a cognitive bottleneck. Kieras and Meyer (1997) write "… we assume that limitations on human ability are all

structural; that is, performance of tasks may be limited by constraints on peripheral perceptual and motor mechanisms or by limited verbal working memory capacity, rather than by a pervasive limit on cognitive-processing capacity."

The visual processor works as a pipeline. For example, the detection of a new visual object is registered 50ms after it becomes visible, shape information becomes available at about 100ms, and task specific pattern recognition (e.g., an icon) may take 250ms. Visual working memory makes available to the cognitive processors items and their attributes that are currently in view; the visual portion of the general working memory contains these as well as those seen previously that have not yet decayed. The availability of attributes is determined by the region (originally fovea, parafovea, and periphery). Users have significant control over the visual processor due to a large number of customizable parameters. It is important to note that the visual processor does not actually recognize objects. Instead, it receives symbolic information from the device and simulates the recognition process.

The auditory processor is similar to the visual processor with minor differences. While the decay was originally fixed, an extension of EPIC resulted in a more realistic working memory. Other differences include that items in auditory memory have links to the previous/next item in memory to preserve serial order and special tags to note external or internal items.

The cognitive processor is based on user defined production (if/then) rules. This means that there is no general executive cognitive processor; a specialized one must be created for each task. This is more notable in multi-task situations rather than single task situations. Processing happens on a 50ms cycle (or optionally stochastically around 50 ms). These are discrete steps; all processing happens during a regularly scheduled cycle. During each cycle there are no central processor bottlenecks, so all rules that can fire (perceptual motor processors are limited resources) will fire. The working memory for the cognitive processor is unlimited and items put there by the cognitive processor do not decay. The cognitive section of the general working memory contains the goals and state of production rules along with general task information. Finally, there is no specific attention mechanism. All items in the general working memory are available for matching production rules.

Hands, eyes, and voice are controlled by the motor processors. Moving the hands requires specifying the style, hand, finger, direction, and extent of the motion. The specification is used to prepare and execute the movement. Only one movement can be prepared or executed at a time, though a movement can be prepared while another is being executed. Each feature takes 50ms to prepare along with a 50ms initiation delay. The actual movement time is based on Fitt's law, with a minimum of 100ms. The motor processor working memory can store and reuse features for similar movements. The vocalization motor is fairly basic, using mostly fixed times for different utterances. The oculomotor processor controls the eye movements in either a voluntary or involuntary mode.

ACT-R

Much like EPIC, ACT-R is composed of a set serial modules running in parallel: perceptual, motor, goal, declarative memory, and cognitive (procedural memory)(Anderson et al., 2004). Further, just like EPIC, the system cycle time is 50ms. However, there are a number of differences between the two architectures in both the general operating principles and the modules themselves.

Foremost of these differences are the operating constraints on memory items. Only a single production rule can fire on any given cycle (central bottleneck theory) in the cognitive module. The cognitive module communicates with the other systems through independent buffers, each of which holds only "chunk" at a time. Other significant differences in ACT-R are the combination of symbolic and subsymbolic aspects as well as the various methods of learning. These differences will become more concrete as the individual modules are examined.

The main difference in the perceptual/motor system is the model of attention. ACT-R contains two visual buffers: *where* and *what*. *Where* contains the location and basic features of the visual items. Putting chunks in the where buffer does not require an attention shift. The *what* buffer contains identified chunks (e.g. the encoded text from a visual display) and requires a shift in attention.

While EPIC has a goal structure based on convention, ACT-R has a goal structure built into the model. Goals can be altered, added to, or removed from the goal buffer. Additionally, as goals are added or removed they are added to declarative memory as additional chunks.

Declarative memory is composed of items called "chunks". An example chunk is show in Figure 9. This chunk specifies that 8 + 4 = 12.

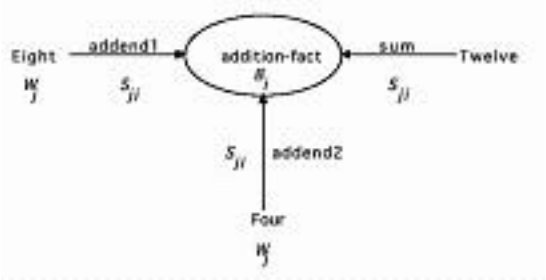


Figure 9. An Example of a Chunk from Anderson et al. (2004)

The availability and recall speed is controlled by activation functions. Chunks are activated based on the slots of the current goal, the weights and strengths of the associations, and the base activation (a separate equation) of the chunk. As a model progresses, the activation values of the chunks will change. This is one form of subsymbolic learning in ACT-R.

During any given cycle, multiple productions may be able to fire. However, because of the central processor bottleneck assumption, only one of the productions can fire. A production is chosen to fire based on an *expected utility* equation. Similar to activation

values, the variables in the production choice equations can also change over time based on successes and/or failures. This results in subymbolic learning for conflict resolution.

ACT-R also features a symbolic method for procedural learning called production compilation. This feature attempts to combine any two productions that occur in sequence into a single production with the effect of both productions. These new productions improve performance in two ways. First, one production is more efficient than two productions. Second, new productions can directly encode previously retrieved declarative memory thus reducing the number of declarative memory retrievals in the future.

Soar

The definitive description of the Soar architecture is provided by Laird & Congden (2004). In their paper they state that the basic component of the Soar architecture is the problem space. To achieve *goals,* Soar uses *operators* to move through a *problem space* represented by *states* that consist of attributes with values and contain a goal and possibly parent and/or child states[5]. The states, with parents and children, form a goal hierarchy. Long term memory (LTM) is made up of productions. It represents general knowledge (such as knowledge about things in general). Working memory (WM) contains the current state (such as knowledge about a particular thing), as well as the state hierarchy.

This architecture is quite different than the previous two architectures. In EPIC and ACT-R, the cognitive processor includes the productions that control behavior in addition to the states of the production rules. Long term memory consists of pieces of information, such as chunks in ACT-R. Soar also takes a different approach to the single/multiple rule firing of ACT-R/EPIC.

The *decision cycle* applies LTM to the current state. There are three stages in the decision cycle: propose operators, select operator, and apply operator. In the propose operator phase, all *elaborations*, operator propositions, and operator comparisons are fired in parallel[6]. This phase continues until no more productions apply (quiescence). Then, from the proposed operators, one is selected.

Elaborations, operator comparisons, and operator selection all merit more description. Elaborations that are proposed by LTM elements will be added as they are true (in "parallel waves"[7]) and retracted when they no longer match. The result of this is a general truth maintenance system as part of the architecture. However, this does introduce the following difficulty. Operators, once fired, make a permanent change to the state. This is called o-support (o is for operator). Elaborations (and operator propositions and comparisons) on the other hand are only making temporary changes to the state that may

[5] This brief description of Soar matches nicely with the definition of a cognitive architecture given by Lehman et al. (1998).

[6] Newell (1990) describes perfect intelligence as bringing all relevant knowledge to bear, which is what happens during the decision cycle.

[7] Firing in "parallel waves" involves serially traversing the rules and firing all that apply. After one traverse, traversing the rules again and firing all that apply. This continues until a state is reached where no more rules can be applied.

be revoked any time the current state changes. This is called i-support (i is for instantiation). When creating models, whether or not something has o- or i-support has a significant effect and requires careful consideration.

Proposed operators can be given preferences to direct the operator selection mechanism by operator comparison rules. These preferences include: acceptable, reject, better, worse, best, worst, indifferent, numeric-indifferent (biased indifference), require, and prohibit.

Finally, if an operator cannot be selected an *impasse* is reached. To resolve this impasse, a new *substate* is created in which Soar attempts to resolve the impasse. This new state is a copy of the current state, but with a goal of resolving the impasse. If resolved, Soar's *chunking* mechanism creates a new LTM production to remember what to do if this impasse arises again. This new production contains the relevant features of the state prior to the impasse with the relevant action (e.g., operator comparison; operator proposal).

	ACT-R (Anderson et al., 2004)	**EPIC** (Kieras & Meyer, 1997)	**Soar** (Laird & Congden, 2004)
Relationship to Neurobiology	ACT-R attempts to match the modules and their buffers to particular functional areas of the human brain.	Epic does not directly tie features to brain areas.	One of the requirements of a unified theory of cognition, and thus of Soar, is that it be neurologically realizable. However, Soar does not directly tie features to brain areas.
Modules	Perceptual, motor, goal, declarative memory, and cognitive.	Perceptual, cognitive, and motor.	Perceptual, working memory, long term memory, and motor.
Working Memory	Working memory is both the productions (procedural memory) and the contents of the module buffers. For productions, and the subsymbolic aspects of productions and declarative memory, there is unlimited capacity and duration. The number of declarative objects is limited by the retrieval buffer size of one chunk.	The cognitive section of the general working memory contains the goals and state of production rules along with general task information. Capacity and duration is unlimited.	Working memory consists of problem states, where states are attribute/value pairs. Capacity is unlimited, but duration of some working memory elements is related to the current state (I-support).
Long Term Memory	Called declarative memory in ACT-R. Composed of items called "chunks" that have some number (usually less than three) of weighted slots and values. Availability and recall speed of chunks is controlled by a number of activation functions. These functions include an activation decay that makes less-often-used items harder to activate.	An unlimited number of productions and propositions can be stored in declarative and working memory	Long term memory consists of production rules. Unlimited capacity.
Goals	Goals are tracked in an architectural goal stack. As goals are completed they are added to declarative memory.	There is a convention to encode a goal structure in the production rules.	The problem states form a goal hierarchy called universal sub-goaling.
Learning	ACT-R has a variety of learning mechanisms. The subsymbolic methods include weight adjustment (for chunk slots) and production strength adjustment (for production utility). Symbolic learning includes learning new productions and learning new chunks.	EPIC does not currently include any learning capability.	When an operator cannot be selected, an impasse is reached. Learning (called chunking) occurs when a resolution is found. This involves adding a new production to LTM.

	ACT-R	EPIC	Soar
Standard Cognitive Cycle	1. Input from the other processors into independent buffers that each hold one chunk. 2. The current goal combined with the contents of the buffers may match a number of productions. 3. The production utility equation determines which production rule fires. 4. Fire the production rule 5. Output to the other processors through the single-item independent buffers The cognitive portion of this cycle generally takes 50ms.	1. Input from the perceptual module into working memory. No covert attention mechanism. 2. The contents of working memory may match a number of productions. 3. All matched productions fire simultaneously. 4. Output to the motor processor. Resource conflicts are a possibility and must be avoided by the modeler. The cognitive portion of this cycle generally takes 50 ms.	1. Input from the environment into the current state 2. Propose operators a. All matching elaborations, operator proposals, and operator comparisons fire in parallel. b. Firing continues until nothing is left to fire. c. These firing have I-support 3. Decide on an operator 4. Apply the operator a. This firing has O-support b. Other, I-supported rules may fire or retract based on the new state 5. Output to the environment The cognitive portion of this cycle takes approximately 60ms in the basic case.
Model Creation (not including constructing the simulated task environment or task instances)	An outline of steps to create an ACT-R model: • Create production rules and declarative memory chunks • Setup task specific parameters. There are a lot more possible parameters in ACT-R than EPIC or Soar. Parameters for declarative memory and production utilities will likely need to be adjusted, especially for partial matching of declarative memory and production conflict resolution.	Some of the steps given by Kieras & Meyer (1997): • Create production rules (the executive process; also called a strategy) to perform the task • Setup task specific parameters for the various processors, especially the visual processor • Select motor movement styles if they are not specified in the task	Constructing a Soar model involves the iterative defining and refining of the following: <u>States representation</u> • Attributes with values • Objects are simply values that contain additional attributes <u>Long term memory productions for</u> • Creating the initial state • Operator proposal • Operator evaluation • Operator application • Elaborating the current state o This includes productions that check for failure/desired states or that provide a resolution to an impasse • Removal of completed operators so that they may be proposed again

Table 3. Features of ACT-R, EPIC, and Soar

Model Examples

There have been a large number of cognitive models created with these three architectures. A quick survey of the ACT-R web page reveals over 500 ACT-R related articles, many of which are models, in their comprehensive publication database. The number of EPIC publications is smaller, with approximately 60 listed on various web pages by EPIC researchers. While no comprehensive archive exists for Soar, it appears that the Soar literature is roughly equal to that of ACT-R based on usage at a number of research institutions throughout the world, internet and article search results, and earlier works such as The Soar Papers (Rosenbloom, Laird, & Newell, 1993).

Four example cognitive models are chosen for discussion as a demonstration of the use of cognitive architectures. The first is a dual-task model created with EPIC. The second is an ACT-R model of childhood verb tense usage and an examination of the model for this task. The final two models are in Soar: a simple model from the Soar tutorial and a complex model of military piloting. Additional issues surrounding cognitive models, and there are quite a few, are discussed in the *Detailed Issues* appendix.

Each of these models illustrates some particular aspect of modeling. The first model is chosen largely because of the importance of dual-task modeling in EPIC and to discuss a single task modeled in different architectures. The second model, verb tense usage, is picked because it demonstrates memory modeling, a cornerstone of ACT-R. The first Soar model is chosen mainly because it is an easy to understand example of how to model in Soar. The second Soar model is included as an often cited modeling success story and a springboard for a number of current research projects.

Dual-Task Model

One example of a simple dual-task model is the Wickens' task (Kieras & Meyer, 1997). The first task, of lower priority, is a tracking task. The participant uses a joystick in their right hand to keep a cursor on a target. The second, higher priority, task is to watch an area of the screen and press one of two buttons with their left hand depending on what is displayed (right or left pointing arrow). The low priority task happens continually; the higher priority task happens occasionally. There are two independent variables (the visual angle between the two; the difficulty of the tracking task) and two dependent variables (reaction time for the choice task; tracking performance for two seconds following a choice). This task emphasizes eye movements and executive control, making it a good task to model in EPIC.

Two different executive models were created. The first is the *simple lockout model*, which puts the low priority task on hold anytime the high priority task needs to be executed. From the flowchart describing their model, the basic executive process is:

- *Start Tracking Task*
- *Keep Eye on Tracking Task Cursor*
    - *Wait for Choice Stimulus*

- *Suspend Tracking Task, Start Choice Task*
    - *Wait for Choice Response Started*
- *Resume Tracking Task*

The second model is an *interleaved model*. In this model the two tasks are performed simultaneously as much as possible, e.g. by making the most efficient use of eye movements. The data predicted from this second model provide a good qualitative fit for the observed human data. The basic executive process given for this model is:

- *Start Tracking Task*
- *Keep Eye on Tracking Task Cursor*
    - *Wait for Choice Stimulus*
- *Suspend Tracking Task, Start Choice Task*
- *Resume Tracking Task*
    - *Wait for Choice Response Selection*
- *Suspend Tracking Task*
- *Permit Choice Task Response*
- *Resume Tracking Task*

One of the differences in this executive controller is that the tracking task is resumed while the cognitive processor is still deciding upon a response to the choice task.

Interestingly, this is the same task used in a paper on the EPIC-Soar hybrid (Chong & Laird, 1997). This hybrid uses Soar as the cognitive processor and EPIC as the perceptual/motor processors. The main thrust of the paper is looking at an executive model that is less "task-dependant" than that used by the EPIC model. They developed two different strategies: sequential (first one task and then the other) and concurrent (running both tasks at once). A problem with concurrently running the tasks is known as "jamming" in EPIC – sending two or commands to a processor that can only do one thing at a time. EPIC uses the executive process to keep this from happening; the Soar model detects jams and then learns to avoid them in the future. In order to produce results similar to that of the EPIC model (and the observed data), a number of specialized rules from the first model were included in the Soar model.

Verb Tense Usage

This model comes from the ACT-R tutorial chapter on production rule learning (*Unit 7: Production Rule Learning*). Here is the description of how the model should work given in the tutorial:

> "The learning process of the English past tense is characterized by the so-called U-shaped learning in the learning of irregular verbs. That is, at a certain age children inflect irregular verbs like "to break" correctly, so they say "broke" if they want to use the past tense. But at a later age, they overgeneralize, and start saying "breaked". At an even later stage they again inflect irregular verbs correctly."

The model contains some production rules that control verb tense formation. Other production rules are learned as the model proceeds. The utilities of both kinds of production rules, which control which production rules are fired, need to be learned from "hearing" verbs from the environment. The initial rule set for this model is found in Figure 10. An explanation follows.

```
(p begin-search-exact
        "begin search"
        =goal>
        isa         past-tense
        status      start
        verb        =word
==>
        +retrieval>
        isa         past-tense
        verb        =word
        status      nil
        =goal>
        status      lookup-exact
)

(p lookup-exact
        "lookup word strategy"
        =goal>
        isa         past-tense
        status      lookup-exact
        verb        =word
        =retrieval>
        isa         past-tense
        verb        =word
        stem        =stem
        suffix      =suffix
==>
        =goal>
        stem        =stem
        suffix      =suffix
        status      done
)

(p lookup-exact-failure
        =goal>
        isa         past-tense
        status      lookup-exact
        =retrieval>
        isa         error
==>
        =goal>
        status      done
)
```

```
(p begin-search-analogy
        =goal>
        isa         past-tense
        status      start
        verb        =word
==>
        +retrieval>
        isa         past-tense
        status      nil
        =goal>
        status      lookup-analogy
)

(p lookup-analogy-one
        =goal>
        isa         past-tense
        status      lookup-analogy
        verb        =actual-word
        =retrieval>
        isa         past-tense
        verb        =word
        stem        =word
        suffix      =suffix
==>
        =goal>
        stem        =actual-word
        suffix      =suffix
        status      done
)

(p lookup-analogy-two
        =goal>
        isa         past-tense
        status      lookup-analogy
        verb        =actual-word
        =retrieval>
        isa         past-tense
        verb        =word
        stem        =stem
        suffix      =suffix
==>
        =goal>
        status      done
)

(p lookup-analogy-failure
        =goal>
        isa         past-tense
        status      lookup-analogy
        =retrieval>
        isa         error
==>
        =goal>
        status      done
)
```

Figure 10. The Production Rules from an ACT-R Model of Verb Tense Lookup

In this model, there are two possible directions from the starting point. The first, shown on the left, is to search for an exact match and place it in the retrieval buffer. The second is to search for an analogical match and place it in the retrieval buffer.

Both sets of rules look quite similar. To specify what the contents of a buffer should be, =*buffer*> is used (e.g., =goal> for the goal buffer). On the left hands side of the rule, before the ==>, is what must be matched in order for the rule to fire. On the right hand side of the rule the contents of the buffers can be changed. To request a retrieval from long term memory of some piece of information, +*buffer*> is used. This can be seen in the begin-search-exact rule which uses +retrieval> to find the matching verb. As another example, +goal> can be used to pop a new goal onto the goal stack (not shown above).

In this model production learning is also enabled. One example of a learned production is:

```
(p Production75
        =goal>
                isa PAST-TENSE
                status Start
                verb =word
==>
        =goal>
                stem =word
                suffix Ed
                status Done
)
```

This production simply adds an “ed” suffix to the given word. It would be correctly applied in the case of end (ended) but wrong for spend (spent).

The results from this model are examined qualitatively for the U-shaped learning curve for irregular verbs seen in human children. Figure 11 shows the results given in the tutorial (benchmark) and the results found in the model built while undertaking this paper. This model did not require parameter adjusting or data fitting (other than designing the strategy to complete the task).

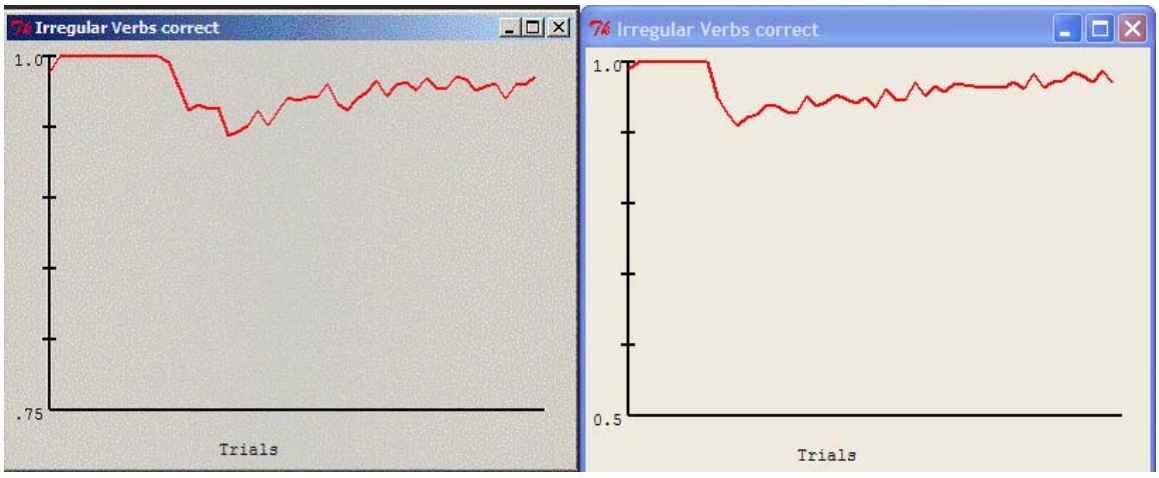


Figure 11. Tutorial (left) and Model (right) Graphs for Irregular Verbs Correct Proportions

Eaters

Eaters is a pac-man like game included in the Soar tutorial. Working through the tutorial chapters, several different models are made for controlling the behavior of an eater. One of the simpler models, shown in Figure 12, is examined (Laird, 2004). This model serves as an easy to understand example of developing a Soar model.

```
sp {propose*move*all-valid
  (state <s> ^io.input-link.my-
location.<dir>.content
{ <content> <> wall })
-->
  (<s> ^operator <o> + =)
  (<o> ^name move
       ^direction <dir>
       ^content <content>)
}

sp {select*move*bonusfood-better-than-
normalfood-empty-eater
  (state <s> ^operator <o1> +
             ^operator <o2> +)
  (<o1> ^name move
       ^content bonusfood)
  (<o2> ^name move
       ^content << normalfood empty eater >>)
-->
  (<s> ^operator <o1> > <o2>)
}

sp {select*move*normalfood-better-than-empty-
eater
  (state <s> ^operator <o1> +
             ^operator <o2> +)
  (<o1> ^name move
       ^content normalfood)
  (<o2> ^name move
       ^content << empty eater >>)
-->
  (<s> ^operator <o1> > <o2>)
}

sp {apply*move
  (state <s> ^operator <o>
             ^io.output-link <out>)
  (<o> ^direction <direction>
       ^name move)
-->
  (<out> ^move.direction <direction>)
}

sp {apply*remove-move
  (state <s> ^operator.name move
             ^io.output-link <out>)
  (<out> ^move <move>)
  (<move> ^status complete)
-->
  (<out> ^move <move> -)
}

sp {monitor*move
  (state <s> ^operator <o>)
  (<o> ^name move
       ^direction <direction>)
-->
  (write |Direction: | <direction>)}
```

Figure 12. Basic Soar model of eater behavior (Laird, 2004)

Soar models are based on the idea of proposing and selecting actions using all available knowledge. All of the matching rules that propose operators are allowed to fire in parallel waves until quiescence is reached. In constructing Soar models, there is not an easy way around this requirement. Propose/select is a built-in part of the Soar modeling language.

The first rule proposes all of the legal moves (i.e., those that do not go into a wall). The first line matches a state <s> where immediately in the direction <dir> there is not a wall. The line (<s> ^operator <o> + =) proposes an operator <o> (hence the +) that is as equally likely as any other operator to be chosen (indicated by the =). The operator itself if then defined as a *move* operator to a specific direction. This rule fires as many times as possible, until no more moves can be added.

While the first rule is firing, the second and third rules are also firing. These two rules both rank some operator as *better than* some other operator. In this case, they are both aimed at maximizing the immediate point value gain. For example, the line (<s> ^operator <o1> > <o2>) states that for state <s> operator <o1> is better than <o2>. These two rules also fire as many times as possible, until there are no more preferences to create. The internal selection process then uses these preferences to select the operator to apply.

After all the possible moves have been proposed, and the operator selected, the last three rules come into play. The first of these actually applies the selected operator by telling the agent to move in the specified direction (<out> ^move .direction <direction>). The last rule fires at the same time, since it matches as well, and sends monitoring information to the screen. The middle of the last three, *apply*remove-move* doesn't fire until the next round, after the action has been completed. Remove-move simply removes the completed move from the output state to make room for the next move.

TacAir-Soar

TacAir-Soar (Jones et al., 1999) is a model of significantly greater complexity than the Eaters model, capable of flying a majority of the militaries' planes on all of the different types of missions. It is an important model for a number of reasons. Foremost of these is the model's size (over 5200 productions, 400 operators, 100 goals) and successes (in types of planes and number of missions modeled as well as evaluated performance of those models). The model is often cited as a large-scale modeling success story.

One of the interesting findings was that roughly 70-90% of the model development time was spent in the communication process – moving information from the subject matter expert to the programmer (Pearson & Laird, 2004). This includes both time spent in pilot model creation and pilot model verification for a number of different mission types. In both cases, the subject matter expert has the knowledge of what the model should look like or should be doing but the developer needs to create/modify the behavior. This problem has spurred on a number of research projects discussed below in *Current and Future Directions*. These include modification and verification of existing large rule sets (Pearson & Laird, 2004) and automatically comparing expert and model behaviors (Wallace & Laird, 2003).

Critiques of Cognitive Architectures

There are four main categories of critiques for cognitive architectures (Clark, 2001; Lewis, 2001): specific models, specific architectures, the general concepts of cognitive architectures and of unified theories of cognition, and the symbolic approach to cognition. Each of these different areas is described below, starting with individual models.

Most models made in a cognitive architecture are subject to scrutiny. Particular areas of examination are the task strategies used by the model (e.g., plausibility), the rules that implement the strategy, the model's use of free parameters (free parameters are adjustments that can be made to cognitive or perceptual parameters on a per model basis), and the goodness of fit between predicted (model) and observed (human) data. With respect to problematic models, Lewis (2001) writes "the theoretical challenge is understanding the extent to which the empirical problems can be resolved within the existing architecture, or whether they point back to problems in the architecture itself."

The underlying theory, operators, and principles of a specific architecture may also be critiqued, for example with respect to ease of use or number of phenomena explained. Another example is the long running debate between EPIC and ACT-R on the existence and nature of cognitive processor limitations (Meyer & Kieras, 1997). A more detailed look at critiques of a particular cognitive architecture can be found in Newell's (1992) response to criticisms of his book *Unified Theories of Cognition*. In this article he responds to a broad array of criticisms, broken into a number of different groups. As an example, the first group is (from Newell, 1992):

- Fundamental failings of the SOAR enterprise
    - The perceptual-motor system
    - The treatment of psychological data and experiments
    - Language
    - The undetermined character of psychological theory in Soar
    - The AI bias of SOAR

Other groups include "missing or wrong critical concepts", "challenges on concepts and capabilities", "the structure of Soar", and "relation to other viewpoints". In other words, nearly every aspect of a specific architecture is open to criticism.

At a slightly higher level, the idea of developing architectures and unified theories of cognition must also be defended. Cooper and Shallice (1995, p. 121) present a number of significant problems with the methodology behind unified theories of cognition. For instance, they find fault with the wide divide between the underlying theory and the implemented details, stating that "in our view, the implementations cannot seriously be taken as statements of psychological theory." While the authors are writing about Soar, the argument is just as valid for other cognitive architectures.

Lewis (2001) summarizes the theoretical issues brought up by various researchers, including Cooper and Shallice, that are still methodological problems today. The three issues discussed by Lewis are: (1) irrelevant specification (also known as the Reitman

specification problem), (2) too many degrees of freedom, and (3) identifiability. The first issue is based around separating the theoretical aspects of the computer implementation from the implementation details, at both the architectural and model levels. The second is that in a complex computer program, such as a cognitive architecture, the number of variables that can be "tweaked" specifically for each model allow the architecture to match nearly any data. The third issue, identifiability, revolves around production systems being universal computational systems. If Soar (or some other production system) is capable of universal computation, then there is no reason to believe that its results will replicate human behavior better than any other universal computation system. More to the point, Lewis defines this as "any sufficiently general proposal for processing schemes or representations can mimic the input/output characteristics of any other general processing or representation scheme."

Cooper and Shallice (1995) also attack the methodological grounds for pursuing universal theories of cognition (UTCs). Specifically, they dismiss most of the advantages that Newell assigns to UTCs and point out the advantages of modern micro theories. The main advantage they do give to UTCs is that, in some cases, different processors in the brain are dependent on one another. While UTCs can capture this relationship, so can sufficiently large micro-theories.

Finally, the whole philosophy of the symbolic approach to cognition is open to questioning. First, the symbolic stance is quickly examined. Following this is a brief discussion of the symbol system problem and some alternatives. This philosophical problem is large and thorny and generally beyond the scope of this paper. See Clark (2001) for a more detailed introduction to this area than the following summary.

The first step in this argument is defining what symbol systems. Newell (1990) defines symbols as "… symbols stand for something and that the token of a symbol occurring in some place in a structure carries the interpretation that the symbols stands for something within the context that is specified because of the symbol token's location." He argues that symbol systems are necessary and sufficient for intelligence: "… (3) cognitive behavior requires symbol systems." This argument is based on the various properties of mental representations that symbol systems possess: (1) productivity (the ability to form a large number of thoughts from a few basic concepts; John, Love, Mary, likes, tomatoes), (2) systematicity (the ability to use similar forms in relationships between symbols; John loves Mary, John likes tomatoes) , (3) compositionality (the ability to use different symbols in similar forms to acquire similar meaning; John Loves Mary and Mary Loves John) and (4) distal access (accessing related information stored somewhere else).

A quote from Clark (2001) gets to the root of one of the main symbol system problems: "it is a commitment to the existence of a computational symbol-manipulating regime *at the level of description most appropriate to understanding the device as a cognitive (reasoning, thinking) engine.*" He points out that symbol systems are especially appealing as a method of describing cognition because they are very similar to the common sense psychology that people use in everyday life. However, the similarity of symbol systems

to how we describe thoughts does not necessarily mean that we actually think using symbols.

Three philosophical arguments summarized by Clark against the symbol system hypothesis are that symbol systems can never truly “understand” the way people do or that the symbols never have any real “meaning”, that humans perform their everyday activities by employing pattern matching not symbol processing, and that the brain is more akin to a “swiss-army knife” than a single set of functions as proposed by UTCs. A number of alternative methods of computation are offered (connectionism, subsumption, and dynamic systems) though none of them make significant headway at getting at issues like the qualitative experience humans enjoy. For now at least, symbolic systems such as cognitive architectures are the reigning method for describing cognitive behavior.

Current and Future Directions

The future directions section covers two distinct areas. The first area consists of some general observations on current ACT-R, EPIC, and Soar research. The second area is a list of possible dissertation topics. The items on this list serve as a number of starting points in the dissertation proposal process.

General Observations

As discussed previously, there are a number of focus areas for research in cognitive architectures. One aim is to create a computer program that implements the particular psychological assumptions in a given candidate unified theory of cognition – a cognitive architecture. The general goal here is the validation and refinement of a particular theory of cognition through model development and analysis.

A second goal is to use cognitive models to study human-computer interaction. As the cognitive theories are shored up with respect to cognitive level human-computer interaction, cognitive models become useful as *a priori* engineering tools.

Human-level behavior is another aim of cognitive modeling. This area of research generally seeks to model human-like behavior at longer durations than human-computer interaction, moving up Newell's time scale (as discussed earlier).

Related to human-level cognitive modeling is the goal of improving, and simplifying, the practice of creating cognitive models. This is desired for both efficiency (it is a very difficult and time consuming task requiring considerable expertise) and maintenance/validation reasons (which also tend to be difficult and time consuming). Some general observations for each of these aims are discussed below.

Architectural Convergence

One striking observation of work in these three cognitive architectures is the degree to which they are becoming more similar. This is not to say they are the same: EPIC still maintains parallelism (Kieras & Meyer, 1997), ACT-R a cognitive bottleneck (Anderson et al., 2004), and Soar a decision cycle (Laird & Congden, 2004).

The foremost example of this convergence is that of the embodiment of cognition as seen in EPIC, where much attention is paid to the fact that cognition happens in the context of perceptual (in) and motor (out) processors. For the class of tasks where these processors, not cognition, are the limiting or significant factor, the operating details of these processors are of primary importance. The embodiment of ACT-R is essentially a re-write of EPIC's perceptual/motor processors[8] (Anderson et al., 2004). Soar has also made use of EPIC's perceptual and motor peripherals in at least one hybrid model (Chong & Laird, 1997). The essential idea is that the various research communities recognized that

[8] Though the authors are quick to point out that "While human cognition is certainly embodied, its embodiment is not what gives human cognition its advantage over that of other species."

(a) this was an important concept and (b) the developed model was a reasonably good model.

This basic phenomenon of developing, implementing, and sharing concepts can be seen in the case of production learning (based on chunking in Soar) in the most recent version of ACT-R (Anderson et al., 2004). Soar is also emulating some of the more powerful concepts in ACT-R such as memory activation (Nuxoll, Laird, & James, 2004), reinforcement learning similar to production utilities (Nason & Laird, 2004), and partial matching and production compilation for episodic memory (Nuxoll & Laird, 2004) .

Human-Computer Interaction

While this was not a focus area of the reading list, a number of readings addressed some of the current human-computer interaction (HCI) research directions (Byrne, 2003a; Kieras, 2003). Significant work has been completed or is underway on architecture and model validation, as well as parameter estimation, for short duration HCI tasks. Other avenues of research include automatic mapping from task descriptions to cognitive models (Byrne, 2001) and the ability to interact with the actual artifacts that human participants use (Shah, Rajyaguru, St. Amant, & Ritter, 2003). Taken together, this research can be seen as working toward the goal of an automated virtual user such as research performed by Hornof (2004, personal communication). This type of *a priori* model would, for example, be very useful in engineering systems involving HCI.

Human-Level Behavior Modeling

The reading list covers human-level behavior modeling examples from both the ACT-R and Soar research communities. Many of these models are designed to "push" the architecture to determine the bounds of what it can handle, i.e. what level of performance and realism it can support. For ACT-R, the best example of this in the reading list is the combination of driving and a cell-phone dialing models (Salvucci, 2001). There are a number of examples of complex models, as well as there creation, maintenance, and validation, in the Soar reading list (Jones et al., 1999; Pearson & Laird, 2004; Wallace & Laird, 2003; Wray & Laird, 2003). While progress is being made, there is still much work to be done in these areas (creation, maintenance, and validation) for large, complex models of human behavior.

Teamwork and social behavior modeling, a requirement for simulating groups of humans, is also an open area of research in human behavior modeling. One paper in the reading list describes a solution using a director which helps choreograph the actions of the individual agents (Magerko, Laird, Assanie, Kerfoot, & Stokes, 2004).

Another research area in human behavior modeling is modeling human emotion (affect), currently a topic of significant interest in artificial intelligence. A number of different papers are examined that provide starting points for investigating the effects of emotion in cognitive architectures (Belavkin, 2001; Marinier & Laird, 2004).

Human error modeling and user variability are also currently of interest. Yoshikawa (2003) reviews a number of psychology-based human error models in developing a human error modeling system for HCI. Byrne (2003b) describes the requirements of a system in order for it to "correctly" produce procedural errors, with ACT-R given as one system that meets these requirements. Wray and Laird (2003) also develop a similar set of requirements in the context of Soar. They point out the need to have error models that produce the desired results at both the population and individual level. Other issues related to modeling human variability, including balancing variability and validation (Pearson & Laird, 2004) and balancing variability and autonomy (Magerko et al., 2004), are discussed later in this paper.

Tools for Model Development

Another general area of research in cognitive architectures focuses on improving tools and methods for cognitive modeling. A wide variety of work is encompassed by this topic. Some examples found in the reading list include: integrated development environments for Soar and ACT-R (Anderson et al., 2004; Laird & Congden, 2004), a tool for automatic construction of GOMS models through interaction with a standard web based artifact (John, Prevas, Salvucci, & Koedinger, 2004), a scenario based knowledge acquisition tool (Pearson & Laird, 2004), and a tool for allowing computer programs to interact with other programs in the exact same way as a human (Shah et al., 2003).

Future Work

What follows is a number of specific areas of interest found during the reading process, roughly ordered form "best" to "worst" in terms of dissertation potential and interest:

- Creating multi-level models that merge high-level AI with lower-level cognitive models to make it easier to simulate human behavior at multiple levels of fidelity.
- Examining the role of creativity in cognitive architecture and models.
- Determining the impact of brain imaging on the theories behind these cognitive architectures.
- Developing and applying human error models. Initial work could focus in more constrained domains. For example, Newell (1990) provides detailed information on typing error rates that could be used to create a realistic model of typing errors.
- Describing a framework for, and models of, augmented cognition. This would especially apply to problem solving tasks and the effects of tools that assist cognition.
- Combining a cognitive model with a large common sense database (e.g. cyc, wordnet) to push the limits of the memory model. There may be room here for comparison of model results to results observed in humans.
- Studying social and team-based action and communication within the context of human-level cognitive models.

The first three items on the list, multi-level models, creativity, and the impact of brain imaging on cognitive architectures are examined in greater detail below.

Multi-Level Models

Cognitive architectures have been widely recognized as useful tools in constructing realistic models of human behavior, especially in military simulations (Pew & Mavor, 1998). One often cited success story of cognitive modeling is TacAir-Soar (Jones et al., 1999). It is an important model for a number of reasons. Foremost among these are the model's size (over 5200 productions, 400 operators, 100 goals) and successes (types of planes, number of missions, evaluated performance). Another example of human behavior modeling is an ACT-R/IMPRINT hybrid pilot model (Craig et al., 2002). This proof of concept combined a task network that described pilot behavior with ACT-R productions that performed various cognitive capabilities.

Despite these and similar successes, there are a number of problems in modeling human behavior using cognitive architectures. These problems include: (1) time spent in model development, maintenance, and verification, (2) modeling variability and cultural effects at the individual agent level (3) orchestrating teamwork for simulated scenarios, (4) adding agent dialog and conversation based interactions, and (5) working with the small grain size of cognitive architectures (around 50ms for ACT-R, EPIC, and Soar).

One of the interesting findings of TacAir-Soar was that roughly 70-90% of the model development time was spent in the communication process – moving information from the subject matter expert to the programmer (Pearson & Laird, 2004). This includes both time spent in pilot model creation and pilot model verification for a number of different mission types. In both cases, the subject matter expert has the knowledge of what the model should look like or should be doing but the developer needs to create/modify the behavior. This problem has spurred on a number of research projects, including modification and verification of existing large rule sets (Pearson & Laird, 2004) and automatically comparing expert and model behaviors (Wallace & Laird, 2003).

Modeling individual variability is also a current problem in cognitive architectures. This variability can be used to account for an array of different behaviors in particular entities such as cultural effects and error modeling. Byrne (2003b) describes the requirements of a system in order for it to "correctly" produce procedural errors, with ACT-R given as one system that meets these requirements. Wray and Laird (2003) also develop a similar set of requirements in the context of Soar. They point out the need to have error models that produce the desired results at both the population and individual level. The same is true of modeling cultural effects, where models must portray the elements of culture at both the general population level and in a particular individual.

The last three issues (3,4, and 5 above) are somewhat related. The problem is the grain size of operations in a model built with a cognitive architecture, around 50ms, can make it difficult to model behavior in the rational and social bands (as defined by Newell, 1990) where interactive behavior must be maintained for minutes or hours. This means that some desired behaviors, such as teamwork and explicit dialogs that humans can participate in, can be difficult to represent using a cognitive architecture by itself. For

example, in order to present a specific scenario to the user Magerko et al. (2004) use a partial order planner to provide direction to a number of autonomous Soar agents.

Various forms of finite state machines (FSMs), an AI-based modeling alternative to cognitive architectures, are often used for developing intelligent agents in the game industry (Fu & Houlette, 2003). Some advantages that FSMs model possess over cognitive models are that they are generally easier to write, maintain, and verify. They also often include useful architectural embellishments such dialog and motion planning systems. Additionally, since behaviors in FSMs are at a much higher level of granularity, modeling higher level behavior such as teamwork is much easier. The main disadvantage is that they are, for the most part, unconstrained. The author can model anything, and the resulting model can be expected to be quite different than the way a human would do it.

Multi-level models hope to leverage the best of both worlds by allowing the user to model behavior at multiple levels of fidelity within a single infrastructure. The end result should be models that are easier to create and maintain than standard cognitive models, but with better approximation to human behavior than standard AI techniques. Feasibility of work in this area was demonstrated by a hybrid task network and ACT-R model of fighter piloting (Craig et al., 2002).

A research program in this area would involve a number of steps. The first is finding a suitable simulation/game with steep demands for higher order behaviors and a need for accurately modeling human behavior. This simulation/game should also come with the user group that would perform the high level modeling. The next step is to examine methods for integrating high level control behavior with existing cognitive architectures. The goal would be to develop a unified framework that meets the following requirements: a single infrastructure that allows modeling at a variety of levels (from FSMs to detailed cognitive models) and that includes all of the standard modeling ideas (e.g., attention, perceptual processing). The final portion of this research would be dedicated to exploring the tradeoffs between ease of modeling/maintenance and similarity to human performance.

Creativity in Cognitive Models

Newell (1990) provides a list of cognitive behaviors required by a unified theory of cognition as shown in Figure 5. Of this list, he writes “we cannot face the entire list all at once. So let us consider it to be a priority list, and work our way down from the top. What I will mean by a unified theory of cognition is a cognitive theory that gets significantly further down the list cumulatively than we have ever done before” (p. 16). One item conspicuously missing from this list is *creativity*. For some researchers, such as Newell, creativity is lumped together with problem solving (Vicinanza & Prietula, 1993). However, such a significant force in human intelligence deserves a closer look.

Boden (1999) provides several useful categorizations of creativity. The first is that creative ideas are both *novel* and *valuable*. In her descriptions, these are the necessary output of a creative process. Of course, there are difficulties in determining what *novel* and *valuable* are. The second categorization Boden makes is between historical (H-) and

personal (P-) creativity. P-creativity is developing an idea that is *novel* and *valuable* to an individual, irrespective of what anyone else thinks. H-creative ideas are those that are *novel* and *valuable* with respect to some community. Boden also makes a distinction between different types of creativity: combinatorial, exploratory, and transformational. Here combinatorial corresponds to brute force, and exploratory corresponds to heuristic, search in some domain. Transformational creativity is the most interesting category, where the basic premise is that the conceptual space of the search will be transformed. This allows creative ideas to be developed that were impossible in the original conceptual space.

While there have been a large number of models of creativity in both the computer science and psychological literature, there has been relatively little work studying creativity in the context of cognitive architectures. A quick review of the literature found only one previous model of creativity (Vicinanza & Prietula, 1993). In their work, creativity is defined as heuristic search through multiple problem spaces. In addition to this work on creativity, a number of other cognitive models may be relevant to research on creativity, such as models of analogy and scientific discovery in ACT-R (e.g., Salvucci & Anderson, 1998; Schunn & Anderson, 1998).

Similar to research cited in this paper on emotion and variability, the first step in studying creativity is to develop a framework for supporting creativity. This involves finding or developing a sufficient model of creativity and describing how this model would be supported in a cognitive architecture. The research by Gabora (2002) is one example of a psychological model of creativity that might form a basis for creativity in architectures.

Once the framework is developed, the architecture can then be extended to implement this framework. The focus of the research would then change to identifying a task/model where creativity is likely to have an effect and testing the model to demonstrate that *creativity*, as it is supported by the architecture, is occurring and affecting the model significantly.

For example, one might loosen the associative memory parameters in ACT-R during a "creative" search phase, gradually returning them to normal at some later time in a simulated annealing inspired algorithm. However, these types of architectural changes may not be enough to support transformational creativity. One of the additional questions in this line of research is determining the types of creativity that can be supported by an architecture and the types that need to be encoded in the model.

There is also an important social aspect of creativity. This is what Boden (1999) is getting at with her discussion of P- and H- creativity. This would be a line of research separate from that of the creativity of an individual described above.

The Impact of Brain Imaging on Cognitive Architectures

Brain imaging technologies can provide valuable guidance in the design of cognitive architectures and their underlying theories and cognitive architectures can help explain the imaging. First, a couple of brain imaging techniques will be discussed. Then some

existing work by the ACT-R research group on brain imaging is summarized. Finally, a brief outline is given for undertaking research in this area.

Functional Magnetic Resonance Imaging (fMRI) is a technique for mapping human brain function in two or three dimensions. It offers "the ability to observe both the structures and also which structures participate in specific functions …, and provides high resolution, noninvasive reports of neural activity detected by a blood oxygen level dependent signal …" (*The Future Role of functional MRI in Medical Applications*). This web site describes one of the active areas of research with fMRI as identifying the particular brain structures that take a role in cognition (e.g., visual perception, language generation, complex problem solving). More information on fMRI can be found at: http://www.functionalmri.org/.

Electroencephalography (EEG) is based on electrical activity in the brain rather than blood flow. Don Tucker, a researcher in the University of Oregon department of psychology, has been involved in the development of an EEG helmet that takes readings from the entire scalp simultaneously (*http://www.egi.com*). While EEG is much faster than fMRI, it has less spatial resolution than fMRI (Marthias, 1996). For this reason, the two technologies are sometimes used together to provide the best possible information.

The various components of ACT-R have been developed with the structure of the brain in mind, with attempts to tie together architecture and brain functionality as shown in Figure 13. The authors describe these relationships as "neural anchors."

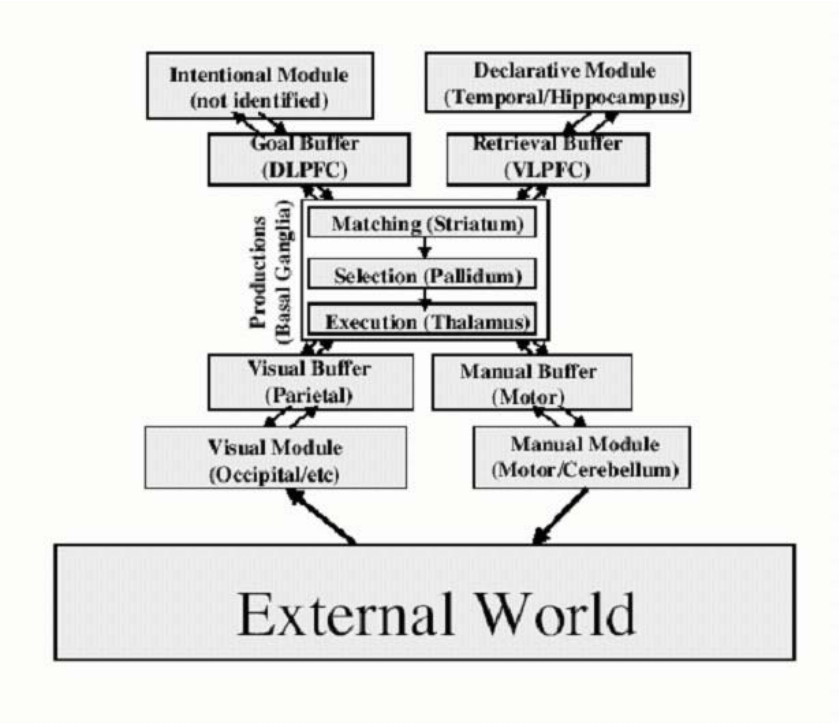


Figure 13. ACT-R to Neurobiology Relationships (from Anderson, et al., 2004)

The research group behind the ACT-R cognitive architecture is currently seeking guidance for further development of their theory from fMRI data (Anderson et al., 2004). They summarize data from fMRI trials that provides evidence for two different hypotheses. First is that there is actually more than one goal module (i.e., more than one area of the brain which keeps track of goal states). The second is that while the Basal Ganglia appears to be involved in cognition (production rules) not all processing actually goes through the Basal Ganglia (central cognitive processor).

While the authors describe some of the difficulty in interpreting fMRI data, it is interesting to think about what a cognitive architecture would look like if it had distributed goal and production modules rather than single centralized modules. Other important questions are which modules communicate with one another and how much.

A research program in this area would involve answering a number of questions to proceed. First, are there tasks where it is expected that a distributed cognitive processor would perform "better" than a single cognitive processor either in human modeling or explanation power? Tasks may need to be designed that would bring out this distinction. Second, is there enough information existing in the literature to support hypotheses on the number, structure, and communication of the sub-cognitive models? If not, then a collaborator may need to be identified in order to pursue this topic further.

Once these questions are answered, the creation of a distributed cognitive component becomes an empirical problem. A distributed cognitive component must be created and situated within one of the existing cognitive architectures. After this is complete, models need to be created within this distributed environment (a research area unto itself). Finally, the results of a model with the distributed cognitive component can be compared to standard cognitive models and the observed data for the selected task(s) to demonstrate the usefulness of distributed cognition.

Conclusion

This paper provides an overview of psychological inspired symbolic cognitive architectures. The goal of this paper is to provide a general introduction to the field. To support this goal, the first section of this paper is dedicated to defining the research area and selecting three architectures to examine: ACT-R, EPIC, and Soar.

Following this is a discussion on what is a cognitive architecture. Unlike many introductory texts, much time is spent describing different ways of viewing the relationships between the three major components of an architecture: theory, architecture, and model. Further, each of these components is defined separately and includes a real-world example.

With a solid foundation of what a cognitive architecture is, the next section dives into specific descriptions of ACT-R, EPIC, and Soar. For each of these architectures, the motivations, assumptions, and features are evaluated. The similarities and differences between the architectures are discussed as issues are raised and the results summarized. A helpful table presents this information in an easy to read format.

A few example cognitive models are described to demonstrate the different feature sets of the architectures. This includes examining an example model, as well as its code, written in each of the covered architectures. The three different example models are a point and press task (EPIC), a verb tense memory model (ACT-R), and a model for a pac-man like character (Soar).

Some critiques of cognitive architectures are also discussed. These critiques fall into four main categories: specific models, specific architectures, the concept of a unified theory of cognition, and the symbolic approach to cognition. Each of these areas is covered.

Finally, some current and future research areas in the field of cognitive architectures are examined as a springboard for future work. Three different future work areas are examined in more detail: modeling at multiple levels of fidelity, the role of creativity in cognitive architectures, and the impact of brain imaging on cognitive architectures.

References

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of mind. *Psychological Review (To Be Published)*.

Anderson, J. R., & Lebiere, C. (1998). *The Atomic Components of Thought*. Mahway, NJ: Erlbaum.

Belavkin, R. V. (2001). The role of emotion in problem solving. In *Proceedings of the AISB'01 Symposium on Emotion, Cognition, and Afective Computing* (pp. 49-57). Heslington, York, England.

Boden, M. (1999). Computer models of creativity. In R. J. Sternberg (Ed.), *Handbook of Creativity* (pp. 351-372). Cambridge: Cambridge University Press.

Byrne, M. D. (2001). ACT-R/PM and menu selection: applying a cognitive architecture to HCI. *International Journal of Human-Computer Studies, 55*, 41-84.

Byrne, M. D. (2003a). Cognitive architecture. In J. A. Jacko & A. Sears (Eds.), *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications* (pp. 97-117). Mahway, NJ: Lawrence Erlbaum Associates.

Byrne, M. D. (2003b). A mechanism-based framework for predicting routine procedural errors. In R. Alterman & D. Kirsh (Eds.), *Proceeding of the Twenty-Fifth Annual Conference of the Cognitive Science Society*. Austin, TX: Cognitive Science Society.

Card, S. K., Moran, T. P., & Newell, A. (1983). *The psychology of human-computer interaction*. Hillsdale, N.J.: L. Erlbaum Associates.

Chong, R. S., & Laird, J. E. (1997). Identifying dual-task executive process knowledge using EPIC-Soar. In M. Shafto & P. Langley (Eds.), *Proceedings of the Nineteenth Annual Conference of the Cognitive Science Society* (pp. 107-112). Hillsdale, NJ: Erlbaum.

Clark, A. (2001). *Mindware: An Introduction to the Philosophy of Cognitive Science*. New York: Oxford University Press.

Cooper, R., & Shallice, T. (1995). Soar and the case for unified thoeries of cognition. *Cognition, 55*, 115-149.

Craig, K., Doyal, J., Brett, B., Lebiere, C., Biefeld, E., & Martin, E. A. (2002). *Development of a hybrid model of tractical figher pilot behavior using IMPRINT task network modeling and the adatpive control of thought - rational (ACT-R).* Paper presented at the 11th Conference on Computer Generated Forces and Behavior Representation.

Fu, D., & Houlette, R. (2003). The ultimate guide to FSMs in games. In S. Rabin (Ed.), *AI Game Programming Wisdom 2*.

The Future Role of functional MRI in Medical Applications. Retrieved October 7, 2004, from http://www.fmri.org/fmri.htm

http://www.egi.com. Retrieved October 11, 2004, from http://www.egi.com

John, B. E., Prevas, K., Salvucci, D. D., & Koedinger, K. (2004). Predictive human performance modeling made easy. In *Human Factors in Computing Systems: CHI 2004 Conference Proceedings*. New York: ACM Press.

Jones, R. M., Laird, J. E., Nielsen, P. E., Coulter, K. J., Kenny, P., & Koss, F. V. (1999). Automated intelligent pilots for combat flight simulation. *AI Magazine, 20*(1), 27-41.

Kieras, D. E.*EPIC: A cognitive architecture for computational modeling of human performance*. Retrieved December 1, 2004, from http://www.eecs.umich.edu/~kieras/epic.html

Kieras, D. E. (2003). Model-based evaluation. In J. A. Jacko & A. Sears (Eds.), *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications* (pp. 1139-1151). Mahway, NJ: Lawrence Erlbaum Associates.

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

Kieras, D. E., & Meyer, D. E. (1998). *The EPIC Architecture: Principles of Operation*. Retrieved 2-22, 2004, from www.eecs.umich.edu/people/kieras/EPIC/EPICArch.pdf

Kieras, D. E., Wood, S. D., & Meyer, D. E. (1997). Predictive engineering models based on the EPIC architecture for a multimodal high-performance human-computer interaction task. *ACM Transactions on Computer-Human Interaction, 4*(3), 230-275.

Laird, J. E. (2004). *The Soar 8 Tutorial*. Retrieved August 16, 2004, from http://sitemaker.umich.edu/soar/soar_software_downloads

Laird, J. E., & Congden, C. B. (2004). *The Soar User's Manual Version 8.5 Edition 1*. Retrieved August 16, 2004, from http://sitemaker.umich.edu/soar/soar_software_downloads

Lehman, J. F., Laird, J. E., & Rosenbloom, P. S. (1998). A gentle introduction to SOAR: An architecture for human cognition. In D. Scarborough & S. Sternberg (Eds.), (2 ed., Vol. 4, pp. 212-249). Cambridge, MA: MIT Press.

Lewis, R. L. (2001). Cognitive theory, Soar. In *International Encyclopedia of the Social and Behavioral Sciences*. Amsterdam: Pergamon (Elsevier Science).

Magerko, B., Laird, J. E., Assanie, M., Kerfoot, A., & Stokes, D. (2004). *AI characters and directors for interactive computer games.* Paper presented at the Innovative Applications of Artificial Intelligence Conference, San Jose, CA.

Marinier, R., & Laird, J. E. (2004). *Toward a comprehensive computational model of emotions and feelings.* Paper presented at the International Conference on Cognitive Modeling.

Marthias, R. (1996). *The Basics of Brain Imaging*. Retrieved October 11, 2004, from http://www.nida.nih.gov/NIDA_Notes/NNVol11N5/Basics.html

Meyer, D. E., & Kieras, D. E. (1997). A computation theory of executive control process and human multiple-task performance: Part 1. Basic Mechanisms. *Psychological Review, 104*, 3-65.

Nason, S., & Laird, J. E. (2004). *Soar-RL: Integrating reinforcement learning with Soar.* Paper presented at the International Conference on Cognitive Modeling, 2004.

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

Newell, A. (1992). SOAR as a unified theory of cogntion: Issues and explanations. *Behavioral and Brain Sciences, 15*(3), 464-492.

Nuxoll, A., & Laird, J. E. (2004). *A cognitive model of episodic memory integrated with a general cognitive architecture.* Paper presented at the International Conference on Cognitive Modeling.

Nuxoll, A., Laird, J. E., & James, M. (2004). *Comprehensive working memory activation in Soar.* Paper presented at the International Conference on Cognitive Modeling, Poster.

Pearson, D., & Laird, J. E. (2004). *Redux: Example-driven diagrammatic tools for rapid knowledge acquisition.* Paper presented at the Behavior Representation in Modeling and Simulation, Washington, D.C.

Pew, R. W., & Mavor, A. S. (1998). *Modeling human and organizational behavior : application to military simulations*. Washington, D.C.: National Academy Press.

Rosenbloom, P. S., Laird, J. E., & Newell, A. (Eds.). (1993). *The Soar Papers: Research on Integrated Intelligence*. Cambridge, MA: MIT Press.

Salvucci, D. D. (2001). Predicting the effects of in-car interfaces on driver behavior using a cognitive architecture. In *Human Factors in Computing Systems: CHI 2001* (pp. 120-127). New York: ACM Press.

Salvucci, D. D., & Anderson, J. R. (1998). Analogy. In J. R. Anderson & C. Lebiere (Eds.), *The Atomic Components of Thought* (pp. 343-384). Mahwah, NJ: Larwrence Erlbaum Associates.

Schunn, C. D., & Anderson, J. R. (1998). Scientific Discovery. In J. R. Anderson & C. Lebiere (Eds.), *The Atomic Components of Thought* (pp. 343-384). Mahwah, NJ: Larwrence Erlbaum Associates.

Shah, K., Rajyaguru, S., St. Amant, R., & Ritter, F. E. (2003). Connecting a cognitive model to dynamic gaming environments: Architectural and image processing issues. In F. Detje, D. Doerner & H. Schaub (Eds.), *Proceedings of the Fifth International Conference on Cognitive Modeling*. Bamberg, Germany: Universitats-Verlag Bamberg.

Unit 7: Production Rule Learning. Retrieved October 31, 2004, from http://act-r.psy.cmu.edu/tutorials/unit7.htm

Vicinanza, S., & Prietula, M. J. (1993). A Computational model of musical creativity. In P. S. Rosenbloom, J. E. Laird & A. Newell (Eds.), *The Soar Papers: Research on Integrated Intelligence* (Vol. 2, pp. 974-981). Cambridge, MA: The MIT Press.

Wallace, S. A., & Laird, J. E. (2003). *Comparing agents and humans using behavioral bounding.* Paper presented at the International Joint Conference on Artificial Intelligence.

Wray, R. E., & Laird, J. E. (2003). *Variability in human behavior modeling for military simulations.* Paper presented at the Behavior Representation in Modeling and Simulation Conference, Scottsdale, AZ.

Yoshikawa, H. (2003). Modeling humans in human-computer interaction. In J. A. Jacko & A. Sears (Eds.), *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications* (pp. 119-146). Mahway, NJ: Lawrence Erlbaum Associates.

Appendix A: Reading List

Anderson, J. R., & Lebiere, C. (1998). The Atomic Components of Thought. Mahway, NJ: Erlbaum.

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of mind. Psychological Review (To Be Published).

Belavkin, R. V. (2001). The role of emotion in problem solving. In Proceedings of the AISB'01 Symposium on Emotion, Cognition, and Afective Computing (pp. 49-57). Heslington, York, England.

Byrne, M. D. (2001). ACT-R/PM and menu selection: applying a cognitive architecture to HCI. International Journal of Human-Computer Studies, 55, 41-84.

Byrne, M. D. (2003). Cognitive architecture. In J. A. Jacko & A. Sears (Eds.), The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications (pp. 97-117). Mahway, NJ: Lawrence Erlbaum Associates.

Byrne, M. D. (2003). A mechanism-based framework for predicting routine procedural errors. In R. Alterman & D. Kirsh (Eds.), Proceeding of the Twenty-Fifth Annual Conference of the Cognitive Science Society. Austin, TX: Cognitive Science Society.

Card, S. K., Moran, T. P., & Newell, A. (1983). The psychology of human-computer interaction. Hillsdale, N.J.: L. Erlbaum Associates.

Chong, R. S., & Laird, J. E. (1997). Identifying dual-task executive process knowledge using EPIC-Soar. In M. Shafto & P. Langley (Eds.), Proceedings of the Nineteenth Annual Conference of the Cognitive Science Society (pp. 107-112). Hillsdale, NJ: Erlbaum.

Gray, W. D., & Altmann, E. M. (1999). Cognitive modeling and human-computer interaction. In W. Karwowski (Ed.), International Encyclopedia of Ergonomics and Human Factors (pp. 387-391). New York: Taylor & Francis, Ltd.

Harrison, A., & Schunn, C. (2002). ACT-R/S: A computational and neurologically inspired model of spatial reasoning. In Proceedings of the 24th Annual Meeting of the Cognitive Science Society. Fairfax, VA.

Hornof, A. J., & Kieras, D. E. (1999). Cognitive modeling demonstrates how people use anticipated location knowledge of menu items. In Proceedings of the SIGCHI conference on Human factors in computing systems (pp. 410-417): ACM Press.

John, B., Vera, A., Matessa, M., Freed, M., & Remington, R. (2002). Automating CPM-GOMS. In Proceedings of the SIGCHI conference on Human factors in computing systems (pp. 147-154). New York: ACM Press.

John, B. E., Prevas, K., Salvucci, D. D., & Koedinger, K. (2004). Predictive human performance modeling made easy. In Human Factors in Computing Systems: CHI 2004 Conference Proceedings. New York: ACM Press.

Jones, R. M., Laird, J. E., Nielsen, P. E., Coulter, K. J., Kenny, P., & Koss, F. V. (1999). Automated intelligent pilots for combat flight simulation. AI Magazine, 20(1), 27-41.

Kieras, D. E., Wood, S. D., & Meyer, D. E. (1997). Predictive engineering models based on the EPIC architecture for a multimodal high-performance human-computer interaction task. ACM Transactions on Computer-Human Interaction, 4(3), 230-275.

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

Kieras, D. E., & Meyer, D. E. (1998). The EPIC Architecture: Principles of Operation. Retrieved 2-22, 2004, from www.eecs.umich.edu/people/kieras/EPIC/EPICArch.pdf

Kieras, D. E., Meyer, D. E., Mueller, S., & Seymour, T. (1999). Insights into working memory from the perspective of the EPIC architecture for modeling skilled perceptual-motor and cognitive human performance. In A. Miyake & P. Shah (Eds.), Models of working memory: Mechanisms of active maintainence and executive control (pp. 183-223). New York: Cambridge University Press.

Kieras, D. E., Meyer, D. E., Ballas, J. A., & Lauber, E. J. (2000). Modern computational perspectives on executive mental pocesses and cognitive control: Where to from here? In S. Monsell & J. Driver (Eds.), Control of cognitive processes: Attention and performance XVIII (pp. 681-712). Cambridge, MA: MIT Press.

Kieras, D. E. (2003). Model-based evaluation. In J. A. Jacko & A. Sears (Eds.), The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications (pp. 1139-1151). Mahway, NJ: Lawrence Erlbaum Associates.

Laird, J. E., & Congden, C. B. (2004). The Soar User's Manual Version 8.5 Edition 1. Retrieved August 16, 2004, from http://sitemaker.umich.edu/soar/soar_software_downloads

Lehman, J. F., Laird, J. E., & Rosenbloom, P. S. (1998). A gentle introduction to SOAR: An architecture for human cognition. In D. Scarborough & S. Sternberg (Eds.), (2 ed., Vol. 4, pp. 212-249). Cambridge, MA: MIT Press.

Lewis, R. L. (2001). Cognitive theory, Soar. In International Encyclopedia of the Social and Behavioral Sciences. Amsterdam: Pergamon (Elsevier Science).

MacLaren, B., & Koedinger, K. (2002). When and why does mastery learning work: Instructional experiments with ACT-R "SimStudents". Paper presented at the Intelligent Tutoring Systems 6th International Conference, Biarritz, France & San Sebastion, Spain.

Magerko, B., Laird, J. E., Assanie, M., Kerfoot, A., & Stokes, D. (2004). AI characters and directors for interactive computer games. Paper presented at the Innovative Applications of Artificial Intelligence Conference, San Jose, CA.

Marinier, R., & Laird, J. E. (2004). Toward a comprehensive computational model of emotions and feelings. Paper presented at the International Conference on Cognitive Modeling.

Meyer, D. E., & Kieras, D. E. (1997). A computation theory of executive control process and human multiple-task performance: Part 1. Basic Mechanisms. Psychological Review, 104, 3-65.

Meyer, D. E., & Kieras, D. E. (1997). A computation theory of executive control process and human multiple-task performance: Part 2. Accounts of Psychological Refractory-Period Phenomena. Psychological Review, 104, 749-791.

Nason, S., & Laird, J. E. (2004). Soar-RL: Integrating reinforcement learning with Soar. Paper presented at the International Conference on Cognitive Modeling, 2004.

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

Nuxoll, A., & Laird, J. E. (2004). A cognitive model of episodic memory integrated with a general cognitive architecture. Paper presented at the International Conference on Cognitive Modeling.

Nuxoll, A., Laird, J. E., & James, M. (2004). Comprehensive working memory activation in Soar. Paper presented at the International Conference on Cognitive Modeling, Poster.

Pearson, D., & Laird, J. E. (2004). Redux: Example-driven diagrammatic tools for rapid knowledge acquisition. Paper presented at the Behavior Representation in Modeling and Simulation, Washington, D.C.

Pew, R. W., & Mavor, A. S. (1998). Modeling human and organizational behavior : application to military simulations. Washington, D.C.: National Academy Press.

Polk, T. A., & Seifert, C. M. (2002). Cognitive modeling. Cambridge, Mass.: MIT Press.

Rosenbaum, D. A. (1980). Human movement initiation: Specification of arm, direction, and extent. Journal of Experimental Psychology: General, 109, 444-474.

Rosenbloom, P. S., Laird, J. E., & Newell, A. (Eds.). (1993). The Soar Papers: Research on Integrated Intelligence. Cambridge, MA: MIT Press.

Salvucci, D. D. (2001). Predicting the effects of in-car interfaces on driver behavior using a cognitive architecture. In Human Factors in Computing Systems: CHI 2001 (pp. 120-127). New York: ACM Press.

Shah, K., Rajyaguru, S., St. Amant, R., & Ritter, F. E. (2003). Connecting a cognitive model to dynamic gaming environments: Architectural and image processing issues. In F. Detje, D. Doerner & H. Schaub (Eds.), Proceedings of the Fifth International Conference on Cognitive Modeling. Bamberg, Germany: Universitats-Verlag Bamberg.

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

Taatgen, N. A., & Lee, F. J. (2003). Production compilation: A simple mechanism to model complex skill acquisition. Human Factors, 45(1), 61-76.

Wallace, S. A., & Laird, J. E. (2003). Comparing agents and humans using behavioral bounding. Paper presented at the International Joint Conference on Artificial Intelligence.

Wray, R. E., & Laird, J. E. (2003). Variability in human behavior modeling for military simulations. Paper presented at the Behavior Representation in Modeling and Simulation Conference, Scottsdale, AZ.

Yoshikawa, H. (2003). Modeling humans in human-computer interaction. In J. A. Jacko & A. Sears (Eds.), The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications (pp. 119-146). Mahway, NJ: Lawrence Erlbaum Associates.

Appendix B: Annotated Bibliography

The annotated bibliography is divided into four reading sections: General, EPIC, ACT-R, and Soar. It was compiled as the papers were read and reflects this order in both the physical ordering of the bibliography and the content of the reviews.

Foundation

Card, S. K., Moran, T. P., & Newell, A. (1983). *The psychology of human-computer interaction*. Hillsdale, N.J.: L. Erlbaum Associates.

Based on the view that a person is an information-processor, the authors develop an argument for, and a description of, applying psychology to the study of human-computer interaction. This influential chapter is still applicable today, where cognitive modeling is helping to extend the science of human-computer interaction.

In the second chapter, the authors proceed to flesh out the Model Human Processor (MHP). The MHP is composed of multiple memories and processors driven by ten principles of operation. Using a number of citations from psychology and engineering examples, details are provided on the operating parameters of the memories and processors (e.g., capacity, decay rate, cycle time) as well as some of the effects of the principles of operation. This chapter provides a mid-level overview of how a cognitive architecture would work, and how psychological measurements fit into it, forming a good introduction to cognitive architectures. One of the main questions that arose from reading this article is why didn't Newell base Soar off of the MHP? It seems that the first person to tackle this was Kieras with EPIC.

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

The first chapter describes what Newell means by a unified theory of cognition – a "… single set of mechanisms for all of cognitive behavior." Unlike the Model Human Processor (MHP), a unified theory of cognition needs to account for "problem solving, decision making, routine action, memory, learning, skill, perception, motor behavior, language, motivation, emotion, imagining, dreaming ..." The eventual goal then is not an engineering tool but to approximate human level cognitive abilities. Newell provides Soar as a candidate unified theory of cognition. Soar is essentially a symbol system based on goal hierarchies, production rules, problem spaces, and a chunking version of learning. This shift in goals explains why Soar is so different from the MHP.

The foundations of cognitive science are the subject of the second chapter. Here, Newell presents a number of familiar terms such as behaving systems, knowledge systems, representation, symbols, computation, architectures, etc. All of these terms were discussed in the context of physical (not necessarily human) systems using concrete examples. A problem with this chapter is that it is not really a chapter on cognitive science so much as a defining of terms necessary to understand human cognition in terms

of a symbol system. Only the aspects of cognitive science that are easily explained using a symbol system are discussed.

In the third chapter, Newell attempted to define a human cognitive architecture. Much of this chapter was devoted to the identification and discussion of various processing levels (neural, neural circuit, simple operations, composed operations [cognitive level]). While much of what was presented is speculation, there are three definite conclusions: the human cognitive architecture is based on neurons, cognitive behavior can be seen around the one second mark, and that "… cognitive behavior requires a symbol system." The third point, that symbol systems are required, was based on the argument that distal access and composition are required to deal with the real world. While Newell was quick to point out that much was speculation, and that the levels wouldn't necessarily have to be strong levels, more discussion on the premise that symbol systems are required is needed.

Gray, W. D., & Altmann, E. M. (1999). Cognitive modeling and human-computer interaction. In W. Karwowski (Ed.), *International Encyclopedia of Ergonomics and Human Factors* (pp. 387-391). New York: Taylor & Francis, Ltd.

This chapter provides a look at some applications of cognitive architectures, where cognitive modeling is used to design and evaluate human-computer interaction. The authors examine three different tasks, one of which is the relatively famous Project Ernestine. Using GOMS techniques, it was predicted that a new system would actually be slower to operate than the existing one. This prediction also included an explanation as to why the new system would be slower. The article goes on to describe how the three models vary over two dimensions: generative vs. descriptive and generality vs. realism. While the cases in this article do not directly refer to the cognitive architectures being investigated, the article does provide a valuable reference on possible uses of cognitive architectures.

Byrne, M. D. (2003). Cognitive architecture. In J. A. Jacko & A. Sears (Eds.), *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications* (pp. 97-117). Mahway, NJ: Lawrence Erlbaum Associates.

The chapter by Byrne is a broad review of cognitive architectures as applied to human-computer interaction. The chapter first covers a definition and history of cognitive architectures, followed by an overview of four contemporary architectures (LICAI/Colides, Soar, Epic, ACT-R/PM). A particularly informative table summarizes the similarities and differences between these architectures. Throughout the chapter, the link between cognitive architectures and human-computer interaction is also discussed. This chapter is valuable both for the overview of cognitive architectures and the description of a large variety of cognitive models built with these architectures.

Kieras, D. (2003). Model-based evaluation. In J. A. Jacko & A. Sears (Eds.), *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications* (pp. 1139-1151). Mahway, NJ: Lawrence Erlbaum Associates.

Kieras gives two solid reasons for using models to study human-computer interaction via simulation: to replace empirical observations and to provide used information about the task and/or system through formalization. He discusses examples of task network models, cognitive architecture models, and GOMS (Goals Operators Methods Selectors rules) models. Following this is an overview of a number of model-based evaluation issues. These include the role of psychological constraints, a summary of past work in cognitive modeling, modeling perceptual-motor tasks relative to cognitions, the visibility of the science base, and the role of detail. Concluding this chapter is a description of a number of GOMS models and recommendations for their use. All of the different areas covered by this paper are relevant to my topic, but the section describing some of the current issues in model-based evaluation was especially valuable.

Yoshikawa, H. (2003). Modeling humans in human-computer interaction. In J. A. Jacko & A. Sears (Eds.), *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications* (pp. 119-146). Mahway, NJ: Lawrence Erlbaum Associates.

While the topic of this chapter is similar to that in Kieras (2003) the line of research described is quite different. This chapter relies heavily on the ideas of Rasmussen, including his detailed design of human cognition and the multiladder model of control, and on the fallible machines work by Reason in sketching out a design of a human cognitive model. The author then goes on to describe the transformation from design to implementation and a number of applications of this model. This article was interesting for a couple of reasons. First, it explores some relevant prior work that I have not come across in other readings. Second, it provides a concise example of how to go about creating a new cognitive model.

Pew, R. W., Mavor, A. S., & National Research Council (U.S.). Panel on Modeling Human Behavior and Command Decision Making: Representations for Military Simulations. (1998). *Modeling human and organizational behavior : application to military simulations*. Washington, D.C.: National Academy Press.

The third chapter in this book briefly covers a wide variety of symbolic cognitive architectures, comparing them across a variety of measures. It is interesting to note that, by and large, the various architectures are quite similar. Almost all are grounded in the stage model of human information processing, have a long term and working memory, and use production rules to simulate human behavior. They do, however, differ widely in the details of implementation and in psychological validity. The descriptions of the various architectures provide a background for the decision to concentrate on EPIC, Soar, and Act-R.

EPIC

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

This paper by Kieras and Meyer is the definitive overview of the EPIC architecture. The paper provides the motivation behind EPIC as well as the details of the major pieces of this architecture and how the various processors work together. Following this is a brief description of how to construct models in EPIC and four example models. Each of these models takes advantage of the parallelism offered by the architecture in order to fit human reaction time data while performing a particular task. They conclude with the thought that cognitive psychology is important to HCI and that HCI is important to cognitive psychology.

Kieras, D. E., Wood, S. D., & Meyer, D. E. (1997). Predictive engineering models based on the EPIC architecture for a multimodal high-performance human-computer interaction task. *ACM Trans. Comput.-Hum. Interact., 4*(3), 230--275.

One of the proposed uses of cognitive architectures is as a predictive tool in human-computer interface design. Kieras, Wood, and Meyer describe the a priori modeling of a telephone operator task (Project Ernestine). Some of the important ideas presented in this paper are that of policies for a priori interface modeling, the effect of situation on task performance, the lack of tuning in a priori models, and discussions on goodness of fit. This paper also provides a couple of different GOMS examples (CPM-GOMS, NGOMSL) that are quite interesting.

Kieras, D. E., Meyer, D. E., Mueller, S., & Seymour, T. (1999). Insights into working memory from the perspective of the EPIC architecture for modeling skilled perceptual-motor and cognitive human performance. In A. Miyake & P. Shah (Eds.), *Models of working memory: Mechanisms of active maintainence and executive control* (pp. 183-223). New York: Cambridge University Press.

This paper is interesting for two main reasons. The first is that the paper describes how the basic EPIC architecture can be extended in a particular direction (e.g., auditory working memory) and then validated. The second is that this paper provides an example of how cognitive modeling can inform cognitive theory through application (e.g., duration of items in auditory working memory). Other topics of interest are a discussion of free parameters vs. experiment factors and a concise summary of the theoretical stance of EPIC.

Hornof, A. J., & Kieras, D. E. (1999). Cognitive modeling demonstrates how people use anticipated location knowledge of menu items. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 410--417): ACM Press.

Hornof and Kieras demonstrate the applicability of EPIC to the study of human-computer interactions by examining a menu selection task. Two different types of models are developed. The first requires some non-standard assumptions in order to get a good fit while the second involves a more subtle task strategy as well as tuning an error coefficient. Similar to the Kieras, Meyer, and Wood article, this paper advances cognitive

theory by pointing out several places where more study is needed. It also provides another example of model building and tuning.

John, B., Vera, A., Matessa, M., Freed, M., & Remington, R. (2002). Automating CPM-GOMS. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 147-154). New York: ACM Press.

While this paper does not discuss the EPIC architecture, it does provide a brief foray into current research in GOMS. Since GOMS and EPIC are often discussed in the same papers, and EPIC seems to have evolved from GOMS, a better understanding of GOMS provides additional insight into EPIC and EPIC alternatives. John et al. describes an automated tool that, somewhat like EPIC, takes a goal decomposition description and a number of low level templates, schedules the given activities into a PERT chart, and returns the critical path (longest time through the chart).

Kieras, D. E., Meyer, D. E., Ballas, J. A., & Lauber, E. J. (2000). Modern computational perspectives on executive mental pocesses and cognitive control: Where to from here? In S. Monsell & J. Driver (Eds.), *Control of cognitive processes: Attention and performance XVIII* (pp. 681-712). Cambridge, MA: MIT Press.

Kieras et al discuss the executive process that controls the interleaving of two tasks performed simultaneously (dual-task models), concentrating on PRP (psychological refractory period) tasks. The authors point out that one limitation of these models is the customization of the executive process for each model. They then look at the possibility of a general executive process, and its learning mechanisms, based on computer operating system research. This paper provides an in-depth review of some of the issues surrounding dual-task control.

Meyer, D. E., & Kieras, D. E. (1997). A computation theory of executive control process and human multiple-task performance: Part 1. Basic Mechanisms. *Psychological Review, 104*, 3-65.

This article is focused on multiple-task performance as well. The paper begins by providing background for some of the assumptions made in EPIC. For example, the authors discuss the multiple-resource theory that is central to the EPIC cognitive architecture and why limited-capacity assumptions are not made. This results in a set of heuristic principles that guide the design of EPIC. After a brief overview of the architecture, a number of strategic response deferment (SRD) multiple-task models are detailed. A number of conclusions are discussed, one of which is how multiple-task performance is strongly affected by perceptual-motor limitations. Meyer and Kieras cover the EPIC architecture from a different angle in this paper, concentrating more on the psychological and less on the computer science aspects.

ACT-R

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin Y. (2004). An integrated theory of mind. *Psychological Review (To Be Published)*.

Anderson et al. provide the definitive overview of ACT-R 5.0. While many of the positions are similar to that taken in EPIC, such as embodied cognition, there are still a

number of differences. These differences include processing limitations (e.g., central bottleneck theory), representation of declarative memory in chunks, subsymbolic aspects of declarative and procedural memory, as well subsymbolic and production rule learning. Particular to ACT-R is an emphasis on relating cognitive architecture functionality to brain regions.

Byrne, M. D. (2001). ACT-R/PM and menu selection: applying a cognitive architecture to HCI. *International Journal of Human-Computer Studies, 55*, 41-84.

A perceptual-motor (PM) extension to ACT-R is described in this paper by Byrne. These extensions are basically a re-implementation of the EPIC PM components with some specializations for ACT-R. One major divergence is the "where" and "what" visual buffers found in ACT-R, as well as the hard limitation of one item in these buffers at a time. EPIC has only a single channel that contains all the visible information as it is available. Two different models are presented that make use of the PM components and the results compared to prior EPIC and ACT-R models for the same task. This paper also discusses artifact (i.e. Lisp program) integration, the ability to use the same tasks as people (if it is typed up in a task specification language), and free parameters (extensively). It is in a free parameter discussion, comparing the various models, where this paper seems to get into trouble by not taking into account the free parameters introduced in creating sets of production rules.

Salvucci, D. D. (2001). Predicting the effects of in-car interfaces on driver behavior using a cognitive architecture. In *Human Factors in Computing Systems: CHI 2001* (pp. 120-127). New York: ACM Press.

This short paper contains a brief description of what is likely a large amount of work. It reports on the construction of a set of ACT-R cell phone dialing models and the general executive strategy used to merge these models with a (simplified) model of driving behavior. Following this, predicted results generated with the model are compared to an empirical study of (simulator) driving and cell phone dialing behavior. For the most part, the modeling effort portrays the same trends found in the human driving and dialing study. While the results found seem obvious, this paper is a good example of the kinds of fairly complex, if short, behaviors that can be modeled.

Harrison, A., & Schunn, C. (2002). ACT-R/S: A computational and neurologically inspired model of spatial reasoning. In *Proceedings of the 24th Annual Meeting of the Cognitive Science Society*. Fairfax, VA.

A one-page proposal on adding neurologically based 3-D modeling. This would be done by adding two different systems to ACT-R/PM. The first would be a manipulative system for 3-D representation and manipulation, based on geons (Biederman, 1987). The second system would be an ego-centric mapping and waypoint system. The main importance of this paper to the exam is to provide an example of an interesting extension to the basic cognitive architecture.

MacLaren, B., & Koedinger, K. (2002). *When and why does mastery learning work: Instructional experiments with ACT-R "SimStudents".* Paper presented at the Intelligent Tutoring Systems 6th International Conference, Biarritz, France & San Sebastion, Spain.

MacLaren and Koedinger describe an interesting use of cognitive models in a mediocre paper. Using ACT-R, they construct simulated student models and try out different curricula for algebra problems. They also look at the effects of mastery learning vs. overly general learning on algebra problem performance. This paper does have some problems though. First, the paper is of limited scope. It seems like a lot more could have been done with their experimental setup. Second, the writing lacks many of the details required to understand the results of the experiments. Despite the 10 page length of the paper it seems like only the surface has been scratched. While these two comments seem contradictory (more breadth, more depth), moving the paper in either direction would make it a much better paper.

Byrne, M. D. (2003). A mechanism-based framework for predicting routine procedural errors. In R. Alterman & D. Kirsh (Eds.), *Proceeding of the Twenty-Fifth Annual Conference of the Cognitive Science Society*. Austin, TX: Cognitive Science Society.

Byrne examines the general types of errors that occur when performing highly specialized, yet routine, procedures (e.g., those encountered by a fixed wing pilot or health care provider). The goal is to develop models that can predict both the type and frequency of errors. The author reviews previous work on errors, which is mostly taxonomic in nature (i.e. not predictive). To address this problem, Byrne describes the criteria for a framework capable of investigating procedural errors and how ACT-R satisfies the given criteria. This paper is a starting point of what looks to be a interesting line of research as well as a review of existing work in the area of routine procedural errors.

Belavkin, R. V. (2001). The role of emotion in problem solving. In *Proceedings of the AISB'01 Symposium on Emotion, Cognition, and Afective Computing* (pp. 49-57). Heslington, York, England.

This paper ties emotions directly to ACT-R parameters and demonstrates the effect that changing emotions play in problem solving. Specifically, motivation is represented as the goal reward G, arousal by the noise parameter τ, and confidence by G/τ. Some heuristics that come out of this are that positive emotions will increase motivation and confidence while negative emotions will decrease motivation and confidence. The result of this is that search will be more directed (hill climbing) as the goal is approached and more like random search when the goal seems far away. This form of simulated annealing could be a critical part of the problem solving process. While the paper is highly speculative, and some of the experiments a little odd, it does describe specifically how certain emotions could effect problem solving.

Shah, K., Rajyaguru, S., St. Amant, R., & Ritter, F. E. (2003). Connecting a cogntive model to dynamic gaming environments: Architectural and image processing issues. In F. Detje, D. Doerner & H. Schaub (Eds.), *Proceedings of the Fifth International Conference on Cognitive Modeling*. Bamberg, Germany: Universitats-Verlag Bamberg.

While the ACT-R (and EPIC) cognitive architectures generally use specialized interface (artifact) development tools that provide a symbolic version of the computer screen, SegMan is an attempt to provide generic, real-time image processing for cognitive modeling. The authors briefly describe how SegMan works and discuss an example that interfaces an ACT-R cognitive model with SegMan to drive a car in a first person driving game. Feature detection is not as fast as in a human, but good enough to drive the car within the game. This paper is preliminary work on a task of great importance to cognitive modeling.

John, B. E., Prevas, K., Salvucci, D. D., & Koedinger, K. (2004). Predictive human performance modeling made easy. In *Human Factors in Computing Systems: CHI 2004 Conference Proceedings*. New York: ACM Press.

As the title states, the paper describes a method for easy predictive modeling. Task interfaces are created using a WYSIWYG html development tool (Dreamweaver) augmented with specialized components (though these specialized components aren't required). Users can perform a task on the html pages with their actions recorded. From this, a keystroke-level model (KLM; a type of GOMS model) is produced. This KLM is then compiled into an ACT-R model. While there are still some technical difficulties, and it remains to be seen whether or not this line of research will pan out, this does sound like a better way of building non-generative cognitive models.

Taatgen, N. A., & Lee, F. J. (2003). Production compilation: A simple mechanism to model complex skill acquisition. *Human Factors, 45*(1), 61-76.

Taatgen and Lee closely examine production compilation, one of the main learning mechanisms in ACT-R 5.0. An air traffic controller experiment serves as their test bed, where the task specific model is originally specified only in declarative memory. It is up to a set of generic production rules to act on the task specific declarative chunks and eventually develop new production rules. These new rules will be faster because they complete the same subgoal with (1) fewer production rules and (2) fewer declarative memory accesses. The model results are compared to humans in the same experiment, at the overall, task, and keystroke levels. One important item discussed in the conclusion is the necessity for the model to make use of cognitive "slack" time by pursuing activities that are likely to be needed in the future in order to attain expert performance. This is somewhat an argument for the intricately orchestrated control mechanisms, such as those seen in EPIC, for modeling expert performance.

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

Symbolic learning is a central aspect of ACT-R 5.0. Taatgen and Wallach lay the foundation for the two types of symbolic learning in the architecture: instance learning and production compilation. Instance learning is the transfer of completed goals to declarative memory, while production compilation is the creation of a new production from two projections firing adjacent to each other. This new production produces the behavior of both of the previous productions. In a series of experiments, they demonstrate

with human participants and cognitive models that both types of learning can occur. The principle of rationality determines which type(s) of learning will occur for a given task.

Soar

Lewis, R. L. (2001). Cognitive theory, Soar. In *International Encyclopedia of the Social and Behavioral Sciences*. Amsterdam: Pergamon (Elsevier Science).

A whirlwind introduction and overview of the Soar cognitive architecture is presented in this paper. The introduction starts with a discussion of the functional requirements (for human level intelligence) that Soar meets as well as some of the issues surrounding cognitive architectures in cognitive science. The overview describes the five main ideas that soar is based upon: (1) physical symbol systems, (2) cognitive architectures, (3) production systems, (4) problem spaces, and (5) chunking. Following this some general Soar predictions, as well as a number of specific Soar models, are outlined. Finally, Lewis writes about a number of critiques (many centered on the *uniformity* assumption) and future directions for Soar. This encyclopedia entry is only a high-level description of Soar, but it is useful as an introduction to the architecture and where it fits in cognitive science.

Lehman, J. F., Laird, J. E., & Rosenbloom, P. S. (1998). A gentle introduction to Soar: An architecture for human cognition. In D. Scarborough & S. Sternberg (Eds.), (2 ed., Vol. 4, pp. 212-249). Cambridge, MA: MIT Press.

Soar is introduced to the reader through a simple baseball example that demonstrates the four basic elements of the architecture: goals, problem spaces, states, and operators. Through this example, details such as long term memory (production rules), working memory (goals with slots and values), the decision cycle, subgoaling, and learning (chunking) are worked through. However, this article is more than just an overview of the Soar computer program architecture. The article tackles such problems as defining cognitive architectures and the types of human behavior they need to support as well outlining some of the basic Soar assumptions. The article concludes with a brief description of a number of Soar extensions and models. A very useful article for understanding the basics of what Soar does and how it does it.

Laird, J. E., & Congden, C. B. (2004). *The Soar User's Manual Version 8.5 Edition 1*. Retrieved August 16, 2004, from http://sitemaker.umich.edu/soar/soar_software_downloads

This manual is the definitive description of the current (version 8.5.2) Soar architecture. Chapter 2 contains a deeper overview of the inner workings of Soar than seen in the previous two papers. More details are given for the general architecture and for the various components: working memory, long term memory, preference memory, execution cycles, substates, and learning. Readers interested in input/output specifications are referred to a different paper. Chapter 4 examines how chunking works in greater detail. Specifically, this chapter focuses on how the condition and actions are generated for learned chunks as well as problems that can be introduced by chunking.

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

From Chapter 4 thru the end of the book, Newell describes the Soar architecture in great detail. The chapter titles give a brief summary of the content covered: (4) symbolic processing for intelligence, (5) immediate behavior, (6) memory, learning, and skill, (7) intendedly rational behavior; and (8) along the frontiers. In addition to covering the Soar architecture and various models, much of these chapters apply to cognitive modeling in general (e.g. types of agreement between theory and data in chapter 5). This book is one of the most influential in the area of cognitive architectures.

Rosenbloom, P. S., Laird, J. E., & Newell, A. (Eds.). (1993). *The Soar Papers: Research on Integrated Intelligence*. Cambridge, MA: MIT Press.

The papers are a multi-volume collection of Soar extensions and models prior to 1992. This comprehensive set of papers serves both as a history of modeling in Soar and a reference for future models.

Chong, R. S., & Laird, J. E. (1997). Identifying dual-task executive process knowledge using EPIC-Soar. In M. Shafto & P. Langley (Eds.), *Proceedings of the Nineteenth Annual Conference of the Cognitive Science Society* (pp. 107-112). Hillsdale, NJ: Erlbaum.

Originally, this paper was included in the EPIC section. However, due to various delays it ended up in the Soar section. The paper could go either way. It combines the cognitive (problem solving and learning) elements of Soar with the perceptual and motor capabilities of EPIC to examine the executive process in dual-tasks. The two programs use sockets to communicate, based on the following cycle: EPIC perceptions -> Soar (1 decision cycle) -> EPIC motor commands. The interesting idea in this paper is how Soar's learning mechanism is used to prevent jams. Essentially, a part of the executive process is being learned. Similar to Kieras et al. (2000), this work found that prevention of jamming is not enough to reproduce results observed in humans. Various optimizations were required to produce the best resuls.

Nason, S., & Laird, J. E. (2004). *Soar-RL: Integrating reinforcement learning with Soar.* Paper presented at the International Conference on Cognitive Modeling, 2004.

Nason and Larid write about the recent addition of reinforcement learning to Soar. This is a sub-symbolic form of production utility learning. While it is similar to production utility learning in ACT-R, the authors are quick to point out a number of differences. This addition mainly centers on learning the numeric preferences that are already a part of Soar's operator preference scheme. Two different models are examined: missionaries and cannibals (a classic AI problem) and eaters (a pac man like game). This paper represents the beginning of adding RL to Soar (there is a lot of future work) as a means to compare/contrast Soar human behavior models with ACT-R models.

Nuxoll, A., & Laird, J. E. (2004). *A Cognitive Model of Episodic Memory Integrated With a General Cognitive Architecture.* Paper presented at the International Conference on Cognitive Modeling.

This paper describes how a new type of episodic, long-term memory, is added to Soar (Soar-EM). Each episode contains a complete description of working memory (WM)

whenever an operator is performed. Episodes are retrieved, on command, by finding the best partial match for a given WM state. The best performance is achieved when the matching is biased towards activation based features (i.e. activation of the features in WM) rather than when all features in WM are considered equal. This activation, described in Nuxoll, Laird, and James (2004), is similar to chunk activation in ACT-R with base activation, activation increases with production rule usage, and activation decay. Examples are given for the eaters domain. Similar to reinforcement learning, this is a preliminary extension to Soar. Soar-EM takes Soar in the direction of ACT-R and away from the assumption of uniformity and LTM as production rules.

Nuxoll, A., Laird, J. E., & James, M. (2004). *Comprehensive working memory activation in Soar.* Poster presented at the International Conference on Cognitive Modeling.

The activation mechanism required by Soar-EM (Nuxoll & Laird, 2004) is described in greater detail in this paper. Of special interest is the brief discussion of the transition from a purely symbolic Soar to a Soar that incorporates sub-symbolic elements. The rest of the paper focuses on the extension of a previous Soar activation mechanism. Most of the new features is based on the ability of i-supported working memory elements (WMEs) being able to push activation back to the creating o-supported elements, and of o-supported elements pushing activation to newly created o-supported WMEs (in place of a fixed initial activation).

Marinier, R., & Laird, J. E. (2004). *Toward a comprehensive computational model of emotions and feelings.* Paper presented at the International Conference on Cognitive Modeling.

Marinier and Laird propose and implement architectural changes to Soar (Soar-Emote) that allow affective reasoning. These changes involve integrating three different levels of emotion: biological, cognitive, and social. The changes come from the combination of two existing theories of affective reasoning. An example of Soar-Emote is given, using a simple water balloon tossing model. This model demonstrates both the architectural additions and affective model requirements. This paper describes yet another Soar architectural direction in its infancy.

Pearson, D., & Laird, J. E. (2004). *Redux: Example-driven diagrammatic tools for rapid knowledge acquisition.* Paper presented at the Behavior Representation in Modeling and Simulation, Washington, D.C.

This paper examines an aspect of modeling that has not previously been covered. It focuses on the knowledge acquisition problem in human behavior modeling. Specifically, they found that 70-90% of human behavior modeling work is the transfer of knowledge from the subject matter expert (SME) to the knowledge engineer (at a number of points in the development cycle). The Redux tool is meant to allow a SME to directly encode a number of graphical scenarios from which a rule based model will be developed through mixed-initiative or automatic methods. Additional benefits are found in both model verification and modification. Part of this effort is in direct response to the expense and difficulty of adding additional tactics to TacAir-Soar and the general brittleness and difficulties associated with large rule-based models.

Wallace, S. A., & Laird, J. E. (2003). *Comparing agents and humans using behavioral bounding.* Paper presented at the International Joint Conference on Artificial Intelligence.

Another solution to the problem of verifying human behavior models is presented by Wallace and Laird (2003). They describe an automated system that builds concise, constrained, hierarchal, goal-based models from agent behavior traces. These models are then used to automatically identify errors in the novice model with respect to the expert model. This paper could be improved in a number of ways. First, relevant prior work in the ITS literature that is very similar to their hierarchical model is ignored (e.g., Ong & Noneman 2000). Second, there is a lack of concrete examples for the core component of error identification across models. Finally, the various figures need more explanation behind them to maximize their usefulness.

Magerko, B., Laird, J. E., Assanie, M., Kerfoot, A., & Stokes, D. (2004). *AI characters and directors for interactive computer games.* Paper presented at the Innovative Applications of Artificial Intelligence Conference, San Jose, CA.

In this research, semi-autonomous agents (Soar models) are guided by a director (it is unclear whether this is a Soar model or not) to produce a believable and adaptive story for a mystery-genre computer game. These Soar models connect to agents realized in the UnrealTournament engine to provide a 3D gaming environment. While the Soar models are not described in great detail, this paper demonstrates an interesting use of Soar human behavior models in computer games.

Wray, R. E., & Laird, J. E. (2003). *Variability in human behavior modeling for military simulations.* Paper presented at the Behavior Representation in Modeling and Simulation Conference, Scottsdale, AZ.

Wray and Laird tackle the problem of introducing within- and across-subject variability to human behavior models. This problem was the driving force behind the original implementation of using a numeric (instead of a strictly random) mechanism to break ties among equivalent operators. The research they present uses this numeric mechanism to create within-subject variability based on *variability parameters*. They hope to extend it to across-subject variability through *variability profiles*. This work is related to a number of other Soar related projects. The numeric mechanism formed the basis of the Soar-RL work (Nason & Laird, 2004). This problem also depends on knowledge acquisition and validation (Pearson & Laird, 2004) as well as balancing variability and autonomy (Magerko et al, 2004).

Jones, R. M., Laird, J. E., Nielsen, P. E., Coulter, K. J., Kenny, P., & Koss, F. V. (1999). Automated intelligent pilots for combat flight simulation. *AI Magazine, 20*(1), 27-41.

TacAir-Soar is a model capable of flying a majority of the militaries planes on all of the different types of missions, and is an important model for a number of reasons. Foremost of these is the model's size (over 5200 productions, 400 operators, 100 goals) and success (types of planes, number of missions, evaluated performance). This model is also important due to the various research projects that sprang up from it. These include modification and verification of existing large rule sets (Pearson & Laird, 2004) and

automatically comparing expert and model behaviors (Wallace & Laird, 2003). Jones et al (1999) is often cited as a large-scale modeling success story.

Appendix C: List of Detailed Issues

The detailed issues are divided into four reading sections: Foundation, EPIC, ACT-R, and Soar. It was compiled after all of the papers in a given section were read.

Foundation

Human as an information processor

(Card, Moran, & Newell, 1983; Pew & Mavor, 1998)

Pew & Mavor (1998) review a large number of symbolic cognitive architectures and determine that all are based on the model of a human as an information processor. This isn't surprising as the model human processor (Card, Moran, & Newell, 1983) is described as part of an applied information processing psychology. The information-processing view sees humans as possessing various components that work together to process information and produce behavior. These components often consist of sensing and perception processors, working memory, long term memory, cognitive processor, and a motor processor.

Modeling Level

(Newell, 1990; Clark, 2001)

Newell (1990) specifies that cognitive behavior becomes evident at about one second in what he calls the cognitive band. This is based on a discussion of system levels where the time difference between levels is about an order of magnitude. Roughly, these levels are biological (up to 10ms), cognitive (100ms – 10 seconds), rational (minutes – hours), and social (days – months). He suggests that symbols are accessed around the 10ms level and that this is where an architecture should begin. The level built on top of this, around 100ms, starts cognitive activity with deliberation (accessing remote knowledge and putting it to use). Composed operations, such as pushing a button, begin around the one second level.

Other choices could be made. For example, only the items in the cognitive band could be modeled, with all lower level behavior treated as a black box. This may or may not be a good solution for a particular task based on the leakiness of the levels. Strength of levels (the lack of strength is referred to as leakiness) is the extent to which the behavior can be explained by referring only to the current level and not to lower levels (Clark, 2001).

Another way of thinking about levels is to imagine how a person would predict the behavior of another person (e.g., will my professor come to class tomorrow). The knowledge level (rational band) is used to describe rational agents making use of knowledge and goals (Newell, 1990), which roughly describes humans.

Discussion of Symbol Systems

(Newell, 1990; Clark 2001; http://www-psychology.concordia.ca/department/PSYC353/Cog2_2001_03.pdf)

Newell (1990) defines symbols as "… symbols stand for something and that the token of a symbol occurring in some place in a structure carries the interpretation that the symbols stands for something within the context that is specified because of the symbol token's location." It is also important to note that symbol systems are universal computational systems (given sufficiency and completeness).

A quote from Clark (2001) gets to the root of the symbol system problem: "it is a commitment to the existence of a computational symbol-manipulating regime *at the level of description most appropriate to understanding the device as a cognitive (reasoning, thinking) engine.*" Clark also points out that symbol systems are especially appealing because they are very similar to the common sense psychology that people use in everyday life.

Newell (1990) argues that symbol systems are necessary and sufficient for intelligence: "… (3) cognitive behavior requires symbol system." His argument is based on various properties of mental representations that symbol systems possess: productivity, systematicity, and compositionality (defined below). Another important property, from a performance perspective, is distal access. Newell wrote that not all information on an object can be processed at once so there must be some form of tokens to access related information stored somewhere else.

Productivity: There are infinitely many propositions thinkable within a finite system, so there is a need to form combinations of elementary thoughts into complex thoughts (e.g., John, Love, Mary, Paul are a small number of concepts that can create a large number of thoughts).

Systematicity: The syntactic structure of mental representations allows us to think about others with similar forms (e.g., the ability to think John loves Mary is connected to the ability to think Mary loves John).

Compositionality: The meaning of complex thoughts is a function of the elementary thoughts that form it (e.g., Mary, Love, and John have similar contributions in John Loves Mary and Mary Loves John).

Model Human Processor

(Card, Moran, & Newell, 1983)

The model human processor is an example of applied information-processing psychology. It consists of a set of connected processes (perceptual, cognitive, and motor) combined with a set of operational principles. Each processor can perform a number of basic tasks, where the average time to complete a task is gathered from the relevant psychology literature. A few of the ten principles of operation are the: (0) recognize-act cycle of the cognitive processor, (5) Fitt's law, (6) the power law of practice, (7) uncertainty principle, (8) rationality principle, and (9) problem space principle. The model human processor was designed especially for the study of computer interfaces.

Definition of a Cognitive Architecture

(Byrne, 2003; Newell, 1990)

"A cognitive architecture is a broad theory of human cognition based on a wide selection of human experimental data and implemented as a running computer simulation program" (Byrne, 2003).

Cognitive architectures have also been described as the hardware of the brain, comparing them to the hardware of the computer. The software of the brain is then the cognitive model described in this document.

"A cognitive architecture is really two things at once. First, it is a fixed set of mechanisms and structures that process content to produce behavior. At the same time, however, it is a theory, or point of view, about what cognitive behaviors have in common" (Lehman et al, 1998)

Definition of a Cognitive Model

(Byrne, 2003)

Byrne (2003) defines a cognitive model as a cognitive architecture combined with the knowledge to perform a task. It is important to consider that acquiring and codifying a task requires knowledge engineering and model programming skills. Further, a model is a program that performs a task in a particular way while there often is more than one way to perform the task.

Role of Psychological Constraints in Cognitive Architectures

(Hornof 2004; Kieras, 2003)

The predictive power of models constructed using cognitive architectures is due to the constraints imposed by and the policies of the architecture. These constraints help provide the psychological validity of a model. The goal is to allow people that aren't cognitive psychologists to construct psychologically valid models. Respecting these constraints, architectures should generally be used without modifications. Modifications that are made are generally referred to as free parameters (discussed more in under the Free Parameters section).

Science Base Visibility in Cognitive Architectures

(Card, Moran, & Newell, 1983; Kieras 2003)

Two quotations from Kieras (2003) sum up his thoughts on science base visibility: "…it is critical that the psychological assumptions be accessible, justified, and intelligible" and "the importance of documented synthesis of the scientific literature cannot be overstated." He goes on to state that updates should be frequent as the science base changes. Card, Moran, and Newell (1983) demonstrate this in the second chapter of their book, providing evidence supporting the timing parameters in their model human processor.

Modeling Perceptual-Motor vs. Cognitive Tasks

(Kieras, 2003; Newell, 1990)

Kieras (2003) writes that modeling purely cognitive tasks is generally impractical due to the open-ended nature of cognitive tasks and the lack of data on how those tasks are performed. Instead, he focuses on tasks with a perceptual-motor component as a good starting point for modeling since the task and interfaces can be well defined and the availability of psychological research for perceptual-motor tasks. On the opposite side of the coin, Newell (1990) writes that cognitive architecture should aspire to support the full spectrum of human cognition. Some of the examples in the ACT-R, EPIC, and Soar readings will provide information on the types of tasks that can be successfully modeled.

The Role of Detail

(Kieras, 2003)

Building a cognitive model is a detail-intensive task, which is a possible drawback to cognitive modeling as it makes them difficult to program (Kieras, 2003). Both the task and the interface must be completely specified along with some assumed values for architectural parameters (when there isn't enough research).

The Value of Generativity

(Gray & Altmann, 1999; Kieras, 2003)

Kieras (2003) lists generativity as one of the issues in model based evaluation. A generative model is one that is capable of adapting to a different set of inputs versus a non-generative model that is capable of performing exactly one task with completely specified inputs. The example that Kieras gives is that of a telephone operator model being able to greet a customer differently based on the information provided in a display. He also points out that most models built with cognitive architectures would be considered to be generative.

Comparison of ACT-R, EPIC, Soar

(Byrne, 2003; Pew & Mavor 1998)

The following table is a subset of the table compiled by Pew & Mavor (1998) which was cited by Byrne (2003) in the construction of a similar table. Some sections have been expanded to include additional information not found in either table. The table below outlines the different decisions made in implementing these theories of cognition.

	Soar	EPIC	ACT-R/PM
Original Purpose	Learning and problem solving	Multiple-task performance	Memory and problem-solving
Basic cycle	Decision cycle; 10 ms. However, to actually do anything takes on the order of	Production cycle (parallel); 50ms	Production cycle (conflict resolution); 50ms

	60 ms (2 DC at 10ms + 2 operators at 20ms)		
Symbolic or activation-based?	Symbolic (with recent experiments in activation)	Symbolic	Both
Sensing and Perception	Visual, Auditory	Visual, Auditory, Tactile	Visual, Auditory, Tactile
Working / STM	Unlimited capacity, duration tied to goals stack	Unlimited capacity and duration	Activation based part of LTM
LTM	Productions	Propositions and productions	Network of schema-like structures plus productions
Outputs	Behaviors	Behaviors	Behaviors
Declarative Knowledge	Productions	Productions	Schema-like structures
Procedural Knowledge	Productions	Productions	Productions
Learning	Chunking – flexible and pervasive (experimenting with other forms of learning)	No learning	Weight adjustment, production strength adjustments, new productions, new schemas
Planning	Creates new plans	Instantiates general plans	Creates new plans
Decision Making	Knowledge-based	Knowledge-based	Knowledge-based, Bayesian
Resource Representation	Serial cognitive processor, limited perceptual and motor resources	Limited perceptual and motor processors, unlimited cognitive processor	Amount of declarative memory activation
Visual attention	An operator fired by central cognition (see Newell, 1990, p. 257)	Based on where eyes are looking	Spotlight metaphor
Architectural goal management	Universal sub-goaling	None	Goal stack
Multiple Human Modeling	Yes	No	Potentially

Free Parameters / Evaluating models

(Pitt & Myung, 2002; Hornof, 2004; Byrne, 2004; Newell, 1990; Schunn, 2001; Kieras, Meyer, Mueller, & Seymour, 1999).

Pitt & Myung (2002) write about a number of issues involved in evaluating cognitive models. While there are various goodness of fit metrics (e.g. see Schunn, 2001), the number of free parameters and the plausibility of the mechanisms are important when examining how fit was achieved. The generalizability (the ability for the model to predict new, similar data) and complexity of a model must also be considered, though the definitions of these aspects are often related to free parameters.

Free parameters are adjustments that can be made to cognitive or perceptual parameters on a per model basis. In Chapter 1 of Unified Theories of Cognition, Newell (1990) points out that free parameters are much more of a problem in small, isolated theories (e.g. Fitt's Law) than for general cognitive simulations where values are approximated while searching for the correct ones.

Kieras, Meyer, Mueller, and Seymour (1999) provide two different examples. In the first, the number of free parameters is close to the number of factors in the experiment. Under these circumstances, it is hard to tell whether or not the model is actually working. The second, similar, example they provide demonstrates that the model still works when the number of free parameters is much smaller than the number of factors in the experiment.

It is important to mention one particular use of free parameters: scaling. In some models, tweaking is used to adjust parameters in order to move the model from a qualitative fit of the observed data to a quantitative fit.

Other sources for information on this come from personal communication (Hornof, 2004) and Byrne (2004).

How to Build a Cognitive Model

(Hornof, 2004)

Building a model is an architecture specific task. For EPIC, the general method is to (1) build a simulated device, (2) encode the perceptual parameters, (3) write production rules, (4) compare the predicted (model) and observed (human) results, and (5) adjust the production rules followed by returning to (4).

Why Model?

(Hornof, 2004; Newell, 1990; Kieras, 2003; Byrne, 2003)

There are two main reasons given for building cognitive models. The first is to advance psychological theory by fusing current research into a candidate unified theory of cognition and implementing the theory as a running program (Newell, 1990). This allows for the validation and refinement of theories of cognition. The second uses cognitive modeling as an engineering tool in studying human-computer interaction (Byrne, 2003; Card, Moran, & Newell, 1983; Kieras, 2003). These models are used to generate quantitative interface measures such as execution time, error rates, and learning curves

(Byrne, 2003). Other reasons for modeling include that more modeling leads to improved cognitive architectures, modeling policies, and modeling evaluation (Hornof, 2004).

EPIC

Motivation

(Kieras & Meyer, 1997)

Kieras and Meyer list three major motivations for EPIC: examining embodied cognition, creating computational models of performance and attention as well as cognition, and studying the executive processor with a focus on multiple task performance.

Key Assumptions

(Kieras & Meyer, 1997; Kieras, Meyer, Mueller, & Seymour, 1999; Meyer & Kieras, 1997)

The basic philosophy is the make the simplest assumptions first and refine later.

One of the main assumptions made by the EPIC architecture, in various areas, is that cognitive limitations are the result of perceptual processors and motor processors (including their respective working memories). This differs from the assumption that limitations are caused by bottlenecks in the central cognitive processor. The parallel rule processing of the cognitive processor is one result of this assumption. It is important to note that performance is still limited by the rate of execution (50ms cycle), perceptual/motor processors (only one set of eyes) and the various working memories.

This basic assumption is propagated into the processors as well. For example, there are no capacity limitations in verbal working memory. Instead, the number of items is limited by the decay rate of items in memory and the rate of subvocalization (i.e. rehearsal).

Meyer & Kieras (1997) provide a historical view of multiple task performance, including an examination of multiple-resource theory. This theory states that multiple, interconnected, processors complete individual tasks. Each individual processor has its own limitations. This is a generalization of the unitary-resource theory. Further, in this article they outline several heuristics that guided the construction of EPIC: *integrated information processing architecture, production-system formalism, omission of limited processing-capacity assumption, emphasis on task strategies and executive processes,* and *detailed treatment of perceptual-motor constraints*.

Architecture Description

(Kieras & Meyer, 1997)

EPIC consists of a series of interconnected processors (visual, auditory, cognitive, motor) based on the multiple-resource theory. Each processor contains its own working memory in addition to a partition in the general working memory with processor specific functionality. A brief overview of the processors follows.

The visual processor works as a pipeline. For example, the detection of a new visual object happens 50ms after it becomes visible, shape information becomes available at about 100ms, and task specific pattern recognition (e.g., an icon) may take 250ms. Visual working memory displays items and their attributes that are currently in view; the visual portion of the general working memory contains these as well as those seen previously that have not yet decayed. The availability of attributes is determined by the region (originally fovea, parafovea, and periphery). User have significant control over the visual processor due to a large number of customizable parameters. It is important to note that the visual processor does not actually recognize objects. Instead, it receives symbolic information from the device and simulates the recognition process.

The auditory processor is similar to the visual processor with minor differences. While the decay was originally fixed, an extension of EPIC resulted in a more realistic working memory. Other differences are that items in auditory memory have links to the previous/next item in memory to preserve serial order and special tags to note external or internal items.

The cognitive processor is based on user defined production (if/then) rules. This means that there is no general executive cognitive processor, a specialized one must be created for each task. This is more notable in multi-task situations rather than single task situations. Processing happens on a 50ms cycle (or optionally stochastically around 50 ms). These are discrete steps; all processing happens during a regularly scheduled cycle. During each cycle there are no central processor bottlenecks, so all rules that can fire (perceptual motor processors are limited resources) will fire. The working memory for the cognitive processor is unlimited and items put there by the cognitive processor do not decay. The cognitive section of the general working memory contains the goals and state of production rules along with general task information. Finally, there is no specific attention mechanism. All items in the general working memory are available for the production rules.

Hands, eyes, and vocal are controlled by the motor processors. Moving the hands requires specifying the style, hand, finger, direction, and extent of the motion. The specification is used to prepare and execute the movement. Only one movement can be prepared or executed at a time, though a movement can be prepared while another is being executed. Each feature takes 50ms to prepare along with a 50ms initiation delay. The actual movement time is based on Fitt's law, with a minimum of 100ms. The motor working memory can store and reuse features for similar movements. The vocalization motor is fairly basic, using mostly fixed times for different utterances. The oculomotor processors controls the eye movements in either a volunatary or involuntary mode.

Model Construction

(Kieras & Meyer, 1997)

Model construction was described by Kieras & Meyer (1997) as five basic steps: (1) create production rules (the executive process) to perform the task, (2) setup task specific parameters for the various processors, (3) select motor movement styles if they are not

specified in the task, (4) construct the simulated task environment, and (5) create the set of task instances to be examined with the model.

Parameters

(Kieras, Meyer, Mueller, & Seymour, 1999; Kieras, & Meyer, 1997)

EPIC models contain three different types of parameters: fixed, typical, and free. Fixed parameters are constrained by the architecture and cannot be adjusted by modelers. Typical parameters have default values that may be adjusted a priori. Free parameters are those that would be adjusted as needed to tune the fit of the model. As research in cognitive architectures progresses, it is desired that parameters move from free to typical to fixed.

Model Policies

(Kieras, Wood, & Meyer, 1997; Kieras & Meyer, 1997)

In order to perform a priori modeling, Kieras & Meyer (1997) write that modeling policies are required. Creating modeling policies require elaboration of the model space. Generally, modeling policies should include that strategies are based on a task analysis, can be easily represented in EPIC, and are empirically accurate. An additional constraint, not mentioned in either paper, is that they be generally plausible as well. Given a basis such as this, the next step is to determine the factors that can vary in the task strategy. In Kieras, Wood, & Meyer (1997) some of the factors were: hierarchical vs. flat and prepared in advance vs. prepared when needed. Modeling policies then allow the strategy space to be examined systematically.

Bracketing

(Kieras, D. E., & Meyer, D. E. (2000). The role of cognitive task analysis in the application of predictive models of human performance. In J. M. Schraagen & S. F. Chipman (Eds.), *Cognitive task analysis* (pp. 237-260). Mahwah, NJ: Erlbaum.)

While this paper was not included in the oral comprehensive reading list, the topic of bracketing (discussed in this paper) is one of the results attributed to the EPIC line of research. The basic idea is to "bracket" the observed results with the results from cognitive models with different strategies. One model should under-predict the observed results, the other over-predict. Bracketing provides information about what strategies are not being used.

Model Outputs

(Kieras & Meyer, 1997; Kieras, Wood, & Meyer, 1997)
EPIC outputs the predicted sequence and timing of the simulated human actions.

Limitations of EPIC

(Kieras, Meyer, James, & Lauber, 2000)

Kieras et al. (2000) writes about three limitations of EPIC. The first is a lack of a context independent executive process capable of running multiple tasks simultaneously. Each model must contain a specialized executive process created to perform the given task.

The second, related to the first, is the lack of a way to resolve resource conflicts that may result in miscommunication or deadlock. The third limitation is the lack of a procedural learning component.

General Executive

(Kieras, Meyer, James, & Lauber, 2000; Meyer & Kieras, 1997)

In order to explore the first limitation, Kieras et al. (2000) examine the idea of an EPIC general executive. They compare three different general executive to the customized executive. The conservative general executive performed worst, followed by the liberal general executive, then the executive that made use of context-dependent knowledge, and finally the original customized executive. The authors also discuss how learning may begin with conservative task strategies and change to customized strategies over time. Meyer & Kieras (1997) provide an in-depth exploration of an executive processor for psychological refractory period (PRP) tasks.

ACT-R

Motivation

(Anderson & Lebiere, 1998; Anderson, et al 2004; Byrne 2003 [from the foundation readings])

The ACT (Adaptive Control of Thought) series of theories and programs have changed significantly since their debut in 1976. Generally, the architecture has been used to study memory and problem solving with a long term goal of becoming a unified theory of cognition. In their book, *The Atomic Components of Thought,* Anderson & Lebiere (1998) make the case for ACT-R being ready for this title.

Key Assumptions

(Anderson et al. 2004; Anderson & Lebiere, 1998)

Anderson & Lebiere (1998) outline twelve of the basic assumptions of ACT-R:

1. *Technical time assumption*
    a. Acting in continuous time or at the closest grain size that matters
2. *Procedural-declarative distinction*
3. *Declarative representation*
    a. Chunks – slots (up to three or so) with values
4. *Procedural representation*
    a. Production rules
5. *Goal-directed processing*
    a. Goal stack
6. *Sources of activation*
    a. Slots in the current goal
7. *Activation in declarative memory*
8. *Production pattern matching*
9. *Production selection*
    a. Utility based conflict resolution

10. Strength in declarative memory
11. Production strength
12. Knowledge compilation
    a. Production compilation
13. Learning production rule utilities

These assumptions are discussed in practice in Anderson (2004). Included in this paper is embodied cognition, the defense of processing/buffer limitations (central bottleneck theory), the representation of memory in "chunks", the subsymbolic aspects of declarative and procedural memory, the learning involved in these subsymbolic aspects, and the learning of productions through production compilation.

One additional assumption is the principle of rationality. "The theoretical foundation of the ACT-R architecture is rational analysis of human cognition (Anderson, 1990). According to rational analysis, each component of the cognitive system is optimized with respect to the demands from the environment given its computational limitations" (Taatgen & Wallach, 2002, p. 6).

Architecture Description

(Anderson et al. 2004; Byrne, 2001; Taatgen & Wallach, 2002)

Much like EPIC, ACT-R is composed of a set serial modules running in parallel: perceptual, motor, goal, declarative memory, and cognitive (procedural memory). Further, just like EPIC, the system cycle time is 50ms. However, there are a number of differences in both the general operating principles and the modules themselves between the two architectures.

Foremost of these differences are the operating constraints on memory items. Only a single production rule can fire on any given cycle (central bottleneck theory) in the cognitive module. The cognitive module communicates with the other systems through independent buffers, each of which holds only "chunk" at a time. Other significant differences in ACT-R are the combination of symbolic and subsymbolic aspects as well as the various methods of learning. These differences will become more concrete as the individual modules are examined.

Perceptual System

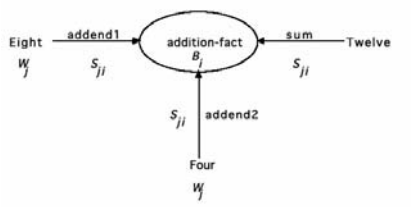
The main difference in the perceptual/motor system is the model of attention. ACT-R contains two visual buffers: *where* and *what*. *Where* contains the location and basic features of the visual items. Putting chunks in the where buffer does not require an attention shift. The *what* buffer contains identified chunks (e.g. the encoded text from a visual display) and requires a shift in attention.

Goal Module

While EPIC has a goal structure based on convention, ACT-R has a goal structure built into the model. Goals can be altered, pushed, or popped. Additionally, as goals are completed they are added to declarative memory as additional chunks.

Declarative Memory

Declarative memory is composed of items called "chunks". An example chunk from Anderson et al. (2004) is shown below. This chunk specifies that 8 + 4 = 12.



The availability and recall speed is controlled by an activation function. Chunks are activated based on the slots of the current goal, the weights and strengths of the associations, and the base activation (a separate equation) of the chunk.

As a model progresses, the activation values of the chunks will change. This is one form of subsymbolic learning in ACT-R. Of note is that some of the learning algorithms (for both declarative and procedural memory) don't scale very well. Optimizations are included that replace these with much faster approximations.

Procedural Memory

During any given cycle, multiple productions may be able to fire. However, because of the central processor bottleneck assumption only one of the productions can fire. A production is chosen to fire based on an expected utility equation. Similar to activation values, the variables in the production choice equations can also change over time based on successes and/or failures. This results in subymbolic learning for conflict resolution.

ACT-R also features a symbolic method for procedural learning called production compilation. This feature attempts to combine any two productions that occur in sequence into a single production with the effect of both productions. An example from Anderson et al. (2004):

IF reading the word for a paired-associate test and a word is being attended THEN retrieve the associate of the word
IF recalling for a paired-associate test and an associate has been retrieved with response N THEN type N

Might combine to form:

IF reading the word for a paired-associate test and "vanilla" is being attended THEN type "7"

These new production improve performance in two ways. First, one production is more efficient than two productions. Second, new productions (as in the example) can encode declarative memory thus reducing the number of declarative memory retrievals.

There are some additional complexities to production compilation. Not all production pairs can be compiled and new productions are not immediately available.

Importance of Stochastic Sampling

(Byrne, 2001; Byrne, 2003)

"Stochasticity plays multiple roles. First, the human perceptual-motor system being modeled is itself noisy. Stochasticity prevents the model from producing exactly the same response time from trial to trial. Second, averaging over noisy Monte Carlo runs tends to produce smoother responses to change than the sharp discontinuities produced by non-stochastic models. Third, the system's behavior when timing is stochastic is not always identical to that when times are fixed." (Byrne, 2001, p.13)

Model Construction

An outline of steps to create an ACT-R model:

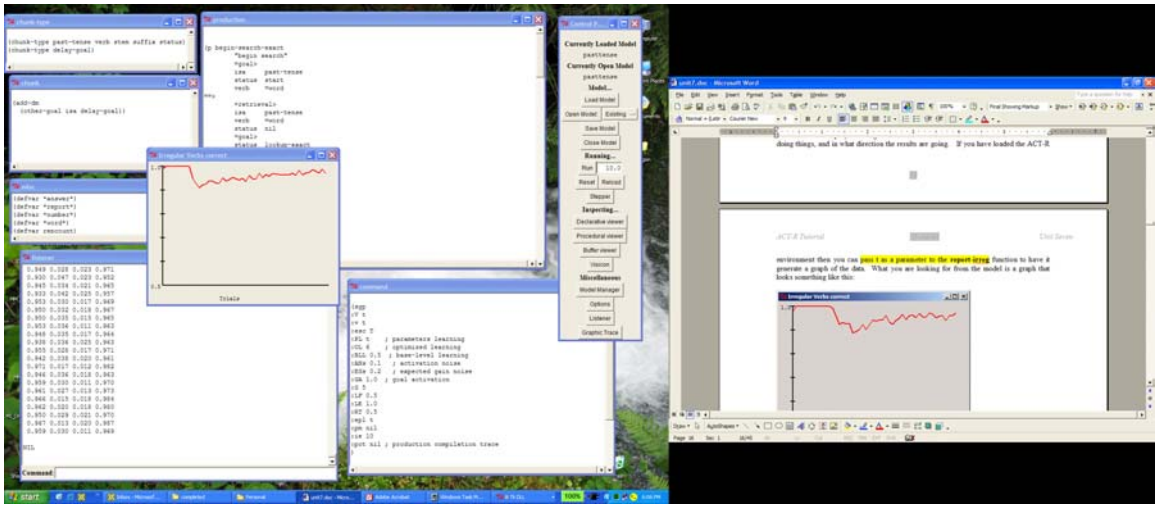
1. Create production rules and declarative memory chunks
2. Setup task specific parameters
    a. There are a lot more possible parameters in ACT-R than EPIC. Parameters for declarative memory and production utilities will likely need to be adjusted, especially for partial matching of declarative memory and production conflict resolution.
3. Construct the simulated task environment (if necessary)
4. Create task instances to be examine in the model

Lessons from the Tutorial

The tutorial covered the following areas.

- Unit 1: Understanding production systems
    - Built an addition model
- Unit 2: Perception and motor actions in ACT-R
    - Extended a screen scanning, matching, and typing model
- Unit 3: Attention
    - Built a model based around visual search, attention, and vocalization
- Unit 4: Activation of chunks and base level learning
    - Extended an alpha-arithmetic model by remembering and retrieving chunks as well as a counting strategy.
    - Performed parameter tuning to fit the data.
- Unit 5: Activation and context
    - Built a model that closely matched a distribution of simple arithmetic errors involving partial matching of declarative memory.
    - Manipulated similarities, activation noise, retrieval threshold, and base-level activations.
- Unit 6: Selecting productions on the basis of their utilities and learning these utilities
    - Built a model demonstrating learning in production rule selection for a heads vs. tails choice task.
    - Wrote LISP code to perform the task instances.
    - Constructed the simulated task environment in specialized LISP code.
- Unit 7: Production rule learning
    - Built a long-term model demonstrating production rule learning with the U-shaped curve for over-regularization.

The stand-alone version of ACT-R was used to complete the tutorial on a Windows machine. The existence of stand alone executables for a variety of operating systems is a very useful feature. However, the interface itself was somewhat disappointing. There exists a huge difference between the usability of the existing ACT-R interface (see image below) and a mature integrated development environment with multiple perspectives such as ECLIPSE for Java. Important issues include keyword highlighting, better compilation / interpretation error feedback, multiple undo functionality, consistent cut and paste functionality, command history, and better window management.



Parameters

(Anderson et al. 2004; Byrne, 2001)

Many ACT-R papers contain a discussion of free parameters. Likely, this is in response to a significant amount of criticism on the number of parameters present in ACT-R. One of the answers has been to work find typical values for many of the parameters. These parameters do not generally need to be changed across modules.

Byrne (2001) discusses another source of free parameters: artifact integration. Using the actual artifact (e.g. software program) that the human uses rather than a simulation of it eliminates the possibility of adding free parameters through the artifact.

The productions and chunks that compose the model are also a source of free parameters (Anderson et al. 2004). Ideally, the model would be constructed automatically from the task description. One of the models described by the authors does just this, building the model from an English language like model specification language. A previous version of Soar (TAQ-SOAR; Newell, 1990, p. 219) performed a similar function. However, these solutions seem to shift the free parameter problem from building the model directly to building the task specification.

Soar

Motivation

(Lehman et al, 1998; Lewis, 2001)

Soar was designed to be a candidate unified theory of cognition, focusing on human problem solving and learning.

Lehman (1998) lists a number of assumptions about what cognitive behaviors have in common that provide motivation for Soar:

7. *It is goal oriented*
8. *It reflects a rich, complex, detailed environment*
9. *It requires a large amount of knowledge*
10. *It requires the use of symbols and abstractions*
11. *It is flexible, and a function of the environment*
12. *It requires learning from the environment and experience*

Lewis (2001) also describes Soar as being driven to be able to duplicate these types of cognitive behaviors. Specifically, he cites "… (a) exhibiting flexible, goal-driven behavior, (b) learning continuously from experience, and (c) exhibiting real-time cognition (elementary cognitive behavior must be evident within about a second)" as motivating elements for Soar.

He also describes soar as the confluence of five technologies:

1. Physical symbol system
2. Cognitive architectures
3. Production systems
4. Search in problem spaces supported by a two-level automatic/deliberate control structure
5. Continuous, impasse-driven learning

Key Assumptions

(Newell, 1990, 2001)

Newell (1990) lists the main characteristics of central cognition in Soar on page 160:

7. Problem spaces represent all tasks
8. Productions provide all long-term memory (symbols)
    a. Search control, operators, declarative knowledge
9. Attribute/value representation is the medium for all things.
10. Preference-based procedure used for all decisions
11. Goals (and goal stack) direct all behavior
12. Chunking of all goal-results (impasse resolutions) occurs continuously

One additional principle, implicit in the above, is *uniformity*. For example, there is one type of LTM (productions) to be used for episodic, semantic, and procedural knowledge.

Another principle is implied by the *decision cycle* (describe in more detail in the architecture description). This principle is that all relevant information be considered before a decision is made.

Architecture Description

(Laird & Congden, 2004; Lehman et al, 1998)

The basic component of the Soar architecture is the problem space. To achieve *goals,* Soar uses *operators* to move through a *problem space* represented by *states* that consist of attributes with values and contain a goal and possibly parent and/or child states. The states, with parents and children, form a goal hierarchy. Long term memory (LTM) is made up of productions. It represents general knowledge (such as knowledge about cups). Working memory (WM) contains the current state (such as knowledge about a particular cup sitting on the desk), as well as the state hierarchy.

The *decision cycle* applies LTM to the current state. There are three stages in the decision cycle: propose operators, select operator, and apply operator. In the propose operator phase, all *elaborations*, operator propositions, and operator comparisons are fired in parallel. This phase continues until no more productions apply (quiescence). Then, from the proposed operators, one is selected.

Drilling down in this description, elaborations, operator comparisons, and operator selection are all areas that merit more description. Elaborations that are proposed by LTM elements will be added as they are true (in parallel waves) and retracted when they no longer match. The result of this is a general truth maintenance system as part of the architecture. However, this does introduce a difficulty. Operators, once fired, make a permanent change to the state. This is called o-support. Elaborations (and operator propositions and comparisons) on the other hand are only making temporary changes to the state that may be revoked any time the current state changes. This is called i-support. When creating models, whether or not something has o- or i-support has a significant effect and requires careful consideration.

Proposed operators can be given preferences to direct the operator selection mechanism by operator comparison rules. These preferences include: acceptable, reject, better, worse, best, worst, indifferent, numeric-indifferent (biased indifference), require, and prohibit.

Finally, if an operator cannot be selected an *impasse* is reached. To resolve this impasse, a new *substate* is created in which Soar attempts to resolve the impasse. This new state is a copy of the current state, but with a goal of resolving the impasse. If resolved, Soar's *chunking* mechanism creates a new LTM production to remember what to do if this impasse arises again. This new production contains the relevant features of the state prior to the impasse with the relevant action (e.g., operator comparison; operator proposal).

Here is a summary of Soar execution without substates (from Laird & Congden, 2004):

d. Input from the environment
e. Propose operators

    a. All matching elaborations, operator proposals, and operator comparisons fire in parallel.
    b. Firing continues until nothing is left to fire.
    c. These firing have I-support
f. Decide on an operator
g. Apply the operator
    d. This firing has O-support
    e. Other, I-supported rules may fire or retract based on the new state
h. Output to the environment

On Soar as a Universal Computation Engine

(Lewis, 2001; Newell, 1990)

Lewis (2001) summarizes one of the major problems facing cognitive architectures that are universal computation engines, known as the problem of *identifiability.* He defines this problem as "any sufficiently general proposal for processing schemes or representations can mimic the input/output characteristics of any other general processing or representation scheme (Anderson, 1978; Pylyshyn, 1973)." That is, why would anyone believe that Soar is any more likely to replicate human behavior than any other universal computation system?

Newell (1990) provides a response to this argument: "Soar does not automatically provide an explanation for anything just because it is a universal computational engine" (p.248)

1. Humans are capable of universal computation, so Soar must be (or, Soar is so soar predicts humans are).
2. "Universality refers only tot the class of functions that can ultimately be composed. It doesn't deal with the sorts of errors that will be made, with how fast Soar will run, and with what profile of durations of its various subcomponents. Most ways of attaining a given result can be seen not to be a way that humans do it, or indeed could do it." (p. 248)

Types of Agreement between Theory and Data

(Newell, 1990, p. 249)

"Parametric: explains variations with experimental parameters
Quantitative: explains means or typical values
Qualitative: explains order, presence or absence of effects
Consonant: assigns explanation elsewhere
No explanation
Contradiction"

Model Construction

Constructing a Soar model involves the iterative defining and refining of the following:

- States representation
    - Attributes with values
    - Objects are simply values that contain additional attributes

- Long term memory productions for:
    - Creating the initial state
    - Operator proposal
    - Operator evaluation
    - Operator application
    - Elaborating the current state
        - This includes productions that check for failure/desired states or that provide a resolution to an impasse
    - Removal of completed operators so that they may be proposed again

Lessons from the Tutorial

- Unit 1: Simple Puzzle Task
    - Water Jug model
- Unit2: Interactions with external environments
    - Basic Eaters model
- Unit 3: Subgoaling and task decomposition
    - Basic Tank Soar model
- Unit4: Problem solving and search
    - Missionaries and Cannibals model
    - State representation and operator design becomes much more important (and difficult) with this more complex problem.
- Unit5: Look-ahead planning and learning
    - Extending the Water Jug and Missionaries and Cannibals models

One of the biggest difficulties in learning to model in Soar is working with the decision cycle. For example, in the missionaries and cannibals problems, the easiest way to make a move is to develop a rule that moves each item (missionary, cannibal, boat) individually even if one of each is moving across. This is a different way of thinking from the standard, single function that moves a number of specified items.

Other difficulties include the Soar syntax. Models in Soar have a large number of special symbols and ways of arranging productions. Of the three, it is the most complex language. While the production rule interface is slightly better than that of ACT-R, the syntax checking mechanism still allows a significant number of beginner mistakes to pass through. Improvements could be made in both areas.

Debugging Soar models seems much more difficult than debugging ACT-R or EPIC models. The reason for this probably lies in the operation of the decision cycle.

Soar makes it non-trivial to do simple tasks such as allowing the same operator to fire more than once (e.g., the MOVE operator in eaters). It isn't hard to correct this, but it seems like odd default behavior.

Finally, while o-support and i-support provide a lot of advantage to Soar, they are also a liability when it comes creating and debugging models.

Issues in Human Behavior Modeling

(Magerko et al, 2004; Marinier & Laird, 2004; Pearson & Laird, 2004; Wallace & Laird, 2004; Wray & Laird, 2003)

While a number of human behavior modeling (HBM) issues are discussed, a few of them stand out. The first is the difficulty of knowledge acquisition, i.e. transferring knowledge from a subject matter expert (SME) to a cognitive modeler. According to Pearson & Laird (2004), this accounts for the bulk of the work in large cognitive models.

A second problem, discussed in both Pearson and Laird (2004) and Wallace and Laird (2004), is that of verifying and modifying a model once it has been completed. First of all, verifying the complete model is a tedious process. Second, it is a process that should be completed again whenever the model is modified. A related problem is the inherent difficulty of modifying a large rule base.

Third is the difficulty of introducing human-like variability into the model. Wray & Larid (2003) wrote "sariability seemed at odds with validation requirements (how could the system be validated if it had any variability at all?) but yet necessary for realistic models (how could we possibly claim to model human behavior without variability?." While random noise may suffice to generate average results for a population, it is not likely to accurately reflect variation at the individual level.

Future Work

(Lehman et al, 1998; Lewis, 2001)

"Still, our methodology in working toward that goal [UTC] is clear: work within the architecture, base content theories on the regularities already available from the contributing disciplines of cognitive science, and combine those content theories to try to explain increasingly complex behaviors (Lehman et al, 1998)."

General future work includes (1) providing ideas for researching complex cognitions (2) applying Soar to AI problems and (3) building and evaluation Soar models of cognitive tasks (Lewis, 2001).

Recently, a number of current Soar research papers indicate that it is moving towards the type of sub-symbolic components currently included in ACT-R. These include reinforcement learning similar to production utilities (Nason & Laird, 2004) and partial matching and production compilation for episodic memory (Nuxoll & Laird, 2004) and working memory activation (Nuxoll, Laird, & James 2004).

Appendix D: Development Environments

In using these three architectures, models are created in a different development environment. The basic modeling environment for creating cognitive models is discussed for each of the architectures. Details of creating artifacts (i.e., the task interface), while important, is not a focus area of this paper. See the documentation of the architectures for more details on artifact creation.

ACT-R

The latest version, 5.0, of the ACT-R development environment is freely available via the internet (http://act-r.psy.cmu.edu/software/). Windows, Mac, and Linux releases are all supported, with integrated development environments for Windows and Macs. The web page also contains an introductory tutorial, references, and a series of seven additional tutorials designed to familiarize the user with ACT-R. The windows installation also included a detailed user's manual.

The integrated development environment (IDE) is shown in Figure 14. Shown are separate windows for chunk types, chunks, production, system settings, debugging, buffer viewing, artifact viewing, artifact creation, logging, and IDE control. While the provided IDE is useful, it is not up to the standard of modern IDEs such as Eclipse.

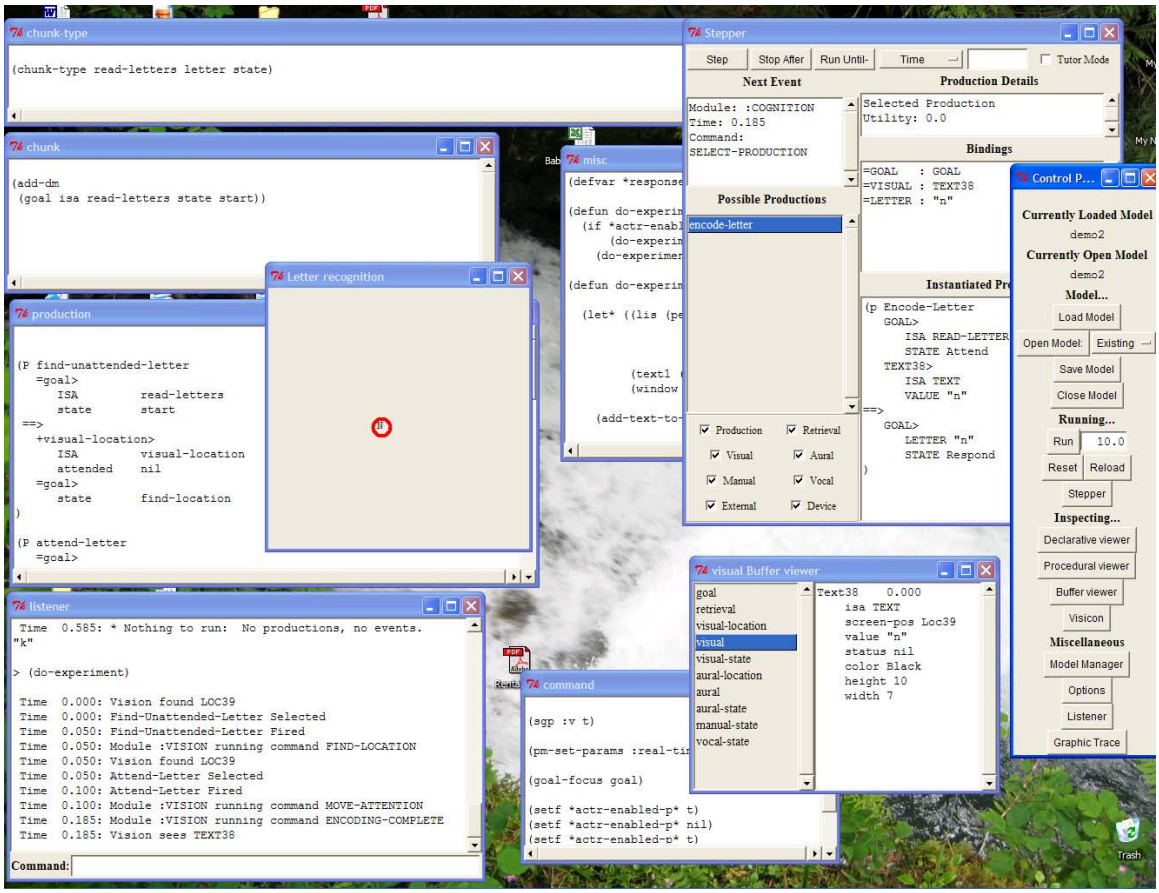


Figure 14. ACT-R IDE for Windows

EPIC

The current EPIC architecture is not available on the project website (http://www.umich.edu/~bcalab/epic.html) for downloading. Access to EPIC was obtained by taking a seminar on modeling in EPIC, where EPIC was run on a Mac. This is the only operating system currently supported by EPIC (Hornof, 2004, personal communication). Cognitive models, such as the example shown in Figure 8, are constructed in a text editor. After compiling the architecture and development environment, it is used to load cognitive models and allow artifacts to use the models to complete tasks. Similar to ACT-R, the EPIC IDE allows the user to check the contents of buffers (e.g., perceptual buffers), the state of working memory, etc. in order to debug models.

The EPIC development environment is lacking in three specific areas as compared to ACT-R and Soar. The first is the lack of a stand alone development environment. The second is that Windows and Linux are not supported. The final area is in up-to-date and comprehensive architecture documentation and introductory tutorials.

Soar

The latest Windows, Linux, and Mac versions of Soar are available on the project webpage (http://sitemaker.umich.edu/soar). The installation includes five tutorial chapters, detailed documentation, a development environment and a runtime environment. Models are created in a development environment (Figure 15) that includes a limited amount of syntactic verification and model development assistance.

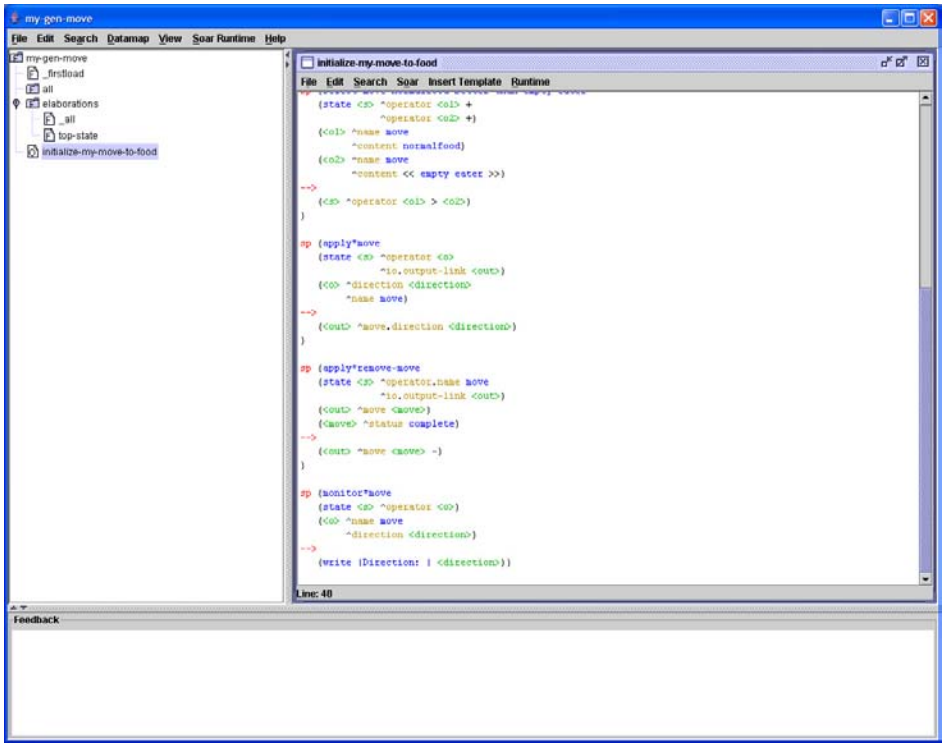


Figure 15. Soar model editor

Models are executed and debugged in the Soar runtime system, shown in Figure 16.

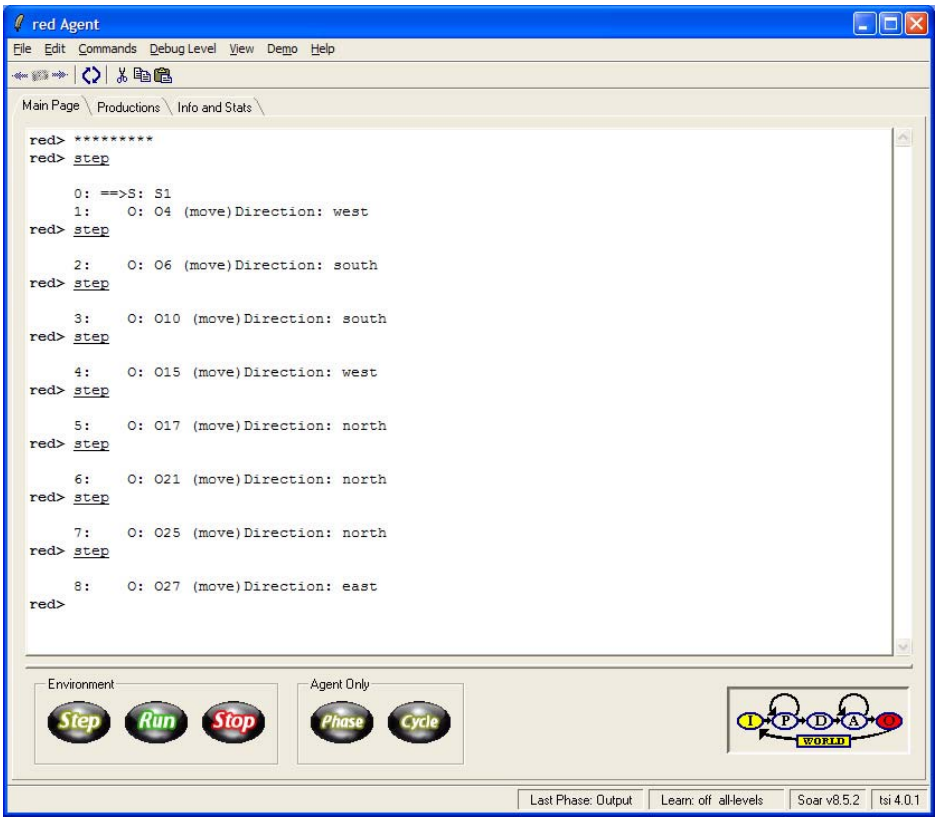


Figure 16. Soar runtime engine

While the Soar development environment has a production rule interface that is slightly better than that of ACT-R, the syntax checking mechanism still allows a significant number of beginner mistakes to pass through. There are also a larger number of special symbols (relative to ACT-R and EPIC) and ways of arranging productions that the development environment does simplify.