ABSTRACT

CORNEL, REUBEN FRANCIS. Coglaborate - An Environment For Collaborative Cognitive Modeling. (Under the direction of Dr. Robert St. Amant).

Cognitive scientists who build computational models of their work, as exemplified by the ACT-R and Soar research communities, have limited means of sharing knowledge: annual conferences and workshops, summer schools, and model code distributed via Web sites. The consequence is that results obtained by different groups are scattered across the Internet, making it difficult for researchers to obtain a comprehensive view of cognitive modeling research. The goal of my project is to develop a collaborative modeling environment for cognitive scientists in which they can develop and share models. The current system supports collaboration by providing a structured representation for ACT-R cognitive models using frames. The rationale for providing a structured representation for cognitive models is two-fold: it not only provides a mechanism for sharing models (i.e. via consistent APIs); it also enables the application of analytical techniques to cognitive models. As a proof of concept for the approach, a medium-scale modeling application has been developed, integrating an extension of ACT-R developed elsewhere, to solve synonym crossword puzzles.

Coglaborate - An Environment For Collaborative Cognitive Modeling

by
Reuben Francis Cornel

A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Computer Science

Raleigh, North Carolina

2009

APPROVED BY:

Dr. James Lester	Dr. Christopher Healey

Dr. Robert St. Amant
Chair of Advisory Committee

DEDICATION

For Mom, Dad and my kid sister Rebecca.

BIOGRAPHY

Reuben Francis Cornel was born in Mumbai, India and spent most of his life in various cities across South India. He received his Bachelor of Engineering in Computer Science and Engineering from Visweswariah Technological University. He is currently pursuing his Masters in Computer Science at the North Carolina State University. He is passionate about Computer Science and completely believes that some day computers will be as smart as humans.

ACKNOWLEDGMENTS

I would like to thank my advisor Dr Robert St. Amant for his guidance and insights through out the course of this project. I am also immensely grateful for the time he spent working with me on the draft of this report. I also thank my committee members for being exceptionally accommodating.

I express my gratitude to the members of the Knowledge Discovery Lab for making my stay in the lab extremely pleasant. I specially thank Dr Healey for picking up games that have left me with innumerable memories of happy evenings in the lab. I would like thank Thomas Horton for helping me through some really tough games, especially the volume one of "Gears Of War".

I would also like to thank all my friends and roommates for putting up with me during the course of the project. Special thanks are in order for Srinath for helping me with a lot of advice.

Most importantly I would like to thank my family for supporting me through the course of my Masters program and times when I was battling problems that were a lot larger than me. Thanks Mom, Dad and Becca.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 Cognitive Architectures

The study of the human mind has been one of the largest and most laborious enterprises undertaken by man. This study has encompassed many centuries and has attracted interest from some of the most brilliant philosophers and scientists human kind has had to offer, from Aristotle, Plato, and Descartes to modern researchers in cognitive science, including Allen Newell, John Anderson, and Marvin Minksy.

The development of computers was an important step for theories of mind, in that it allowed researchers to begin to simulate the structure of thought. Approaches vary. Some researchers, such as McClelland and Rumelhart [1, 2], have worked with abstractions of low-level biological components of the brain—artificial neurons—to produce connectionist models of the mind. Others, such as Newell and Anderson, have worked at a higher level, focusing on the manipulation of symbols as a foundation for thinking. Beyond this broad division there have been many other approaches over the past half-century.

Despite large advances made towards understanding the workings of the human mind in the field of psychology during the 70s, general principles were mainly to be found in micro theories, such as Fitts' law and the Power law of Practice. In his influential book, *Unified Theories of Cognition* [3], Newell argues that the aim of psychology is to provide a framework that would help explain and predict the behavior of the human mind, and although micro theories are useful in explaining certain phenomena they do not provide an adequate framework for explaining and predicting human thinking in more general terms.

He advocates the need for a comprehensive theory that would fill in this void.

One such theory has been under development by Anderson since the 1970s. It takes the form of a cognitive architecture, a computational simulation of problem solving constrained by empirical findings concerning human cognition. Anderson began with a framework that tried to provide a working model of the human memory, called ACT*. As this work progressed, new modules to represent motor and visual faculties were added to this framework. This came to be known as ACT-PM and eventually ACT-R. ACT-R provides the conceptual focus for this thesis.

1.2 Collaboration

A thriving research community has grown up around the core group of ACT-R researchers at Carnegie Mellon University. There are annual workshops, a "summer school" to introduce new researchers to the framework, an active mailing list, and any number of small interdisciplinary groups of collaborators distributed throughout the world. The result has been a continuous stream of refinements to ACT-R, both the theory and the software, as well as models, experiments, development tools, and the like.

This provides an opportunity, one identified in a different area of computer science, computer-supported collaborative work. J. C. R. Licklider described his vision of what he called online communities in 1968 [4]:

> There are at present perhaps only as few as half a dozen interactive multi-access computer communities. These communities are socio-technical pioneers, in several ways out ahead of the rest of the computer world: What makes them so? First, some of their members are computer scientists and engineers who understand the concept of man-computer interaction and the technology of interactive multiaccess systems. Second, others of their members are creative people in other fields and disciplines who recognize the usefulness and who sense the impact of interactive multiaccess computing upon their work. Third, the communities have large multiaccess computers and have learned to use them. And, fourth, their efforts are regenerative...
>
> What will on-line interactive communities be like? In most fields they will consist of geographically separated members, sometimes grouped in small clusters and sometimes working individually. They will be communities not of common location, but of common interest. In each field, the overall community of interest will be large enough to support a comprehensive system of field-oriented programs and data.

A wide range of tools to support online collaboration has been developed over the intervening years. The objective of my research is to provide a collaborative environment where cognitive scientists can share models, to explore the potential benefits of this approach to cognitive modeling. I describe this further in the following section and chapters.

1.3 Contributions

The aim of this project is to provide a collaborative environment for researchers who develop computational representations of their research on cognition, specifically in the ACT-R architecture. This is achieved by allowing researchers to access ACT-R on top of a web-based framework where they can create and share models. The benefits of this project are as follows:

- We provide a software environment completely set up and ready for use. As a result, researchers can get to work without being concerned about software dependency issues, a traditional barrier in collaboration as well as in ACT-R modeling.
- We attempt to foster collaboration in the cognitive modeling community. Researchers can build models and these can be accessed by other individuals and groups, and can be modified and shared in return.
- By providing a centralized system, hardware resources can be shared across a large number of users. This should in principle make it inexpensive to run and be of use to research groups that cannot invest in the resources necessary for some cognitive modeling efforts.
- Since users store their models on a centralized system, it acts as a repository for models that can be used to learn about cognitive modeling.

Frames [5] are used as the central representation to support collaboration, at the software level. A frame is a data structure consisting of slots, where each slot can either be another frame or an instance of data. Apart from a means to support collaboration, this data structure lets us represent models in a structured manner, something that has not been possible before with ACT-R. This opens up a number of further avenues for research,

such as the ability to mine models to find patterns in different models and hence build up a more complete picture of human cognition.

1.4 Thesis Organization

In Chapter 2 I discuss cognitive architectures and the features of various cognitive modeling architectures. Chapter 3 gives a brief overview of the field of computer-supported collaborative work, concentrating on research relevant to this thesis. Chapter 4 discusses the design and implementation of the Coglaborate system. I conclude this thesis with a discussion of the limitations of Coglaborate and future work for the system.

Chapter 2

A survey of Cognitive Architectures

The quest to understand the working of the human mind has spanned many centuries starting with Plato when he asked, "**How is it that human beings, whose contacts with the world are brief, personal and limited, are nevertheless able to know as much as they do?**" [6]

Cognitive science brings together the varied disciplines of psychology, neuroscience, computer science, linguistics and philosophy in an attempt to answer the above question, using information processing as a means to emulate the human mind. Psychology, especially cognitive psychology, contributes theories on cognitive capacities, information processing capabilities, sensory and motor performance, and so forth. Perhaps most importantly, it theorizes about the overall picture of the human mind. Neuroscience, the study of the brain and nervous system, provides a frame of reference against which theories developed in cognitive science can be validated since it deals with the brain at the lowest level. Computer science contributes knowledge representation, which is used to develop theories to represent the way knowledge is stored, artificial intelligence, which is used to analyze and create methods for problem solving, and the theory of computation, which is used as a means to understand foundational limits on cognition as information processing.

The objective of this chapter is three-fold. Firstly it aims to provide a very brief introduction to human cognitive architectures from both the cognitivist and emergent perspectives [7]. Secondly it discusses cognitive architectures in general. Finally it goes on to

compare some currently and widely used cognitive architectures.

2.1 The nature of cognition

Any attempt to deal with the architecture of cognition has to answer the following questions.

- *What is knowledge and how can it be categorized?* Since the aim of cognitive science is to understand the working of the human mind it is essential the nature of knowledge be understood because human beings function by processing information. Therefore it is imperative that we understand what knowledge is and the different forms of knowledge that are available.

  Although the nature of knowledge has been studied over many millennia we are still not certain of its characteristics. For example, Brachman and Levesque [8] describe knowledge as a function that maps a "knower" to a proposition. A proposition is a statement that determines the truth-value of the belief of the knower. But they also acknowledge that not all knowledge is of this form, for example, "how to" knowledge that enables someone to ride a bicycle.

- *How is knowledge acquired, represented and utilized?* When solving problems the human mind has the ability to retrieve and apply previously stored knowledge to the problem; for example, consider solving a calculus-based integration problem. We are able to retrieve standard representations of the forms of equations and apply them to the problem to simplify it and solve it. Issues of knowledge retrieval and application are central to understanding cognition.

- *How do various processes act on this knowledge and how do they achieve the effect they intend to achieve?* These two areas are significant because of their relationship to the techniques of deduction and inference that we use to solve problems on an everyday basis, inferences as simple and routine as when we diagnose a faulty light, or the more complex techniques we use when solving a crossword puzzle.

- How can these processes and structures be manifested in the real world? Purely theoretical accounts of cognition are important, but they must be validated by evidence

that they can be realized by physical systems. For scientists who study human cognition, this means ensuring that explanations of cognitive processing do not overstep the known limitations of the human brain and its processing.

These questions provide us with a very general framework for understanding results produced in cognitive science. Newell [9] describes the study of the working of the mind as a problem of satisfying the "conjunction of constraints on the nature of mind-like systems." He describes the characteristics of what is to be expected of any theory that claims to propose a model of human cognition. Newell mentions that this list is not comprehensive, but in the view of Anderson and Lebiere [10] it can used to provide a broad framework against which all theories that claim to explain the human mind can be tested.

The criteria listed below have been discussed in the literature [10, 3]. The purpose of listing them here is to explain as to what the study of the mind would require. A cognitive system must

- *Behave flexibly as a function of the environment:* At first glance this statement may seem frivolous, as it seems to imply that human cognition functions in a haphazard manner. But Newell makes it clear that he is referring to the view that a cognitive system can be viewed as an instance of a universal computer, specifically a Turing machine, despite its occasional failings and lack of infinite memory. He further explains that this view does not indicate the inability to perform special operations, for example, vision. He explains that, like computers with special processing units, the cognitive system can be made up of special purpose systems that specialize in a certain task. As an example consider a chemist who can perform cognitive tasks in the laboratory—and who can also drive a car.

- *Operate in real time:* A system that models cognition should be able to provide a plausible explanation for how humans are able to perform cognitive tasks at the rate they do. This criterion is important because otherwise a system could lead us to wrong assumptions about how humans think.

- *Exhibit rational adaptive behavior:* It must be able to explain this because humans perform computations, in the words of Newell[3], are for "the service of goals and rationally related to obtaining things that let the organism survive and propagate."

- *Display dynamic behavior:* Humans operate in an environment that is ever changing. They draw in this information from their environment and act on it appropriately. For example, if you are driving a car and at some moment a deer decides to sprint in front of the car, you would hit the brakes.

- *Integrate diverse knowledge:* Humans acquire knowledge from diverse sources and are able to integrate them. For example consider a computer programmer working in the banking industry. He can go to school to obtain knowledge of the working of the finance industry. He can use this knowledge along with his knowledge of computers science to write programs for the industry. Here we see that our fictitious programmer integrating knowledge, unlike expert systems where knowledge is vertical and cannot be integrated as easily.

- *Exhibits a sense of consciousness:* Newell does not point out the direct relation between consciousness and human cognition, but he mentions it as a criterion in his tests of human cognition. One interpretation [10] is that Newell is simply asking for the identification of cognitive properties that could support consciousness [11]

- *Learning from the environment:* This point should be self-evident, we gain new knowledge from the world around us. But then the type of learning itself should be based on whether it can learn based on semantic memory, skill, priming and conditioning.

- *Arise through evolution:* It is understood that the algorithms implicitly embodied by our cognitive processing are those that have arisen naturally over a period of time. Hence any cognitive architecture should be able to learn and improve the algorithms through a process of improvement.

- *Use of natural language:* Any theory that claims to decipher human cognition must be able to explain as to how we are able to comprehend what we listen to and understand what we say, because this is a function that is core to the way we communicate with each other.

- *Be realizable with in the brain:* This point is critical because it serves as proof that a given theory is congruous with actual computations in the brain.

2.2 Approaches towards explaining cognition

There are many theories on the nature of cognition, each taking a position on what constitutes cognitive functions and how they are carried out. But these approaches can be bifurcated into approaches that adhere to the *cognitivist approach* [7], theories that view cognition as information processes manipulating symbols, and those that adhere to the *emergent approach*, theories that treat cognition as a process where models reflect the processes of cognition by reorganization over time.

The goal of this section is to explore these disparate points of view and to bring out differences between them. We will then examine a number of cognitive architectures in detail.

2.2.1 The Cognitivist view

The cognitivist perspective views human cognition as a set of information processes working over a set of representations that point to the actual knowledge which may be stored elsewhere, vis-á-vis as symbols. These information processes are said to be purposeful, contentful, representational, and can be described formally[12]. Knowledge derived from these computations can be stored and used later to improve the reasoning of the system. The cognitivist view of perception is that it functions to generate an appropriate representation of the world around the system, which the system can then reason about [7].

The task of building models in cognitivist systems is generally done by a programmer. This has advantages, in that that these representations and structures can be viewed and interpreted by humans. But it may also bias the system, constraining it to an idealized cognitive environment. This causes problems when the system has to stray away from its original requirements. This gap between perception, which is in interpretation of reality, and actual reality begins to widen. This would then have to be filled in with more programmer knowledge to close this "semantic gap"[7].

2.2.2 The Connectionist approach

Until the 1980s the cognitivist viewpoint was the primary means of explaining the nature of human cognition. Interest in self-organizing systems led to an area of research that advocated the view that human cognition is made up of smaller units that rearrange

themselves as the system acquires a skill or recognizes a change in its environment. This approach to understanding is known as the *emergent approach* [7].

Although there are multiple methods are used to in the area of emergent systems, I will describe only connectionist view point. Connectionism is defined by Medler [13] as "a theory of information that uses parallel processing of sub-symbols, using statistical properties instead of logical rules to transform information," rather than rules as used in classical cognitivist systems.

The basic feature in a connectionist system is a network. A connectionist network is made up of a number of simple computational units that communicate with each other with via connections. These connections are capable of carrying only simple information.

The computational units in a connectionist system are arranged in a number of hierarchical layers. Conventional systems include three types of layers: the input layer, at which input signals are set; the hidden layer or layers, which allow for combination of inputs; and the output layer. These networks can be arranged into two basic configurations, namely *feed forward* networks and *recurrent* networks.

Feed forward networks(Fig 2.1) are those networks in which information flows in one direction only, that is, from the input layer to hidden layers (if they exist) and then to the output layer. Recurrent networks are those networks that have loops and hence backward connections.

Connectionist models learn by adjusting the weights on the individual computational units. This implies that learning in connectionist models can be viewed more as a skill-building exercise, rather than an exercise in knowledge acquisition as in the case of the cognitivist approaches [7].

The main attraction of connectionism is that it provides neural plausibility[12] to theories of cognitive science in its ability to simulate the massively parallel processing in the brain and in its ability to learn by adjusting weights. It is also attractive because it provides cognitive plausibility by allowing problems to be studied using simpler mechanisms. This can help in studying the processes underlying pattern-recognition and memory retrieval, as well as soft constraints for representing schematic knowledge.

Despite these attractions, connectionist modelers find it difficult to explain the ability of the human mind to integrate diverse knowledge from various sources, the ability to use pre-existing knowledge, and the ability to respond in the time constraints that humans

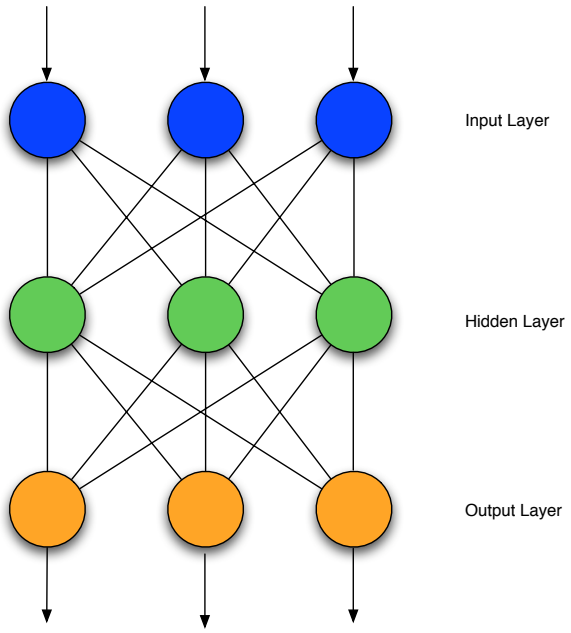

Figure 2.1: A feed forward network

do.

2.3 Cognitive Architectures

Early work in psychology focused on producing micro theories, accounts of highly constrained cognitive phenomena. Although each of these micro theories explained its individual specialization well, there were few attempts to integrate these in a complete picture of human cognition [14]. Newell's work can be seen as a call for cognitive scientists to provide a complete framework of human cognition and to test theories of cognition within it [3, 14]. Based on this we can define a cognitive architecture as

> A framework that provides an interpretation of the working of human cognition within which theories related to various aspects of cognition can be validated based on that interpretation.

A basic view of human cognition is that of solving a constraint satisfaction problem [3]. Cognitive science is still a developing field and there can be numerous ways as to

how these constraints can be interpreted and solved by researchers. Based on this view, cognitive scientists are providing us with interpretations of factors they believe constrain the way the mind works. A cognitive architecture provides us with an invariant environment where we can perform detailed modeling that helps us in understanding the processes, structures and the organization between these structures in the brain[15].

Sun [15] describes the significance of the study of cognitive architectures as follows.

- They provide a means of understanding cognition.
- They provide a foundation to build intelligent systems that are cognitively realistic.

We will now discuss criteria for comparing cognitive architecture and we will also look at some sample cognitive architectures.

2.4 Survey of cognitive architectures

In this section I will describe a number of popular cognitive architectures that are currently in use. I will later compare them based on a number of criteria that reflect the functions that a cognitive architecture is supposed to perform.

2.4.1 ACT-R

ACT-R was developed by Anderson et al. [16] at Carnegie Mellon University to validate results performed by in the area of cognitive psychology. ACT-R 's origins lie in an attempt by Anderson and Bower to describe a theory that explained the working of memory called Human Associative Memory(HAM)[17]. HAM eventually grew into ACT[18] which in due course evolved into ACT-R [19].

ACT-R consists of a set of core modules. These modules aim to represent constituent functions of the brain. The operation of these modules is coordinated by a central production system. This system can identify patterns and make changes to the contents of buffers. The organization of these modules can be seen in figure 2.2. I describe each of these modules below.

- **Visual Module**: This module provides ACT-R with the capability of identifying objects visually.

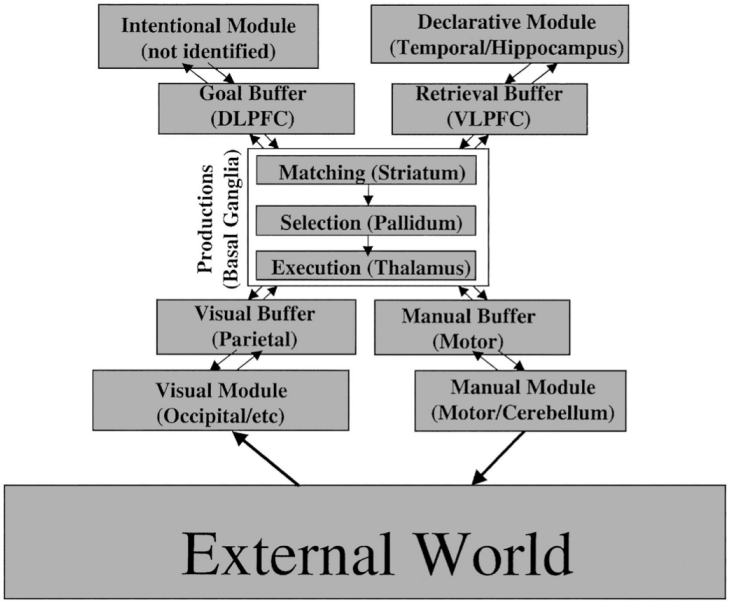

Figure 2.2: The architecture of ACT-R 5.0[16]

- **Manual Module**: This module provides ACT-R with the ability to perform motor actions.
- **Intentional Module**: This module helps ACT-R simulate the ability to keep track of current goals.
- **Retrieval Module**: This module implements the mechanism to retrieve chunks of information from the declarative memory.

In ACT-R information stored in human memory such as phone numbers and words are stored in chunks. Chunks can either be declared at the start of the model or are created during the execution of the model. Chunks represent information in the declarative memory of the architecture. The probability and the speed with which a chunk might be retrieved is controlled by its activation. This is defined by the equation

$$A_i = B_i + \sum_j W_j S_{ij}, \tag{2.1}$$

where A_i is the activation level of the chunk, B_i is the base level activation of the chunk, W_j is the attentional weighting of the elements of the current goal, and S_{ij} represents the

strengths of associations from elements j to chunk i.

The production system controls the coordination between various modules. This is done through selecting the appropriate production and firing it. Though there may be a number of competing productions, the single production with the highest utility is selected. The utility of a production is defined by

$$U_i = P_i G - C_i, \tag{2.2}$$

where P_i is the probability that the production i will achieve the goal G. C_i defines the cost of completing the goal. This is estimated by the amount of time left to complete the goal.

ACT-R learns through a process of production compilation [20]. It attempts to teach successive pairs of productions and tries to unify them into a single production. But this is not a complete explanation; it breaks down when the productions consist directives to the visual-motor modules.

The ACT-R motor-perceptual system is modeled after EPIC (executive-process / interactive-control) [21]. The perception of vision is provided by the visual module, which itself consists of two modules, the visual-location module and visual-object module. The visual-location module helps the visual module to focus its attention to a location in the visual field specified by the buffer. The visual-object module holds information about the object in the visual field. Motor control is provided by the manual module. ACT-R also has rudimentary auditory and vocal modules[16].

2.4.2 Soar

Soar is a general cognitive architecture proposed originally by Allen Newell [3, 22], when he argued for the need for psychology to deal with human cognition as a whole, rather than in form of micro strategies. The major ideas[23, 22] that define the characteristics of Soar are these:

- *Physical symbol system hypothesis*: General intelligence can be realized with a symbolic system[9].
- *Goal structure hypothesis*: Control of general intelligence is maintained by a symbolic goal system.

- *Uniform elementary-representation hypothesis*: There is a single representation for declarative knowledge.
- *Problem space hypothesis*: problem spaces are the fundamental organizational unit of all goal directed behavior.
- *Production system hypothesis*: Production systems are the appropriate encoding for long-term knowledge.
- *Universal sub-goaling hypothesis*: Any decision can be an object of goal-oriented attention.
- *Automatic sub-goaling hypothesis*: All goals arise dynamically in response to impasses and are generated automatically by the architecture.
- *Control-knowledge hypothesis*: Any decision can be controlled by indefinite amounts of knowledge that can be either domain- dependent or independent.
- *Weak-method hypothesis*: Weak methods form the basic methods of intelligence.
- *Weak-method emergence hypothesis*: Weak methods arise directly from the system responding to its environment based on its knowledge of the task.
- *Uniform learning hypothesis*: Goal-based chunking is the general learning mechanism.

According to the Soar theory, human memory is composed to two separate memories: long term memories stores information for a long period of time, and working memory stores information required for the task currently at hand.

Long term memory (LTM) stores information that can be applied a number of instances of tasks. LTM is composed of three memories: procedural memory, semantic memory, and episodic memory. Procedural memory stores information about how a task is supposed to be carried out, for example it would store information about tasks such as changing a light bulb, looking up a word in the dictionary, even eating breakfast. Semantic memory is used to store information related to facts about the environment in which the agent operates. For example, an agent tried to decide what to eat for breakfast knows the kinds of food that are generally eaten at that time of day. Episodic memory stores information about specific incidents along with contextual information. For example, the

breakfast-choosing agent might recall information from episodic memory about what it had for breakfast yesterday. Information stored in the procedural memory is accessed more frequently than information from either the semantic or episodic memories, because this information directly controls the behavior of the agent.

All information in the LTM is stored in the form of *productions*. A production consists of a set of conditions and a set of corresponding actions. Productions provide control information for the application of operators on a candidate state in working memory. Productions perform four different roles: three knowledge retrieval problem solving functions and a state elaboration function. The knowledge retrieval functions are operator proposal, operator comparison, and operator application [24]. Productions in Soar are fired in parallel without conflict resolution (unlike ACT-R , which has a conflict resolution mechanism to select between competing productions).

Preference Memory is used to decide which operator is the best in the given circumstance. Preferences in the memory are used to compare a selected set of operators with one another in an attempt to pick the best operator. Actions in productions are instrumental in creating productions. These remain active as long as the production is deemed suitable for the situation and are removed once the production is no longer valid.

Working Memory (WM) describes memory that hold information related to the task that is currently being executed. This includes states and sub-states created to solve the problem, operators that are applicable to the current states and Working Memory Elements (WME) that hold specific information about a specific identifier. Elements in working memory are created by the action of productions, a state created as a result of an *impasse*, or because of the action of external I/O systems. Information from the working memory is removed only when it becomes irrelevant to the existing situation. This architecture can be seen in figure 2.3.

The Soar theory maintains that cognition is flexible and goal-driven, that learning is a continuous process that arises from experience. This assumption is realized as a search in a problem space supported by a *recognize-decide-act* control structure[23]. The *recognize* phase consists of firing all productions that are relevant to the phase, which results in addition of new content to the working memory. The *decide* phase makes a decision about what the system should do next. If all the processes of the decide step converge to a decision, then that decision is placed in the working memory with the assertion that this decision

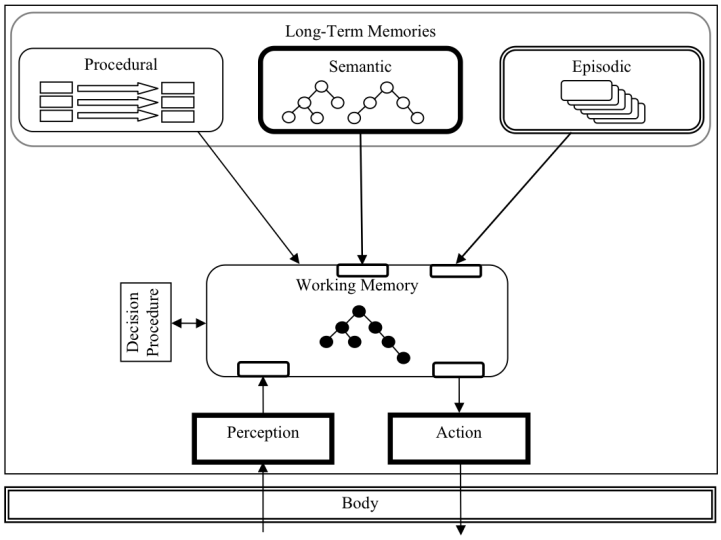

Figure 2.3: Soar Architecture[25]

would be the next step. The *act* phase involves actually realizing the step.

During the decision phase it is possible that the preferences might be incomplete or inconsistent. Such a situation is called an *impasse*. In the event that an impasse is detected, a substate of the current state is created whose goal is to resolve the inconsistency. New productions are produced as a result of this problem-solving activity. The conditions of the new productions consist of information about the state of working memory before the impasse; the actions consist of new information obtained from resolving the impasse. If a similar situation is encountered in the future, the conditions are matched and the corresponding actions are executed. This process is called *chunking*. This is how Soar obtains knowledge when solving a specific problem.

2.4.3 EPIC

The Executive-Process/Interactive-Control(EPIC) system [21] attempts to take on the challenge of developing a theory for cognition from a different perspective. The EPIC architecture maintains that human cognition is constrained by the physiological aspects of the human rather than its cognitive machinery. The motivations for developing this architecture are

- *To analyze a cognitive system that is tied closely to its perceptual-motor faculties*: Most cognitive architectures around the time when EPIC was developed represented human cognition as being *disembodied*. They represented cognition as a system that perceived the environment and directly acted on it without considering the limitation and interactions between of intermediate perceptual-motor systems and the cognitive systems.

- *Study models that simulated both performance and cognition*: Kieras and Meyer were interested in studying the field of human attention and performance from the computational modeling paradigm and hence to further the state of this psychological theory.

- *Study the role of executive processes and performance in environment that require execution of multiple tasks*: From the cognitive modeling perspective, an executive process is one that controls all other cognitive processes. Kieras and Meyer intended to study this aspect of cognition. They assert that the best way to do so is to study human performance in an environment that entails execution of multiple tasks, and that human cognitive capability is constrained only by the physical limitations.

The architecture of EPIC is shown in figure 2.4. The EPIC architecture uses separate memories to store procedural information, such as production rules and separate memory to store formal knowledge. The working memory consists of two 'amodal' types of information. One stores control information such as information pertaining to goals and steps with in the current procedure. The second memory is called *general WM* and is used to store miscellaneous task information. The EPIC architecture does not make any assumptions about decay, capacities, or representational properties of the working memory[26].

Like Soar , the EPIC architecture executes its productions in the form of a decision cycle, where each cycle lasts for a period of 50 milliseconds. The cycle starts with obtaining inputs from its perceptual processors. The production system then fires all productions whose conditions match the contents of working memory. All their corresponding actions are executed. The actions of the action items are restricted to adding and removing items from the working memory and to send commands to the motor processor.

The EPIC architecture models perceptual and motor facilities with good accuracy. The perceptual and motor modules that interface with the world are known as processors.

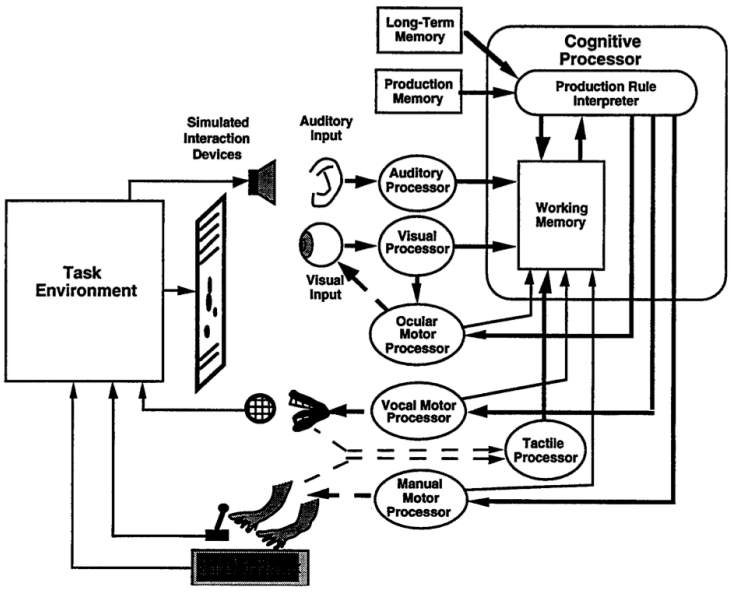

Figure 2.4: The architecture of EPIC[26]

Both the perceptual and the motor processors can function in parallel. Perceptual processors simulate the working of human perceptual facilities: these include visual processors and auditory processors. The motor processors deal with performing actions that affect the environment: some examples of motor processors are the manual motor processor, the vocal motor processor, and the oculomotor processor.

The visual processor is used to identify visual objects in the environment. The visual processor maintains information which objects are visible and what their properties are. The auditory processor accepts auditory input and outputs a representation of auditory events. These events fade away after a period of time (4 seconds). To represent the sequential order of speech, the auditory processor adds tags to the inputs in sequence to indicate temporal relationships between words.

The motor processors require two distinct phases to execute a motor command, the preparation phase and the execution phase. The preparation phase involves compiling the commands from the cognitive processor into a set of features that characterize the movement. The execution phase begins after the preparation phase has completed. The movement features remain in the processor's memory. As a result future movements that

perform the same action are performed faster. The manual processor represents physical movements like pressing keys, moving fingers and hands to another location, and so forth. The vocal motor processor is used to create utterances of speech. The oculomotor processor controls the movement of the eyes.

2.4.4 CLARION

CLARION (Connectionist Learning with Adaptive Rule Induction ON-line) [27] was developed with two major motivations.

- *Study of low-level skill acquisition*: Sun et al. [27] concentrate on low-level cognitive skills in CLARION . They cite reactive sequential decision making as an instance of low-level cognitive skill. In such a task the agent would have to pick an action to achieve an objective based on the information available at a given point in time.
- *Model learning that does not require a large amount of priori knowledge*: Sun et al. state that the process of learning in situations where agents are given the required amount of knowledge which they convert into procedural skill is considerably different from a situation where the agent is not provided with enough *a priori* knowledge. The authors suggest that in such cases the learning that occurs is of "bottom-up" nature.

Research with CLARION advocates that cognitive systems are two-level architectures. A top level encodes explicit knowledge such as production rules, explicitly specified prior knowledge. The lower level is used to store implicit knowledge, such as memory associations, in the form of neural networks. CLARION consists of four major subsystems. These are described below; the schematic of the same can be seen in Figure 2.5.

The action-centered subsystem is perhaps the most important component of the system. Its functions include calculating the quality of various actions given the current scenario, choosing an appropriate action for the situation, performing the action, and observing the change in the state. Updates are made to the internal representation during this process, including changes to the rule network (shown in the upper level of the action centered subsystem) using a rule extraction and refinement algorithm. The lower level of the action-centered subsystem also holds sensory information, working memory items, and information from the goal structure [28].

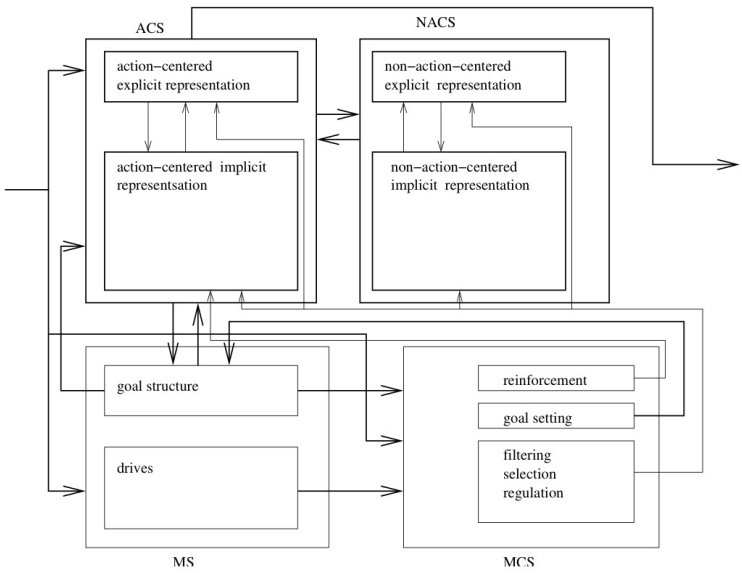

Figure 2.5: The architecture of CLARION [26]

The Non-Action-Centered subsystem is used to represent general information about the real world. A back propagation algorithm is used to establish relationships between various pairs of inputs and outputs. The Non-Action-Centered subsystem can also apply similarity-based reasoning in addition to associative rules.

Real-world cognition is driven by the environment around an agent, but it also depends on the underlying motivation of the agent. With CLARION , Sun et al. also study the underlying motivations that drive the actions an agent intends to achieve. This area of cognition is modeled and studied using the Motivational and Meta-Cognitive subsystems. These subsystems explain, for example, what makes one course of action or cognitive processing more attractive than another.

2.4.5 Criteria for comparison

This section provides a comparison of various integrated cognitive architectures on various parameters, shown in Table 2.4.5. The parameters have been selected based on the work of Anderson and Lebiere [10].

Table 2.1: Comparison of various cognitive architectures

Parameters	ACT-R	EPIC	SOAR	CLARION
Representation of knowledge in memory	Knowledge represented as chunks. Memory consists of a working memory and long-term memory.	Long-term and short-term memory where information was stored as productions in long-term memory	Knowledge in both the short term and long term memory represented as production rules	Procedural at the bottom level and declarative at the top
Goals	Maintained by the goal buffer	Priority based task deferment [29]	Represented as states	Goal stack or list
Problem Solving	Based on selection and execution of production and the underlying neural network.	Problem solving based on	Search of the solution space	Combination of calculated Q-Values and rules
Planning	Creation of sub-goals	General planning	Selection of productions that leads the system closer to achieving its goals.	Search through the Q-Value space
Learning	Production compilation	No learning	Learning through chunking	Reinforcement learning and rule extraction

Table 2.1: Comparison of various cognitive architectures

Parameters	ACT-R	EPIC	SOAR	CLARION
Perception	Provides vision and auditory faculties through the relevant buffers	Perceptual processors for auditory, visual and tactile faculties	(No explicit commitments.)	Input in terms of dimension/value pairs
Motor	Provides motor control through the manual and vocal modules	manual, vocal and oculo-motor processors	(No explicit commitments.)	
Mapping to neuro-biology	Study currently on that maps regions of the brain to specific modules	No mapping	No mapping	based on neural network but no reference to the brain anatomy[30]

2.5 Challenges facing cognitive architectures

This section provides a brief, general summary of limitations of attempts to model human cognition. Despite the tremendous amount of effort that has gone into this area, we are nowhere near the end. There are a number of research questions that are open for curious minds to explore. These areas are spelled out clearly by Langley et al. [31]:

- *Architectures need to explore categorization and understanding*: Currently cognitive architectures have not paid significant attention to the means by which we understand concepts and ideas. This is also the case with categorization, the ability of humans to categorize object that may seem dissimilar at first.

- *Accurately reflecting perceptive abilities of physical agents*: Most architectures tend to ignore the fact that a human has limited abilities to gather information relevant to a given task. Architectures need to replicate this behavior if we are to acquire a greater understanding of human cognition.

- *Further emphasis on episodic memory*: Episodic memory is a form of declarative memory that requires context and personal participation[32]; for example, the memory associations of one person's eighteenth birthday may be completely different from those of another. The focus on problem solving and skills has distracted cognitive architecture researchers from studying the complete nature of episodic memory.

- *Further study of knowledge representation*: Humans rely on a number of ways of representing knowledge using visual or auditory representations. This stands in contrast to most cognitive architectures, which have essentially been tied down to representing knowledge in the form of production rules or similar forms. There must be further study of different representational schemes that can be used (e.g., a more flexible frames-based approach [5] or description logics [33]).

- *Study of body-mind interaction*: Despite the large number of topics addressed in cognitive architecture research, the area of body-mind interaction has not been explored in detail.

- *Study of emotion*: Emotions are arguably what define humans and in quite a few cases determine their choices. Cognitive science and cognitive architecture research needs

to explore this area to get a more complete picture of human cognition.

The challenges described above are conceptual, but they also have an engineering component. From an engineering standpoint, a cognitive modeler essentially builds intelligent agents by writing programs within a framework that postulates a theory of cognition. There are situations where a similar task may be modeled by one or more researchers. In the current state of affairs, collaboration happens by a variety of informal and opportunistic ways. This results in duplication of effort and, in some cases, work that might be improved if tighter collaboration had been possible. The aim of this project is to take a step toward improving this situation.

Chapter 3

Computer Supported Collaborative Work Systems

This chapter introduces the system Coglaborate. The objective of Coglaborate is to provide a collaborative modeling environment for the cognitive science community. It does so by mounting ACT-R on the Biobike infrastructure. The system currently supports collaboration asynchronously. This means that one researcher can develop a model and share it with his colleague, who can work on that model separately and share it with other individuals.

This chapter starts off with a discussion of collaborative systems. This is followed by a discussion of biobike, its objectives, and which part of the spectrum of collaborative systems it fits into.

3.1 Collaboration

Collaboration is the key to building large structures in almost all human endeavors be it in the fields of either the arts or sciences. Collaboration permits breaking down large unwieldy tasks into more manageable chunks of work. It permits sharing of knowledge and resources. Computer networks have provided us with a means of communication between users of computers. This as a result has led to an area of research that studies how computers can help users communicate their ideas through computers and therefore use computers as a means of collaboration. This field is commonly known as Computer Supported Collaborative

Work (CSCW).

3.1.1 Advantages of collaboration using computers

Using networks of computers to support collaboration leads to three outcomes. Computers assist in sharing knowledge, for example as with a wiki, freely and easily. Computers enhance communication by allowing us to share richer content like video with each other. Finally networked computers allow us to share resources such as processing power and storage capacity. Each of these outcomes has its advantages, discussed in more detail below.

The ability to share knowledge through computers helps us save time. For example, if a researcher develops a model of some phenomenon on a computer, that model can be shared with a larger community. Some other person working on an extension of the project would save the time taken to build this model in the first place. And because it is knowledge being shared, collaboration through computers lends itself to pedagogy. For instance, consider an author of a textbook. He can easily create content, such as presentations; he can write programs to demonstrate concepts; he can share his work on a Web site. This information can later be used by other student and teachers to learn more effectively and to improve the way they teach respectively. Sharing knowledge also implies sharing data. Running experiments can be time consuming and costly. For example, a researcher studying weather in one lab could simulate a scenario and make the data for the same available for others; this saves time and cost over the entire community of people interested in the phenomena under study.

Collaboration through the means of computers provides us with richer forms of communication unavailable to us before, such as audio-based and video-based communication. As a result we can communicate with each other more comprehensibly; this in turn avoids transfer of ambiguous information. For example, we have systems currently that help us share display screen information. Such systems can be used by a customer to clearly communicate his requirements unambiguously to a vendor, who may be in a different geographical area.

The ability to tie computers together allows us to share resources. This can be in the form of hardware, where a power server or servers are shared by many people, or in the form of human resources, where people from different geographical locations can collaborate

effectively.

3.2 Classification of CSCW systems

According to Rodden [34], there are two characteristics of all CSCW systems, namely, the form of interaction and the geographical nature of the users, as shown in Figure 3.2.

The form of interaction can be described as the method by which users of a groups working together interact. This could either be synchronously, where every member of a group contributes in real time, as in a brain storming process, or asynchronously, where the members of the group interact with one another over time, not necessarily contributing simultaneously. An example of this might be a group of students collaborating with each other on a homework problem via a message board.

The geographical nature of the users describes whether the users interact with each other remotely or locally. For example, consider the case of developing the linux kernel: developers and testers worked with each other on the kernel in geographically disparate locations, remotely. Alternatively, users may be co-located, as with the users of a meeting room system like Colab [35].

3.2.1 Message Systems

Message systems derive their origins from email-based systems. What differentiates message systems for CSCW from email-based systems is that such message systems try to attach more semantic data to the messages they process. Most of these systems use an object model to represent messages. To instantiate messages the user fills context-specific data into slots; this data is used by the system to carry out further processing by the system. Examples of message systems for CSCW include COSMOS [36] and Information Lens [37].

COSMOS represented research being carried out in the European CSCW community. COSMOS was a project in the UK to design and develop a configurable message system that supported structured group work. The system was developed with a focus on specifying the users' communicative work in the form of participants actions. Group activity tasks are represented in the system using the concept of *communication structures* (CS) [36]. Communication structures are further described using a structure definition lan-

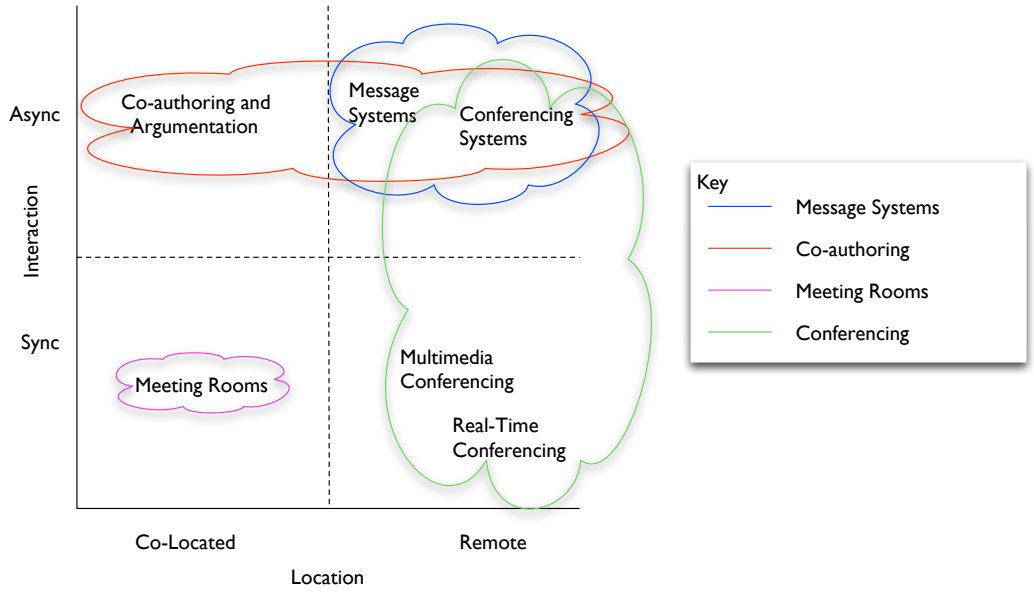

Figure 3.1: Classification of CSCW systems[34]

guage (SDL). The tasks a COSMOS system is capable of carrying out are defined by the communication structures in the system. The COSMOS system also defines roles for users (or agents), message objects, actions, rules, and actions in context of the work being carried out. Any communication activity by the user is instantiated when the user creates an instance of a communication structure. The user does this by filling in the data required by the communication structure. Further communication occurs by an exchange of messages between users, who compose a specific role specified in the communication structure.

The Information Lens takes a less formal approach [34]. The aim is to develop information systems that minimize information overload on the users of the system caused by the amount of incoming messages. The key ideas that defined Information lens are

- *Use of semi-structured messages to represent data:* Messages are represented as a set of semi-structured data called frames. A frame is a template that includes information for date, time, place, organizer, and any other unstructured data. This idea plays an important role because semi-structured messages make it easier for the computer to parse messages, according to informal studies conducted by the developers of the system [37]. People do most of their processing on a set of unstructured information

and thus the use of semi-structured fields, rather than more formal structures, enables authors to include more context-related information than would otherwise be practical.

- *Use of production rules:* The system uses sets of production rules, each of which might contain multiple levels of reasoning that specify how to process messages.
- *Use of specialized editors for creating and editing message templates and productions:* The Information Lens is made more usable with multiple editors to create and edit message templates and productions.
- *Use of frame inheritance:* Using frame inheritance in messages allows messages to be specialized very easily.
- *Introduction of the system slowly to the users:* The developers believed that the introduction and evolution of a group work system is more useful if it is introduced slowly to users. Users are encouraged to switch to the system by incremental rewards.

The message templates consist of forms having fields where information could be filled in. Messages are edited using a display-oriented editor. The users construct the IF part of a rule by specifying the conditions that are required to satisfy specific conditions, using logical operators like *and*, *or*, and *not*. The users rely on message-handling primitives like *move*, *delete*, *save*, etc. to specify the action to be taken for that message.

3.2.2 Conferencing systems

Conferencing systems were first developed in the 1970s when the US government was looking for a way to deal with emergencies. EMISARI [38] (Emergency Management Information System and Reference Index) was one result of this effort. It consists of two main ideas that even today form the backbone of current conferencing systems, as illustrated in Figure 3.2.2. This model of communication allows users to interact through a shared information space, and at the same time it allows users to interact individually with one another.

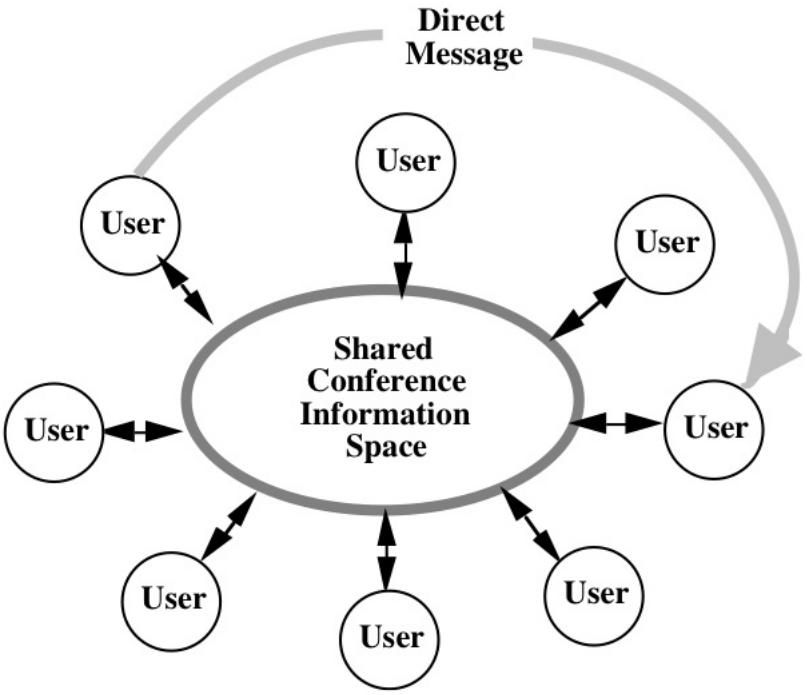

Figure 3.2: Communication in a conference system[34]

Traditional conferencing systems

Traditional conference systems are also extensions of email systems. The users of a system can subscribe to various conferences and the system sends messages as comments are posted. These are different from mailing lists in that they allow discussions to be moderated by users. They also provide flexibility in the way conferences are created. Examples of traditional conference systems include NOTEPAD, by InfoMEDIA Corp, and Confer, a conferencing system developed at the university of Michigan.

Real-time conferencing systems

Real-time conferencing enables an instantaneous transfer of ideas and flow of information between participants of the conference. Real-time conferencing is important in situations that require real-time group participation. RTCAL [39] was a system developed at the MIT Laboratory for Computer Science. It supported scheduling meetings by building a shared workspace of information from the participants' calendars. Although it did not

provide a tool that automatically selects times for meetings, it provides a set of decision support tools to help schedule meetings. RTCAL demonstrated a number of features that are still relevant and applicable to most real time conferencing systems today, most notable Microsoft Live Meeting.

Real-time conferencing systems may sometime also provide the user with the facility to share screens. These are called shared screen systems. Much of the work on shared screen systems are developed on the work of Colab [35]. Yuuguu is a modern example of an instant screen sharing system.

3.2.3 Meeting Rooms Systems

Meeting room systems find their roots in work on Group Decision Support Systems (GDSS). Kraemer [40] cites DeSanctis and Gallupe[41] in a definition of a Group Decision Support System as "an interactive computer-based system that facilitates the solution of unstructured problems by a set of decision makers working together as a group." The eventual aim of such systems is to help groups improve their decision making process, by providing structure for the process and a set of tools that help groups to make better decisions.

More modern meeting room support systems are designed to enable collaboration between co-located groups of participants. This is typically arranged in a meeting room that contains a large projector along with a number of individual terminals. Figure 3.3 illustrates this arrangement.

CoLab [35] is one of the earliest and most influential examples of a meeting room system. It aimed to explore the idea of using computers in meeting rooms to carry out tasks that generally required media, such as chalkboards. Although the use of chalkboards and similar media generally helps with group focus, these media have limitations. For example, the contents of a standard chalkboard cannot be reproduced once erased, and rearranging items on a chalk board is inconvenient.

CoLab attempted to provide a way around these drawbacks by having a shared information space where participants of a conference interacted with one another. This led to investigation of multi-user interfaces, where users can interact with one another using a computer. One concept that resulted from this work was What You See Is What I See (WYSIWIS): every user sees exactly the same information. Work on this concept produced

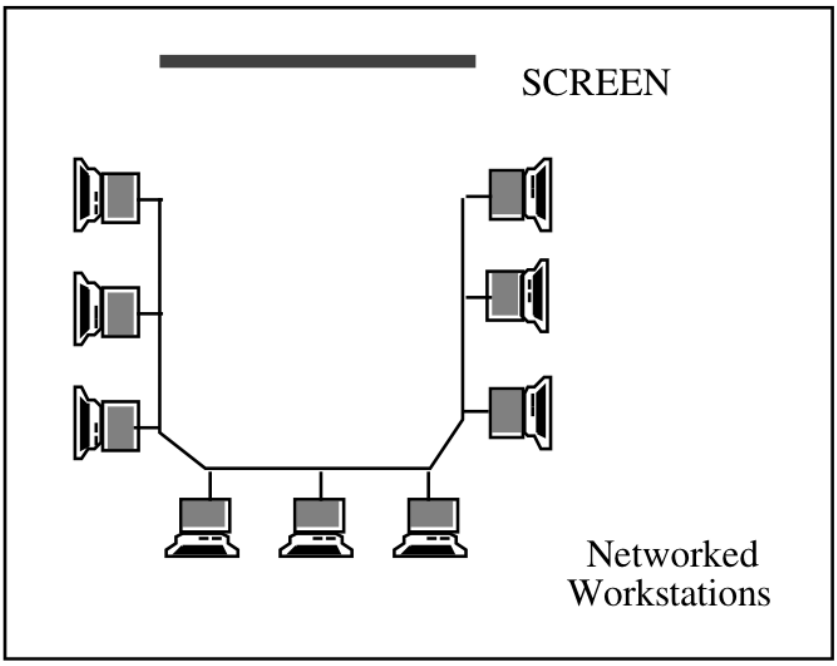

Figure 3.3: Arrangement of a meeting room system[34]

the finding that strict WYSIWIS interfaces are too limiting. The designers of CoLab thus opted for a more relaxed interpretation of WYSIWIS, where every user had private windows in addition to a display shared with everyone else for interaction.

Since CoLab allowed a number of collaborators to interact with one another, its basic design led conflicts between individual users trying to modify the same object at the same time. As a result the team experimented with a number of mechanisms to maintain concurrency [35].

3.2.4 Co-authoring Systems

A number of projects can be accomplished only by the effort of a number of authors. Co-authoring systems help support such activity. A wiki can be considered an example of such a system. Any user can read an article and, when required, they modify the content of the article. A wiki also provides space for authors to discuss posted information via a separate space for comments. A wiki provides version and configuration control when saving the updates made to a given document.

3.3 Coglaborate

Coglaborate, the system developed for this thesis, aims to support collaboration between cognitive scientists who build computational models of their work. Currently these users are restricted to sharing knowledge through the informal means of annual conferences, workshops, summer school and model code distributed via web sites. Coglaborate is a collaborative modeling environment in which cognitive modeling researchers can develop and share models.

Coglaborate currently provides the following features, which are described in detail in Chapter 4:

- A live lisp listener in a web browser where code can be evaluated.
- A structured representation for models using frames.
- A method for sharing and examining models at any level of detail.

Coglaborate provides support for collaboration by compiling the user's models into a frame-based representation. These representations are stored in a common area of the memory of the system. Any user who is familiar with the name of the model can search for the model and convert the model to code. This code can then be edited and recompiled, as a result of this process the original representation is modified to reflect the change.

Coglaborate can be classified as a co-authoring system. The rationale behind choosing this classification is that different users can build the same models incrementally by adding productions or editing existing models. The following chapter will examine in detail the design and implementation of the system.

Chapter 4

Design and Implementation

The purpose of this chapter is threefold: it discusses the motivations that govern the design of the Coglaborate system, such as using the BioBike chassis [42] and using frames as a data structure to represent ACT-R based cognitive models; it describes the work flow of the system that enables users to build and share models; finally it discusses in detail a medium-scale model that was built as a proof of concept, a demonstration of how Coglaborate would be used in practice.

4.1 Problem Definition for Coglaborate

The primary goal of the Coglaborate system is to provide a collaborative tool-based environment for the computational cognitive modeling community. Currently there are no such environments that provide this facility. Apart from supporting collaboration, Coglaborate also targets a number of secondary goals:

- *A sandbox environment*: Currently there are number of interesting ACT-R extensions available to modelers. But occasionally it might be difficult to set up these extensions. With a sandbox environment, modelers can connect to it and use the module with out any setup cost. This has a number of advantages. The users do not have to keep track of the version of the extension and they can focus on completing their tasks rather than having to tinker with their environment. Further, developers can work to improve existing extensions without as much concern for deployment issues.

- *Resource Sharing*: When we refer to resource sharing we mean sharing hardware. With a Web-based platform that runs on a powerful computer, researchers can simulate models that require excessive computational power and time.

- *Tool Integration*: If there are computational tools that a model requires they can be hooked in once, as a shared software resource. For example, the R statistical environment can be accessed as a library. Once loaded, the facilities can be accessed by many users, who do not need to modify their local environments.

- *Knowledge Agglomeration*: As modelers explore various aspects of cognition, they can contribute their results to a general repository that reflects the state of the art.

4.2 Design Decisions

Two major decisions influenced the design of the project: firstly the use of the BioBike chassis as a platform to mount the ACT-R framework, and secondly the use of frames as means to represent ACT-R based cognitive models in the shared memory. This section discusses the alternatives that were analyzed and the reasons we made these decisions.

4.2.1 Choice of platform

We chose the BioBike framework as the platform for the system. This section examines the basis of that decision by analyzing the capabilities of BioBike. BioBike is an instantiation of KnowOS [43], a concept based on knowledge being treated on par with other elements that make up a computer system. Operating systems provide useful abstractions for users to work with the elements of a system. A simple example is a file, an abstraction over regions of hardware storage. An operating system provides functions by which this file can be renamed, copied, deleted, and so forth. It also provides further abstractions that allow significant modifications to raw data. The KnowOS vision extends this analogy to the realm of knowledge. An implementation of the KnowOS consists of the following layers [43]:

- A knowledge base centered on the frame system.

- An efficient and extensible programming language that provides abstractions for users to work with the system.

- A Web-based interface to the programming language and to other KnowOS services.

BioBike (originally known as BioLingua) provides biologists with the ability to perform computational biology operations on large data sets using a simple language. BioBike ties a number of knowledge bases together transparently, using a frame-based representation to represent organisms. As an implementation of the KnowOS vision, it provides features customized for molecular biologists. These include [42]

- A common framework to access genomic, metabolic and experimental data.
- A general-purpose programming language (Lisp) customized for transparent access to the underlying knowledge bases.
- A highly interactive environment where code can be evaluated and its results displayed immediately.
- A number of general-purpose tools that help in analyzing interactions.
- A wiki through which scientists can collaborate and announce results.

The BioBike language is built on top of Common Lisp. Users interact with the system using a Lisp listener. This results in a Unix style of interaction between the system and the user, where the user types in a command and can see the results of that action immediately. All Lisp listeners of the system share the same workspace, which means that they can also share code.

BioBike provides biologists, in principle, with an environment in which they interact with the computer in the same terms as they would interact with their peers; with a uniform framework for accessing knowledge from a number of different knowledge bases; and with a common work area where data and results can be shared and external tools can be integrated. BioBike has been in place over a number years and has demonstrated benefits to collaborating teams of biologists and computer scientists during that time[42].

Design Choices

When making the decision about the platform of the system, another option we had was to implement a similar system from scratch. But when we considered the capabilities

```
(p name ‘‘optional documentation string’’
   buffer tests
  ==>
   buffer changes and requests)
```

Figure 4.1: General form of a production[44]

provided by BioBike (such as a live Lisp listener, a system for integrating third-party tools, support for collaboration, and a robust code base) this made the decision of choosing a platform a trivial one. Further benefits of this decision are related to software engineering concerns. The code for the system is open sourced under the MIT license and is actively maintained. As a result we will be able to download updates to the code and apply those changes in a way we see fit to the system.

4.2.2 Choice of structure for representation

The anatomy of an ACT-R model

An ACT-R model is a representation of a hypothesis of how a specific aspect of human functions. As described in Chapter 2 the ACT-R theory postulates the existence of two types of knowledge namely: declarative knowledge[44], is the type of knowledge that a person is supposed to know when solving a problem and procedural memory, is the knowledge that governs our behavior.

Declarative memory in ACT-R is represented by chunks[44]. Every chunk is composed of slots and values; it is these slots and values that define the characteristics of chunks. Chunk types are entities that define the slots for chunks. They serve the purpose of categorizing chunks[44]. A chunk type is defined using the `chunk-type` macro. Chunks are added to the declarative memory using the `add-dm` macro.

Productions are generally of the form as shown in figure 4.2.2. They consist of condition-action pair. The conditions tests the values in *buffers*, buffers are interface between ACT-R's procedural module and other components, if the buffer tests are satisfied the production applies and the actions in the productions are executed.

Consider an example ACT-R model, this example has been taken from[44], shown in figure 4.2.2. This is a model that simulates how to count. As described earlier we

```
(define-model count
    (chunk-type count-order first second)
    (chunk-type count-from start end count)

    (add-dm
         (b ISA count-order first 1 second 2)
         (c ISA count-order first 2 second 3)
         (d ISA count-order first 3 second 4)
         (e ISA count-order first 4 second 5)
         (f ISA count-order first 5 second 6)
         (first-goal ISA count-from start 2 end 4))

   (p start
      =goal>
       ISA         count-from
       start       =num1
       count       nil
     ==>
      =goal>
       count       =num1
     +retrieval>
       ISA         count-order
       first       =num1)
   .
   .
   .
)
```

Figure 4.2: Example of a simple ACT-R model

define chunk types using the `chunk-type` command. In this model we use two chunk type definitions: a count order chunk type that declares the order in which numbers are meant to appear and a count-from chunk type that helps us define the numbers from and to we are counting. The definition of chunk-type for `count-from` defines the number we are supposed to start counting at indicated by the slot `start`. The number we are supposed to stop counting at is defined by the slot `end`. The slot `count` helps us maintain state in the model by storing the number we are currently at.

The call to `add-dm` adds chunks into the declarative memory. The chunk definitions define the type of chunk and specify slots and their values.

A production called `start` is defined. This production has one condition and two actions separated by `==>`. We deal with two buffers in this production namely: the `goal` buffer, that helps up maintain the state of model and the `retrieval` buffer that helps us retrieve chunks from the declarative memory. In any production test, the start of a condition is identified by the symbol `=` followed by the name of the buffer and ends with the symbol `>`. The condition must always test the type of chunk in the buffer that is done using the `ISA` test. In the tests terms starting with an `=` sign are variables, the scope of these variables extend only to the current production. Anything else in the terms is a hard value against which we test the production.

The actions section the production can define three types of actions that can be performed on the buffers: buffer modification, where we modify the contents of a slot of a chunk in the buffer; buffer retrieval request, where we request the buffer to pick up a chunk from the declarative memory and a buffer clear, where we clear the chunk from a buffer. These operations are indicated by the symbols: `=`, for buffer modification; `+` for buffer retrieval and `-` for clearing a buffer. These symbols are followed by the name of the buffer on which these operations will be performed followed by the `>` symbol. Specific actions or values are defined in the lines the follow this line. These definitions can be seen in figure 4.2.2.

Representation of the model

A structured representation of an ACT-R model is intended to provide an abstraction above the level of modeling code An abstraction should give a number of benefits: the ability to share models more conveniently than by exchanging Lisp code; the potential for

an "ecosystem" of tools for to support ACT-R modeling, in construction and analysis; and the ability to maintain revisions of models[1] and the ability to extend ACT-R models in ways explicitly planned for.

We had to decide between three main choices of structures for representing an ACT-R model: a direct representation of models in the ACT-R modeling language, a semantic network, and frames.

ACT-R models are essentially Lisp data structures. Therefore the direct representation of the model would store the code for the model as is. The benefit of this approach lies in the simplicity of the implementation, which would have involved storing the code as symbols, and the ease with which modeling researchers can understand a familiar representation. But this approach has a number of disadvantages. Building tools for this representation would be exceptionally hard because the tools would have to parse the model before being able to perform any operations. The representation does not allow us to store meta data. It does not allow us to easily add capabilities for reuse of the model. Finally it would be cumbersome to implement a system of revision control for this data structure.

An alternative to the direct representation is using semantic networks to represent ACT-R models. Semantic networks are a notation for representing knowledge, using nodes to represent objects, concepts or situations in a domain, and using arcs to represent relations between them [45, 46]. This would mean representing the internal components of ACT-R models as node: the productions, and the conditions and actions they contain, as well as models themselves. The advantages of using this representation are that tools written for ACT-R could act directly on the nodes representing productions and models; the representation would make it easy to implement facilities for reuse of productions; queries could be constructed to determine the type of objects and their relationships. The deficiencies of this representation are that implementing a system of revision control would be difficult because we would not be able to represent version information in a straightforward way. It would also be difficult to add capabilities of model reuse.

The frame data structure was introduced by Marvin Minsky [5] in 1975, in one

[1]This is a continuing concern in ACT-R modeling research. The ACT-R architecture has gone through several major revisions over the years (the current version is ACT-R 6.0, but modeling researchers rarely return to models based on earlier versions to test whether changes in the implicit architectural dependencies have rendered their past work invalid. Improved modeling abstractions may make this process easier.

the seminal papers in the area of knowledge representation. Frames are structures that can represent objects, situations and concepts. Frames are arranged in a hierarchical taxonomy where every frame is linked with its parent frame. The parent frame represents a more general concept than its child[47]. A frame is made up of a number of slots. Slots define the properties of the object being represented by the frame. Slots are sometimes also used to represent relationships between two frames. This representation provides us with a number of advantages. Unlike semantic networks we have all related data stored in a single frame; frames are dynamic and allow slots and their values to be updated very easily. Since frames are structured, it is easy to write tools to help perform various forms of analysis. Finally, in the future frames might help in analyzing more complex forms of reuse of cognitive models.

In knowledge representation research it is common to find integrations of frames and semantic networks; for example, a semantic network might include frames rather than atomic symbolic concepts in its nodes. The distinctions made in the above discussion are between their "core" properties.

4.3 System Design

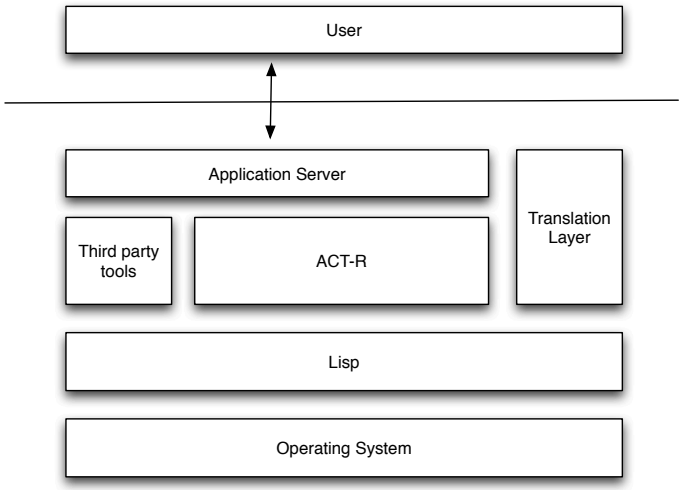

Figure 4.3: System Design

This section aims to explain how various components of the system work together to have the desired effect. Figure 4.3 provides a high level overview of the system. This

section will examine each component in detail. The system currently runs on a server running GNU/Linux. As a result we have a secure, robust, and stable platform on which to base our system.

The User

The user of this system would be a researcher who uses ACT-R for modeling cognitive tasks. He typically would use a text editor to create his models and a graphical user interface provided by ACT-R to run his model. His tasks essentially consist of defining knowledge structures that define the model like chunk-types, chunks and productions that define the cognitive task.

The user interacts with the coglaborate system through a web browser. When a user logs in he is put into the ACT-R package. This is done because it was not possible to export all the required variables and functions into the user package. This is because ACT-R either lets us export all its symbols into to the `cl-user` package or the `act-r` package.

The only thing different about the models being written to run on their own system and those that are run on the Coglaborate system is that the users have to wrap their code with a call to the `with-user-meta-process` macro. This macro creates new meta process for the user and lets them run the model in that ACT-R meta process so as to avoid conflicts with other users running their models in ACT-R.

Application Server

We use the AllegroServe Web Application server to serve the front end that the user sees. The AllegroServe[48] application server provides us with: a HTTP/1.1 compliant web server capable of serving static and dynamic pages; a HTML generation facility that allows us to merge static and dynamic content; secure socket layer for the server and clients; a comprehensive test suite to verify the functionality of the client, server, proxy and SSL. It is licensed under the Lisp Lesser GNU Public License.

ACT-R

When a user evaluates a model it is compiled by the ACT-R model into a thunk. My code is plugged into the parser of the ACT-R compiler so that we can access the data

structure that is generated as the model is parsed. This data structure is then converted into frames. A detailed description of the structure of the frames can be found in the following sections.

Translation Layer

This is a conceptual layer that represents the implementation of the ideas suggested in this project. This layer consists of two main sections: a section that converts ACT-R model data structures into frames and a section that converts frames back into ACT-R models. Since the code in this section is trivial, this section aims to explain the structure of the frames that are created. Every frame has a few elements in common that describe: the type of object that the current frame represents and the name of the object that the frame represents. The structure of all the frames created by the code is described below.

- *The model frame:* A model frame represents an ACT-R model. It consists of a code slot that holds all the code that is required by the model: this includes code for initialization of the model; chunk definitions for the model and miscellaneous utility functions that may be required by the model. It has a slot for productions that are a part of the model.

- *The production frame:* A production frame consists of a conditions slot, which defines the conditions that are required for the condition to fire, and a actions slot which lists all the actions that will be executed if that production is fired.

- *The buffer test frames:* Every production in ACT-R consists of a number of conditions. Each condition consists of a number of tests. Each buffer test frame represents one such test. This frame has a slot to represent individual clauses within the test.

- *The conditions frames:* The conditions frames represent an individual clause it consists of a test field that represents the buffer variable it is testing and a value field that represents a value against which we are testing it. This field can also hold variable as in ACT-R productions.

- *The buffer actions frames:* ACT-R productions can perform a number of actions that can either modify, clear or retrieve a chunk in a buffer. These actions are represented by these frames.

- *The action frames:* Individual clauses that represent the modifications that are required to be made to the buffer are represented by these frames.

The names of the all frames except the name of the frame that represents the model are appended with a random symbol so as to avoid name clashes.

When a user clicks a link representing the frame a function named html-for-browse-frame is called. This function searches for the frame and then converts it to code that is then returned to the browser.

4.4 Work Flow

This section illustrates a scenario where a user creates a model and evaluates it. Another user comes in searches for the model created by the first user, navigates around its productions and obtains the code for it.

A user creates a frame representation by evaluating the code that he types into his lisp listener as shown in figure 4.4. The lisp listener has two text boxes, the large text area is used to type in complete models and the smaller text box is used to type in one off commands.

Once a frame representation of the model has been created any other user can search for the model if he knows the name of the model. Clicking the "Find frames" link does this. The result of the search is shown in figure 4.5.

Figure 4.6 shows the frame representation of what the model that the user was searching for. Once the user finds the model that he was searching for, he can navigate the model by clicking on the links inside the frame table.

Figure 4.7 shows us the frame representation of a production; this is obtained by clicking any link in the slot value column of the model frame for the slot name "productions". The user can investigate a production further by clicking on the links in the slot value section.

If a user wants to obtain code for the model he can click the Frame→Listener link on the index page of the model. Figure 4.8 shows the resulting code generated as a result of clicking the said link.

Figure 4.4: A user creates and evaluates a model

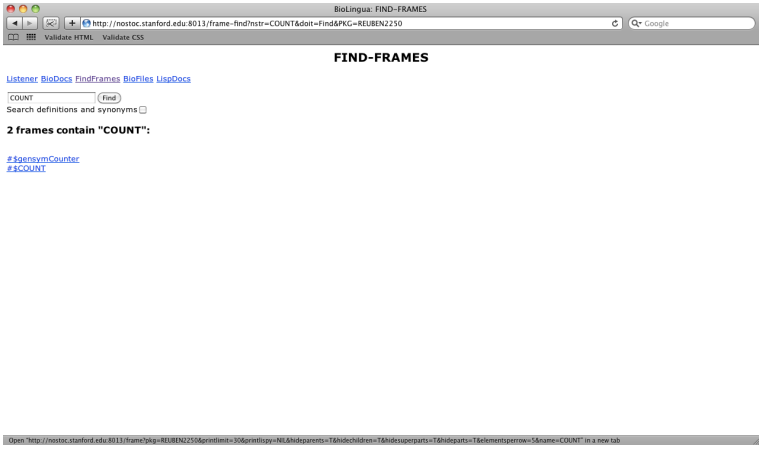

Figure 4.5: Search results after searching for a model

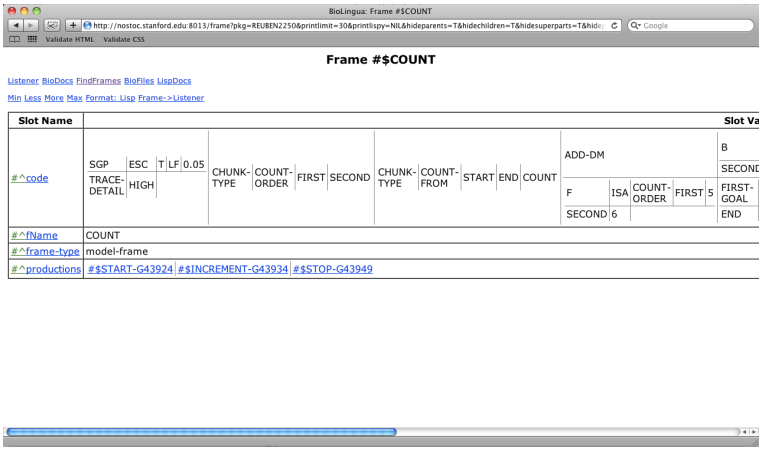

Figure 4.6: Navigating a model represented as a frame

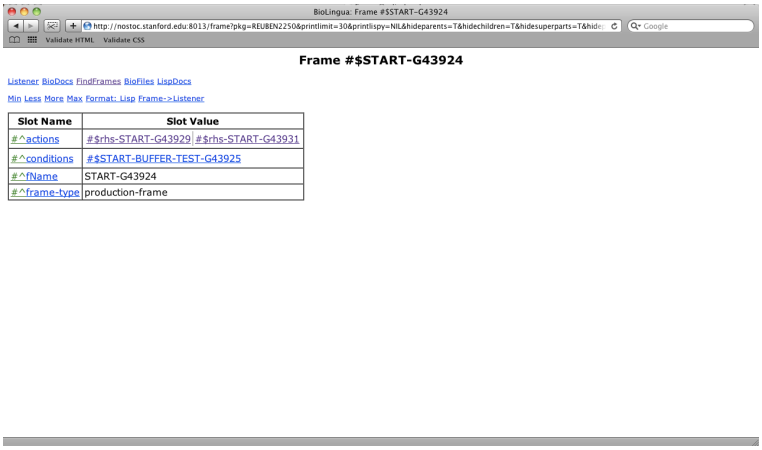

Figure 4.7: Navigating a production represented as a frame

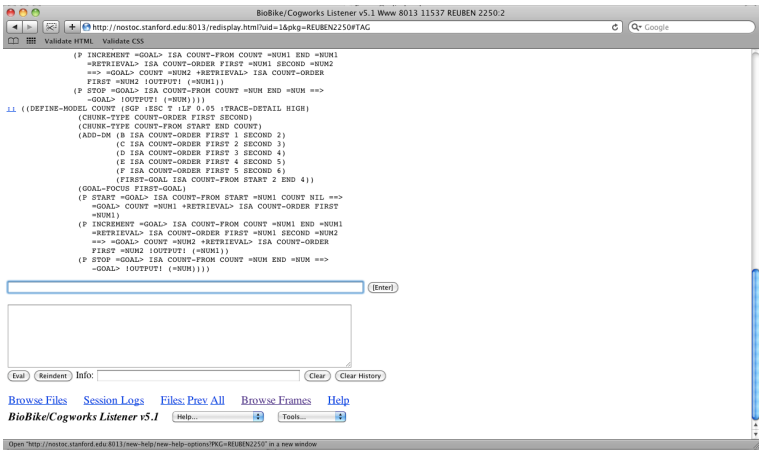

Figure 4.8: The resultant code after clicking the Frame → Listener link

4.5 Proof of concept

To evaluate the capabilities of the system we built a simple but a medium scale models. The essence of this exercise are two fold: firstly we show that the system is capable of performing a non-trivial cognitive modeling exercise and secondly to demonstrate the level of maturity of the system. This section discusses the problem description, the approaches that we have chosen and what we can deduce from this.

4.5.1 Problem description

A crossword is a puzzle in which words or phrases are entered into an interlocking grid either horizontally or vertically. The words are determined by a series of clues that define the word or the phrase. Two or more words are synonyms if they can be exchanged in the same context while maintaining the meaning of the context. Our proof of concept problem is a crossword puzzle where the clues and the solutions are defined as being synonyms of each other.

This problem is appropriate for the following reasons: it demonstrates that this system is a practical and is ready for the outside world, albeit the code needs little bit of polishing; it demonstrates that users can use this system to write and test sand boxed act-r modules and finally it places considerable demands on the hardware of the computer, in terms of memory and CPU. As a consequence we can demonstrate that we have achieved

most of the objectives we had in mind for the system.

4.5.2 Implementation

The data that helps us determine the solution to our clues is provided by WordNet[49]. Miller defines WordNet in the following terms:

> WordNet is an online lexical database designed for use under program control. English nouns, verbs, adjectives, and adverbs are organized into sets of synonyms, each representing a lexicalized concept. Semantic relations link the synonym sets.

We use a module(WNLexical[50]) that enables ACT-R to make use of the WordNet lexical database. The module does so by converting the WordNet database into set of s-expressions that can be read in by the WNLexical module and stored as ACT-R chunks. The WNLexical module exposes its functionality through the means of an ACT-R buffer, aptly named `wn-lexical`. This buffer can query the WordNet database for synonyms, antonyms, hyponyms, meronyms, glossaries of words etc. The results of these queries are available from accessing the slots of the buffer.

The data structure for the clues is exceedingly simple. The data structure is a list of clues(which are represented as chunks), where every clue is a list that consists of the starting co-ordinates of the word, the direction(either across or down), the clue string, a location to put in a solution and the actual solution itself. The data structure is manipulated using the `Crossword` module. The crossword module has the ability: to set words in the specified location; to verify if the crossword still maintains constraints and to query various parameters of a specific clue.

When the model(the source code of the model is available in appendix A) is evaluated it initializes three chunk definitions: a chunk type to maintain the state of the goal buffer initially; a chunk type that defines the clue and a chunk type that maintains the state of the goal buffer as we actually get into the process of solving the crossword. It sets up a few parameters so as to prevent the decay of crossword chunks in the memory. We then check the memory to see if we can find any clues that have not been added to the data structure. If we find a chunk that has not been added to the crossword we add it in. Once added in we update the goal to find all the synsets of the word. This is done by querying the WordNet module. For every synset found we create a chunk using the `imaginal` module. If

we cannot find the word we display an error message. For every chunk that we had created earlier, we query WordNet to obtain all the words in the synset if the word we have found matches the solution we set it in the clue and mark the clue as solved. This process repeats till all the clues have been solved or have been marked as being unsolvable.

The output trace of this model is available in Appendix B. It shows the trace for two words: one where we are able to solve the clue the other being unsolvable. The trace clearly shows the working of the algorithm in explicit detail.

4.5.3 Conclusions

The proof of concept demonstrates that we have a system that can be used in the real world. We can assert this claim with some confidence because we have been able to write and run a medium-scale model on the system.

We can further assert that we have met most of the objectives that we had set for ourselves through this model. We set up an environment where any other researcher can work with the WordNet module, we also wrote up a module that was required for the data structure.

We can demonstrate the capability of resource sharing. Running the model on my personal computer brought my computer to a crawl because the WordNet module loads all the chunks into memory, whereas the model runs fine when executed on the server.

We have demonstrated that we can share software resources. As a result we gain a saving in the amount of effort spent. For example, it took me a while to figure out that there was a bug in the version of the WN-Lexical module, and I had to upgrade the module. Now consider the amount of effort wasted by numerous other researchers working with the same module. With a system like Coglaborate we would require to expend that amount of effort only once and the entire community would be benefited.

Although the model did not make use of any third party tools the system has a regular expression library and a package management, Another System Definition Facility(ASDF), system integrated into it.

The project added about one thousand lines of code to the existing code base of ACT-R and BioBike. That figure does not include the lines of code of the model. The challenges faced during the development of the project involved

- Deciding on the structure of the frames so that we have the right level of granularity and at the same time we don't lose any information related to the productions;
- Developing an interface and workflow consistent with existing practices within the cognitive modeling community;
- Figuring out a way to support collaboration using frames;
- Analyzing the ACT-R compiler to identify locations where the translation layer would hook in;
- Identifying and fixing two existing bugs in the system.

The model consists of sixteen productions with a total of about four hundred and sixty lines of code.

An evaluation of the collaborative aspects of the use of Coglaborate are unfortunately beyond the scope of this thesis. Evidence for this capability is provided by experience with BioBike; some similarities in the benefits can be expected. Nevertheless an illustration of the use of Coglaborate to solve a medium-sized problem should give us insight into some of the strengths and weaknesses of the approach.

Chapter 5

Conclusion

This project began with very ambitious goals, but in the end most of the original objectives were achieved. This chapter discusses the utility of the project, the challenges that were faced, and a way to move forward on this work.

5.1 Discussion

Readers may wonder about the utility of this undertaking. We have some confidence that this project will add value to the cognitive modeling community. It provides a central location where students and researchers can learn from one another; it improves support and resources available for ambitious cognitive modeling projects that in some cases might not be possible otherwise, due to constraints on hardware or software infrastructure; finally it has the potential to foster a stronger culture of collaboration among community members.

5.1.1 Challenges and future work

Coglaborate has limitations, both from an engineering and a conceptual viewpoint.

To begin with the engineering limitations, there exists a graphical user interface to ACT-R, which we have mentioned only briefly. Coglaborate, in its current state, does not provide all of the same functionality of this interface. For example, Coglaborate does not enable the user to step through productions or view the contents of buffers as a model is being executed. The existing graphical user interface may thus make it easier to build and

debug models. However, Coglaborate offers compensating advantages, and it could easily be extended to incorporate existing development aids.

Like BioBike, Coglaborate provides persistence that is only apparent; objects persist over the lifetime of the active Lisp environment. The practical limitation here is the lack of standardization between database systems that support persistence and Lisp environments in common use. Recently Allegro has introduced an object persistence library called Allegro Cache as part of its platform; adding persistence in the future should be a simple matter of programming.

The current version of the system uses an abstraction of ACT-R called meta processes to provide isolation between models, although as of now it serves its purpose, true isolation could be achieved by permitting the application server let every user spawn a new Lisp process if required. This facility would make the system more robust and might help if load balancing is needed in the future.

There are two main conceptual limitations to the work. First, Coglaborate relies on BioBike as a platform to support collaboration, but there are many other possible approaches that have not been explored in this research. For example, because ACT-R models are software, some level of collaboration could be supported by systems for collaborative software development and management, beginning with something as simple as a configuration management tool. We believe that BioBike and Coglaborate gain power from the KnowOS concept, but this project has not directly explored the relative advantages compared with more conventional techniques in software engineering. Second, Coglaborate's support for collaboration, in the current state of its implementation, operates at a relatively granular level. Models, tools, and architecture extensions can be added to the system by individual users, so that they can be more easily used by others. There is no support for multi-user model development or integration of multiple smaller models into larger ones, however. Even though we are confident that analysis tools (made possible by Coglaborate's model representation) could contribute to such collaboration, these tools remain to be written.

Other limitations remain to be discovered. We believe that the system can best be tested for effectiveness by opening it up the community, obtaining the feedback and improving it. This way we can test the effectiveness of collaboration in cognitive modeling in the real world.

5.1.2 Implications of this work

On creating a frame-based abstraction for ACT-R models it quickly became clear that this representation could be used to explore a number of other possibilities. This section discusses these possibilities.

As observed by Langley et al. [31], an important issue facing cognitive modeling is support for software reuse. This project promotes reuse of models in the sense that the representation allows for models to be represented, analyzed, and distributed in a much more transparent fashion than in their current representation as Lisp code. Today, it is impossible to determine the similarity between two ACT-R models except through code inspection and ad hoc judgments. The frame-based representation introduced in this research makes more sophisticated analysis possible: comparison of the use of buffers across productions, for example. Such analyses remain for future work.

Another interesting research direction would be to investigate software reuse as provided by object-oriented programming environments. That is, we can develop features such that models can inherit behavior from other more general models. This way we should be able identify general patterns that emerge from human cognition.

Yet another area of interest for this system is the investigation of user interfaces that allow cognitive scientists to create models without actually having to learn Lisp. This could lead to a gentler learning curve for people working with ACT-R.

Bibliography

[1] D.E. Rumelhart, J.L. McClelland, and PDP CORPORATE. *Research Group, Parallel distributed processing: explorations in the microstructure of cognition, vol. 1: foundations*. MIT Press, Cambridge, MA, 1986.

[2] D.E. Rumelhart, J.L. McClelland, and PDP CORPORATE. Research Group, Parallel distributed processing: explorations in the microstructure of cognition, vol. 2: psychological and biological models, 1986.

[3] Allen Newell. *Unified Theories Of Cognition*. Havard University Press, 1990.

[4] J.C.R. Licklider and R.W. Taylor. The computer as a communication device. *Science and technology*, 76:21–31, 1968.

[5] Marvin Minsky. A framework for representing knowledge. Technical report, Cambridge, MA, USA, 1974.

[6] Radu J Bogdan. History of cognitive science. *Dictionnaire Critique de la Communication*, pages 870–878, 1993.

[7] David Vernon, Giorgio Metta, and Giulio Sandini. A survey of artificial cognitive systems: Implications for the autonomous development of mental capabilities in computational agents. *IEEE Trans. Evolutionary Computation*, 11(2):151–180, 2007.

[8] Ronald J. Brachman and Hector Levesque. *Knowledge Representation and Reasoning*. Elsevier, Amsterdam, 2004.

[9] Allen Newell. Physical symbol systems. *Cognitive Science*, 4(2):135 – 183, 1980.

[10] John R. Anderson and Christian Lebiere. The newell test for a theory of cognition. *Behavioral and Brain Sciences*, 26(05):587–601, 2003.

[11] Jonathan D. Cohen and Jonathan W. Schooler, editors. *Scientific Approaches to Consciousness*. Lawrence Erlbaum, 1996.

[12] Neil A. Stillings, Mark H. Feinstein, Jay L. Garfield, Edwina L. Rissland, David A. Rosenbaum, Steven E. Weisler, and Lynne Baker-Ward. *Cognitive science: an introduction*. MIT Press, Cambridge, MA, USA, 1987.

[13] David A. Medler. A brief history of connectionism. *Neural Computing Surveys*, 1:61–101, 1998.

[14] A. Newell. *You Can't Play Twenty Questions With Nature and Win*, pages 283–308. Academic Press., New York, 1973.

[15] Ron Sun. The importance of cognitive architectures: an analysis based on CLARION. *J. Exp. Theor. Artif. Intell*, 19(2):159–193, 2007.

[16] John R. Anderson, Daniel Bothell, and Michael D. Byrne. An integrated theory of the mind. *Psychological Review*, 111(4):1036–1060, 2004.

[17] J. Anderson and G. Bower. *Human Associative Memory*. Winston, Washington D.C., 1973.

[18] J. R. Anderson. *Language, Memory and Thought*. Erlbaum, Hillsdale, N. J., 1976.

[19] John Anderson. *The Architecture of Cognition*. Harvard University Press, Cambridge, MA, 1983.

[20] Niels A. Taatgen and John R. Anderson. Why do children learn to say "broke"?, February 07 2002.

[21] D.E. Kieras and D.E. Meyer. An overview of the EPIC architecture for cognition and performance with application to human-computer interaction. *Human-Computer Interaction*, 12(4):391–438, 1997.

[22] John E. Laird, Allen Newell, and Paul S. Rosenbloom. Soar: an architecture for general intelligence. *Artif. Intell.*, 33(1):1–64, 1987.

[23] R. L. Lewis. Cognitive theory, soar. *International Encylopedia of the Social and Behavioral Sciences*, 2001.

[24] John E. Laird and Clare Bates Congdon. The soar user's manual. Manual, University of Michigan, 2006.

[25] John Laird Jill Fain Lehman and Paul Rosenbloom. A gentle introduction to soar, an architecture for human cognition. Architecture update, University of Michigan, 2006.

[26] David E. Kieras, David E. Kieras, David E. Meyer, and David E. Meyer. An overview of the epic architecture for cognition and performance with application to human-computer interaction. *Human-Computer Interaction*, 12:391–438, 1997.

[27] Ron Sun, Edward Merrill, and Todd Peterson. From implicit skills to explicit knowledge: a bottom-up model of skill learning. *Cognitive Science*, 25(2):203–244, 2001.

[28] Ron Sun. A tutorial on clarion 5.0. Technical report, Rensselaer Polytechnic Institute, 2003.

[29] Richard W. Pew and Anne S. Mavo. *Modeling human and organizational behavior: application to military simulations*. National Academy Press, 1998.

[30] Hui-Qing Chong, Ah-Hwee Tan, and Gee-Wah Ng. Integrated cognitive architectures: a survey. *Artificial Intelligence Review*, 28(2):103–130, 2007.

[31] Pat Langley, John E. Laird, and Seth Rogers. Cognitive architectures: Research issues and challenges. *Cognitive Systems Research*, 10(2):141–160, June 2009.

[32] Randy L. Buckner and Deanna Barch. Cognition: Memory, 1: Episodic Memory Retrieval. *Am J Psychiatry*, 156(9):1311–, 1999.

[33] Daniele Nardi and Ronald J. Brachman. An introduction to description logics. In Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter Patel-Schneider, editors, *The Description Logic Handbook: Theory, Implementation and Applications*, pages 1–40. Cambridge University Press, Cambridge, England, 2003.

[34] Tom Rodden. A survey of CSCW systems. *Interacting with Computers*, 3(3):319–353, 1991.

[35] M. Stefik, G. Foster, D. G. Bobrow, K. Kahn, S. Lanning, and L. Suchman. Beyond the chalkboard: Using computers to support collaboration and problem solving in meetings. *Communications of the ACM*, 30:32–47, 1987.

[36] John Bowers and John Churcher. Local and global structuring of computer mediated communication: Developing linguistic perspectives on CSCW in cosmos. In *CSCW*, pages 125–139, 1988.

[37] T. Malone, K. Grant, F. Turbak, S. Brobst, and M. Cohen. Intelligent information-sharing systems. *Communications of the ACM*, 30(5), May 1987.

[38] Starr Roxanne Hiltz and Murry Turoff. *The Network Nation: Human Communication via Computer*. Addison-Weseley, 1978.

[39] S. Sarin and I. Greif. Computer-based real-time conferencing systems. *IEEE Computer*, 18(10):33–45, 1985.

[40] Kraemer and King. Computer-based systems for cooperative work and group decision making. *CSURV: Computing Surveys*, 20, 1988.

[41] G. DeSanctis and B. R. Gallupe. Group decision support systems: A new frontier. *Data Base*, pages 3–10, Winter 1985.

[42] J. P. Massar, Michael Travers, Jeff Elhai, and Jeff Shrager. Biolingua: a programmable knowledge environment for biologists. *Bioinformatics*, 21(2):199–207, 2005.

[43] Mike Travers, Jp Massar, and Jeff Shrager. KnowOS: The (re)birth of the knowledge operating system, February 07 2008.

[44] Act-r 6.0 tutorial.

[45] J. F. Sowa. Semantic networks. In S. C. Shapiro, editor, *Encyclopedia of Artificial Intelligence*, New York, 1987. Wiley.

[46] A. Barr and E. A. Feigenbaum. *The Handbook of Artificial Intelligence*. William Kaufman, Inc., Los Altos, CA, 1981.

[47] P. D. Karp. The design space of frame knowledge representation systems. Technical Note 520, SRI International, Menlo Park (CA), USA, 1993.

[48] Allegroserve - a web application server, 10 2009.

[49] George A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, 1995.

[50] B. Emond. WN-LEXICAL: An ACT-R module built from the WordNet lexical database. In *Proceedings of the Seventh International Conference on Cognitive Modeling*, 2006.

Appendices

Appendix A

Appendix

A.1 ACT-R model

```
(clear-all)

(define-model test-model

     (defun my-string-not-equal(a b)
       (not (string= a b)))

  (sgp :wnl-chunks wnl :imaginal-delay 0
       :bll 0 :er nil :esc nil
       :declarative-finst-span 1000 :declarative-num-finsts 1000
       :do-not-harvest wn-lexical )

  (chunk-type goal state number)
  (chunk-type clue clue-id hint solution model-solution position state type )
  (chunk-type syn-goal word-str synset-id state clue-id solution model-solution)

  (add-dm
   ;;Module chunks
   (in-crossword isa chunk)
   ;;model chunks
   (g1 isa goal state find-clue number 1)
   (solve-puzzle isa chunk)
   (fill-horizontal-clues isa chunk)
   (number-of-horizontal-solutions isa chunk)

   (find-word-definition isa chunk)
```

```
  (find-word-gloss isa chunk)
  (find-synonyms isa chunk)
  (say-definition isa chunk)

  (clue-1 isa clue clue-id 1 hint "gold"
          solution "gilded" position '(0 . 1)
          state not-in-crossword type cross)
  (clue-4 isa clue clue-id 4 hint "ljad;"
          solution "vomit" position '(0 . 1)
          state not-in-crossword type cross))

(p find-clue
    =goal>
    isa goal
    state find-clue

    ?crossword>
    state insertion-not-complete

    ==>

    +retrieval>
    isa clue
    state not-in-crossword

    !output! (picking up a new clue)

    +goal>
    isa syn-goal
    state add-word
    )

 (p update-goal-to-get-synonyms

    =retrieval>
    isa clue
    state not-in-crossword
    hint =word
    clue-id =id
    solution =soln

    =goal>
    isa syn-goal
```

```
   state add-word

   ?wn-lexical>
   state free

   ==>

   =goal>
   state find-word-definition
   word-str =word
   clue-id =id
   solution =soln

   +wn-lexical>
   isa wnl-clear-context


   =retrieval>
   state in-crossword)

(p find-word-definition
   =goal>
   isa syn-goal
   state find-word-definition
   word-str =word

   ?wn-lexical>
   state free

   ?wn-lexical>
   state success

   ==>

   +wn-lexical>
   isa wnl-request
   wn-operator s
   word =word
   context-criterion set-difference

   !output! (Looking for sense of =word)

   =goal>
```

```
   state find-word-gloss
   )

(p find-word-gloss
   =goal>
   isa syn-goal
   state find-word-gloss
   word-str =word

   ?wn-lexical>
   state free

   ?wn-lexical>
   state success

   =wn-lexical>
   isa s
   word =word
   synset-id =synset-id

   ==>

   +wn-lexical>
   isa wnl-request
   wn-operator g
   synset-id =synset-id

   !output! (Looking for definition of =word)

   =goal>
   state say-definition
   )

(p say-definition
   =goal>
   isa syn-goal
   state say-definition
   word-str =word

   ?wn-lexical>
   state free

   ?wn-lexical>
```

```
   state success

   =wn-lexical>
   isa g
   gloss =gloss
   synset-id =synset-id

   ==>

   !output! (the definition of =word is =gloss)
   =goal>
   state find-synonyms
   synset-id =synset-id

   +wn-lexical>
   isa wnl-request
   wn-operator s
   synset-id =synset-id

   )


(p find-synonyms
   =goal>
   isa syn-goal
   state find-synonyms
   synset-id =synset-id

   ?wn-lexical>
   state free

   ?wn-lexical>
   state success

   =wn-lexical>
   isa s
   word =word
   synset-id =synset-id

   ==>

   !output! (=word is a synonyms)
```

```
   +imaginal>
   isa syn-goal
   state not-examined
   synset-id =synset-id
   word-str =word

   =goal>
   state find-word-definition
   )

(p do-not-know-word
   =goal>
   isa syn-goal
   state find-word-gloss
   word-str =word

   ?wn-lexical>
   state free

   ?wn-lexical>
   state error

   ?retrieval>
   state free

   ==>

   !output! (I know nothing about =word)

   +retrieval>
   isa syn-goal
   state not-examined
   :recently-retrieved nil

   =goal>
   state not-examined

   -imaginal>

   -wn-lexical>
   )

(p start-exploring-word
```

```
   =goal>
   isa syn-goal
   state not-examined


   =retrieval>
   isa syn-goal
   state not-examined
   synset-id =synset-id

   ?wn-lexical>
   state free

   ==>

   =retrieval>
   state examined

   =goal>
   state explore-word
   synset-id =synset-id

   +wn-lexical>
   isa wnl-clear-context)

(p explore-word

   =goal>
   isa syn-goal
   state explore-word
   synset-id =synset-id

   ?wn-lexical>
   state free

   ==>

   +wn-lexical>
   isa wnl-request
   wn-operator s
   synset-id =synset-id
   context-criterion set-difference
```

```
   =goal>
   state say-word)

(p no-more-words-in-this-sense

   =goal>
   isa syn-goal
   state say-word
   clue-id =id

   ?wn-lexical>
   state error

   ?wn-lexical>
   state free

   ?retrieval>
   state free
   ==>

   ;;use retrieval pick next word
   +retrieval>
   isa syn-goal
   state not-examined
   :recently-retrieved nil

   !output! (picking up next word)


   =goal>
   state examined)

(p temp-prod

   =goal>
   isa syn-goal
   state examined

   ==>

   =goal>
   state not-examined)
```

```
(p explored-all-senses

   =goal>
   isa syn-goal
   state not-examined
   clue-id =id

   ?retrieval>
   state error

   ?wn-lexical>
   state free
   ==>

   +goal>
   isa syn-goal
   state add-word

   !output! (could not solve clue =id)

   +retrieval>
   isa clue
   state not-in-crossword
   :recently-retrieved nil

   -wn-lexical>

   !output! (explored all senses))



(p say-word

   =wn-lexical>
   isa s
   word =word
   synset-id =synset-id

   =goal>
   isa syn-goal
   state say-word
   solution =soln
```

```
   clue-id =id

   ?retrieval>
   state free

   !safe-eval! (string-equal =word =soln)

   ==>

   !output! (selected word =word)
   !output! (solution =soln)

   =goal>
   state update-clue
   model-solution =soln

   +retrieval>
   isa clue
   clue-id =id
   state in-crossword
   :recently-retrieved nil
   )

(p update-clue

   =goal>
   isa syn-goal
   state update-clue
   model-solution =soln
   clue-id =id

   =retrieval>
   isa clue
   clue-id =id
   ==>

   =retrieval>
   model-solution =soln
   state solved

   !output! (solved clue =id)

   =goal>
```

```
   state clear-buffer-and-pick-up-next-clue

   )

(p clear-buffer-and-pick-up-next-clue

   =goal>
   isa syn-goal
   state clear-buffer-and-pick-up-next-clue

   ?retrieval>
   state free

   ?wn-lexical>
   state free

   ==>

   +goal>
   isa syn-goal
   state add-word

   +retrieval>
   isa clue
   state not-in-crossword
   :recently-retrieved nil

   -wn-lexical>
   )

(p say-word-1

   =wn-lexical>
   isa s
   word =word
   synset-id =synset-id

   =goal>
   isa syn-goal
   solution =soln
   state say-word

   !safe-eval! (my-string-not-equal =word =soln)
```

```
    ==>

    !output! (word =word)

    =goal>
    state explore-word)
  )

(goal-focus g1)
(run 40)
```

A.2 Sample run of the model

I have provided a sample run of the act-r synonym crossword model. In this run it solves a clue with the hint "gold" and identifies that a clue with the hint "ljad" cannot be solved.

```
     0.000   GOAL                    SET-BUFFER-CHUNK GOAL G1 REQUESTED NIL
     0.000   PROCEDURAL              CONFLICT-RESOLUTION
     0.050   PROCEDURAL              PRODUCTION-FIRED FIND-CLUE
PICKING UP A NEW CLUE
     0.050   PROCEDURAL              CLEAR-BUFFER RETRIEVAL
     0.050   PROCEDURAL              CLEAR-BUFFER GOAL
     0.050   GOAL                    SET-BUFFER-CHUNK GOAL SYN-GOAL0
     0.050   DECLARATIVE             START-RETRIEVAL
     0.050   DECLARATIVE             RETRIEVED-CHUNK CLUE-1
     0.050   DECLARATIVE             SET-BUFFER-CHUNK RETRIEVAL CLUE-1
     0.050   PROCEDURAL              CONFLICT-RESOLUTION
     0.100   PROCEDURAL              PRODUCTION-FIRED
UPDATE-GOAL-TO-GET-SYNONYMS
     0.100   PROCEDURAL              CLEAR-BUFFER WN-LEXICAL
     0.100   WN-LEXICAL              CLEAR-WN-LEXICAL-SYNSET-CONTEXT
     0.100   PROCEDURAL              CONFLICT-RESOLUTION
     0.150   PROCEDURAL              PRODUCTION-FIRED FIND-WORD-DEFINITION
LOOKING FOR SENSE OF "gold"
     0.150   PROCEDURAL              CLEAR-BUFFER WN-LEXICAL
     0.150   WN-LEXICAL              RETRIEVE-WN-CHUNKS
     0.150   WN-LEXICAL              SELECT-FROM-RETRIEVED-WN-CHUNKS SET OF
SIZE 7
     0.150   WN-LEXICAL              SET-BUFFER-CHUNK WN-LEXICAL
S-300369941-4
     0.150   PROCEDURAL              CONFLICT-RESOLUTION
     0.200   PROCEDURAL              PRODUCTION-FIRED FIND-WORD-GLOSS
LOOKING FOR DEFINITION OF "gold"
     0.200   PROCEDURAL              CLEAR-BUFFER WN-LEXICAL
     0.200   WN-LEXICAL              RETRIEVE-WN-CHUNKS
     0.200   WN-LEXICAL              SELECT-FROM-RETRIEVED-WN-CHUNKS SET OF
SIZE 1
     0.200   WN-LEXICAL              SET-BUFFER-CHUNK WN-LEXICAL
G-300369941-1
     0.200   PROCEDURAL              CONFLICT-RESOLUTION
     0.250   PROCEDURAL              PRODUCTION-FIRED SAY-DEFINITION
THE DEFINITION OF "gold" IS "having the deep slightly
      brownish color of gold; \"long aureate (or golden) hair\";
```

```
     \"a gold carpet\""
     0.250   PROCEDURAL            CLEAR-BUFFER WN-LEXICAL
     0.250   WN-LEXICAL            RETRIEVE-WN-CHUNKS
     0.250   WN-LEXICAL            SELECT-FROM-RETRIEVED-WN-CHUNKS SET OF
SIZE 5
     0.250   WN-LEXICAL            SET-BUFFER-CHUNK WN-LEXICAL
S-300369941-5
     0.250   PROCEDURAL            CONFLICT-RESOLUTION
     0.300   PROCEDURAL            PRODUCTION-FIRED FIND-SYNONYMS
"golden" IS A SYNONYMS
     0.300   PROCEDURAL            CLEAR-BUFFER IMAGINAL
     0.300   IMAGINAL              SET-BUFFER-CHUNK IMAGINAL SYN-GOAL1
     0.300   PROCEDURAL            CONFLICT-RESOLUTION
     0.350   PROCEDURAL            PRODUCTION-FIRED FIND-WORD-DEFINITION
LOOKING FOR SENSE OF "gold"
     0.350   PROCEDURAL            CLEAR-BUFFER WN-LEXICAL
     0.350   WN-LEXICAL            RETRIEVE-WN-CHUNKS
     0.350   WN-LEXICAL            SELECT-FROM-RETRIEVED-WN-CHUNKS SET OF
SIZE 7
     0.350   WN-LEXICAL            SET-BUFFER-CHUNK WN-LEXICAL
S-114638799-1
     0.350   PROCEDURAL            CONFLICT-RESOLUTION
     0.400   PROCEDURAL            PRODUCTION-FIRED FIND-WORD-GLOSS
LOOKING FOR DEFINITION OF "gold"
     0.400   PROCEDURAL            CLEAR-BUFFER WN-LEXICAL
     0.400   WN-LEXICAL            RETRIEVE-WN-CHUNKS
     0.400   WN-LEXICAL            SELECT-FROM-RETRIEVED-WN-CHUNKS SET OF
SIZE 1
     0.400   WN-LEXICAL            SET-BUFFER-CHUNK WN-LEXICAL
G-114638799-1
     0.400   PROCEDURAL            CONFLICT-RESOLUTION
     0.450   PROCEDURAL            PRODUCTION-FIRED SAY-DEFINITION
THE DEFINITION OF "gold" IS "a soft yellow malleable ductile
 (trivalent and univalent) metallic element; occurs mainly as
  nuggets in rocks and alluvial deposits; does not react with
  most chemicals but is attacked by chlorine and aqua regia"
     0.450   PROCEDURAL            CLEAR-BUFFER WN-LEXICAL
     0.450   WN-LEXICAL            RETRIEVE-WN-CHUNKS
     0.450   WN-LEXICAL            SELECT-FROM-RETRIEVED-WN-CHUNKS SET OF
SIZE 3
     0.450   WN-LEXICAL            SET-BUFFER-CHUNK WN-LEXICAL
S-114638799-2
     0.450   PROCEDURAL            CONFLICT-RESOLUTION
```

```
     0.500   PROCEDURAL            PRODUCTION-FIRED FIND-SYNONYMS
"Au" IS A SYNONYMS
     0.500   PROCEDURAL            CLEAR-BUFFER IMAGINAL
     0.500   IMAGINAL              SET-BUFFER-CHUNK IMAGINAL SYN-GOAL2
     0.500   PROCEDURAL            CONFLICT-RESOLUTION
     0.550   PROCEDURAL            PRODUCTION-FIRED FIND-WORD-DEFINITION
LOOKING FOR SENSE OF "gold"
     0.550   PROCEDURAL            CLEAR-BUFFER WN-LEXICAL
     0.550   WN-LEXICAL            RETRIEVE-WN-CHUNKS
     0.550   WN-LEXICAL            SELECT-FROM-RETRIEVED-WN-CHUNKS SET OF
SIZE 7
     0.550   WN-LEXICAL            SET-BUFFER-CHUNK WN-LEXICAL
S-113371760-1
     0.550   PROCEDURAL            CONFLICT-RESOLUTION
     0.600   PROCEDURAL            PRODUCTION-FIRED FIND-WORD-GLOSS
LOOKING FOR DEFINITION OF "gold"
     0.600   PROCEDURAL            CLEAR-BUFFER WN-LEXICAL
     0.600   WN-LEXICAL            RETRIEVE-WN-CHUNKS
     0.600   WN-LEXICAL            SELECT-FROM-RETRIEVED-WN-CHUNKS SET OF
SIZE 1
     0.600   WN-LEXICAL            SET-BUFFER-CHUNK WN-LEXICAL
G-113371760-1
     0.600   PROCEDURAL            CONFLICT-RESOLUTION
     0.650   PROCEDURAL            PRODUCTION-FIRED SAY-DEFINITION
THE DEFINITION OF "gold" IS "coins made of gold"
     0.650   PROCEDURAL            CLEAR-BUFFER WN-LEXICAL
     0.650   WN-LEXICAL            RETRIEVE-WN-CHUNKS
     0.650   WN-LEXICAL            SELECT-FROM-RETRIEVED-WN-CHUNKS SET OF
SIZE 1
     0.650   WN-LEXICAL            SET-BUFFER-CHUNK WN-LEXICAL
S-113371760-1
     0.650   PROCEDURAL            CONFLICT-RESOLUTION
     0.700   PROCEDURAL            PRODUCTION-FIRED FIND-SYNONYMS
"gold" IS A SYNONYMS
     0.700   PROCEDURAL            CLEAR-BUFFER IMAGINAL
     0.700   IMAGINAL              SET-BUFFER-CHUNK IMAGINAL SYN-GOAL3
     0.700   PROCEDURAL            CONFLICT-RESOLUTION
     0.750   PROCEDURAL            PRODUCTION-FIRED FIND-WORD-DEFINITION
LOOKING FOR SENSE OF "gold"
     0.750   PROCEDURAL            CLEAR-BUFFER WN-LEXICAL
     0.750   WN-LEXICAL            RETRIEVE-WN-CHUNKS
     0.750   WN-LEXICAL            SELECT-FROM-RETRIEVED-WN-CHUNKS SET OF
SIZE 7
```

```
     0.750   WN-LEXICAL             SET-BUFFER-CHUNK WN-LEXICAL
S-113353446-1
     0.750   PROCEDURAL             CONFLICT-RESOLUTION
     0.800   PROCEDURAL             PRODUCTION-FIRED FIND-WORD-GLOSS
LOOKING FOR DEFINITION OF "gold"
     0.800   PROCEDURAL             CLEAR-BUFFER WN-LEXICAL
     0.800   WN-LEXICAL             RETRIEVE-WN-CHUNKS
     0.800   WN-LEXICAL             SELECT-FROM-RETRIEVED-WN-CHUNKS SET OF
SIZE 1
     0.800   WN-LEXICAL             SET-BUFFER-CHUNK WN-LEXICAL
G-113353446-1
     0.800   PROCEDURAL             CONFLICT-RESOLUTION
     0.850   PROCEDURAL             PRODUCTION-FIRED SAY-DEFINITION
THE DEFINITION OF "gold" IS "great wealth;
 \"Whilst that for which all virtue now is sold, and almost
  every vice--almighty gold\"--Ben Jonson"
     0.850   PROCEDURAL             CLEAR-BUFFER WN-LEXICAL
     0.850   WN-LEXICAL             RETRIEVE-WN-CHUNKS
     0.850   WN-LEXICAL             SELECT-FROM-RETRIEVED-WN-CHUNKS SET OF
SIZE 1
     0.850   WN-LEXICAL             SET-BUFFER-CHUNK WN-LEXICAL
S-113353446-1
     0.850   PROCEDURAL             CONFLICT-RESOLUTION
     0.900   PROCEDURAL             PRODUCTION-FIRED FIND-SYNONYMS
"gold" IS A SYNONYMS
     0.900   PROCEDURAL             CLEAR-BUFFER IMAGINAL
     0.900   IMAGINAL               SET-BUFFER-CHUNK IMAGINAL SYN-GOAL4
     0.900   PROCEDURAL             CONFLICT-RESOLUTION
     0.950   PROCEDURAL             PRODUCTION-FIRED FIND-WORD-DEFINITION
LOOKING FOR SENSE OF "gold"
     0.950   PROCEDURAL             CLEAR-BUFFER WN-LEXICAL
     0.950   WN-LEXICAL             RETRIEVE-WN-CHUNKS
     0.950   WN-LEXICAL             SELECT-FROM-RETRIEVED-WN-CHUNKS SET OF
SIZE 7
     0.950   WN-LEXICAL             SET-BUFFER-CHUNK WN-LEXICAL
S-104966240-2
     0.950   PROCEDURAL             CONFLICT-RESOLUTION
     1.000   PROCEDURAL             PRODUCTION-FIRED FIND-WORD-GLOSS
LOOKING FOR DEFINITION OF "gold"
     1.000   PROCEDURAL             CLEAR-BUFFER WN-LEXICAL
     1.000   WN-LEXICAL             RETRIEVE-WN-CHUNKS
     1.000   WN-LEXICAL             SELECT-FROM-RETRIEVED-WN-CHUNKS SET OF
SIZE 1
```

```
     1.000   WN-LEXICAL            SET-BUFFER-CHUNK WN-LEXICAL
G-104966240-1
     1.000   PROCEDURAL            CONFLICT-RESOLUTION
     1.050   PROCEDURAL            PRODUCTION-FIRED SAY-DEFINITION
THE DEFINITION OF "gold" IS "a deep yellow color;
     \"an amber light illuminated the room\";
     \"he admired the gold of her hair\""
     1.050   PROCEDURAL            CLEAR-BUFFER WN-LEXICAL
     1.050   WN-LEXICAL            RETRIEVE-WN-CHUNKS
     1.050   WN-LEXICAL            SELECT-FROM-RETRIEVED-WN-CHUNKS SET OF
SIZE 2
     1.050   WN-LEXICAL            SET-BUFFER-CHUNK WN-LEXICAL
S-104966240-1
     1.050   PROCEDURAL            CONFLICT-RESOLUTION
     1.100   PROCEDURAL            PRODUCTION-FIRED FIND-SYNONYMS
"amber" IS A SYNONYMS
     1.100   PROCEDURAL            CLEAR-BUFFER IMAGINAL
     1.100   IMAGINAL              SET-BUFFER-CHUNK IMAGINAL SYN-GOAL5
     1.100   PROCEDURAL            CONFLICT-RESOLUTION
     1.150   PROCEDURAL            PRODUCTION-FIRED FIND-WORD-DEFINITION
LOOKING FOR SENSE OF "gold"
     1.150   PROCEDURAL            CLEAR-BUFFER WN-LEXICAL
     1.150   WN-LEXICAL            RETRIEVE-WN-CHUNKS
     1.150   WN-LEXICAL            SELECT-FROM-RETRIEVED-WN-CHUNKS SET OF
SIZE 7
     1.150   WN-LEXICAL            SET-BUFFER-CHUNK WN-LEXICAL
S-301528730-1
     1.150   PROCEDURAL            CONFLICT-RESOLUTION
     1.200   PROCEDURAL            PRODUCTION-FIRED FIND-WORD-GLOSS
LOOKING FOR DEFINITION OF "gold"
     1.200   PROCEDURAL            CLEAR-BUFFER WN-LEXICAL
     1.200   WN-LEXICAL            RETRIEVE-WN-CHUNKS
     1.200   WN-LEXICAL            SELECT-FROM-RETRIEVED-WN-CHUNKS SET OF
SIZE 1
     1.200   WN-LEXICAL            SET-BUFFER-CHUNK WN-LEXICAL
G-301528730-1
     1.200   PROCEDURAL            CONFLICT-RESOLUTION
     1.250   PROCEDURAL            PRODUCTION-FIRED SAY-DEFINITION
THE DEFINITION OF "gold" IS "made from or covered with gold;
         \"gold coins\";
         \"the gold dome of the Capitol\";
         \"the golden calf\"; \"gilded icons\""
     1.250   PROCEDURAL            CLEAR-BUFFER WN-LEXICAL
```

```
     1.250   WN-LEXICAL             RETRIEVE-WN-CHUNKS
     1.250   WN-LEXICAL             SELECT-FROM-RETRIEVED-WN-CHUNKS SET OF
SIZE 3
     1.250   WN-LEXICAL             SET-BUFFER-CHUNK WN-LEXICAL
S-301528730-2
     1.250   PROCEDURAL             CONFLICT-RESOLUTION
     1.300   PROCEDURAL             PRODUCTION-FIRED FIND-SYNONYMS
"golden" IS A SYNONYMS
     1.300   PROCEDURAL             CLEAR-BUFFER IMAGINAL
     1.300   IMAGINAL               SET-BUFFER-CHUNK IMAGINAL SYN-GOAL6
     1.300   PROCEDURAL             CONFLICT-RESOLUTION
     1.350   PROCEDURAL             PRODUCTION-FIRED FIND-WORD-DEFINITION
LOOKING FOR SENSE OF "gold"
     1.350   PROCEDURAL             CLEAR-BUFFER WN-LEXICAL
     1.350   WN-LEXICAL             RETRIEVE-WN-CHUNKS
     1.350   WN-LEXICAL             SELECT-FROM-RETRIEVED-WN-CHUNKS SET OF
SIZE 7
     1.350   WN-LEXICAL             SET-BUFFER-CHUNK WN-LEXICAL
S-105141492-1
     1.350   PROCEDURAL             CONFLICT-RESOLUTION
     1.400   PROCEDURAL             PRODUCTION-FIRED FIND-WORD-GLOSS
LOOKING FOR DEFINITION OF "gold"
     1.400   PROCEDURAL             CLEAR-BUFFER WN-LEXICAL
     1.400   WN-LEXICAL             RETRIEVE-WN-CHUNKS
     1.400   WN-LEXICAL             SELECT-FROM-RETRIEVED-WN-CHUNKS SET OF
SIZE 1
     1.400   WN-LEXICAL             SET-BUFFER-CHUNK WN-LEXICAL
G-105141492-1
     1.400   PROCEDURAL             CONFLICT-RESOLUTION
     1.450   PROCEDURAL             PRODUCTION-FIRED SAY-DEFINITION
THE DEFINITION OF "gold" IS "something likened to the metal
    in brightness or preciousness or superiority etc.;
    \"the child was as good as gold\"; \"she has a heart of gold\""
     1.450   PROCEDURAL             CLEAR-BUFFER WN-LEXICAL
     1.450   WN-LEXICAL             RETRIEVE-WN-CHUNKS
     1.450   WN-LEXICAL             SELECT-FROM-RETRIEVED-WN-CHUNKS SET OF
SIZE 1
     1.450   WN-LEXICAL             SET-BUFFER-CHUNK WN-LEXICAL
S-105141492-1
     1.450   PROCEDURAL             CONFLICT-RESOLUTION
     1.500   PROCEDURAL             PRODUCTION-FIRED FIND-SYNONYMS
"gold" IS A SYNONYMS
     1.500   PROCEDURAL             CLEAR-BUFFER IMAGINAL
```

```
     1.500   IMAGINAL              SET-BUFFER-CHUNK IMAGINAL SYN-GOAL7
     1.500   PROCEDURAL            CONFLICT-RESOLUTION
     1.550   PROCEDURAL            PRODUCTION-FIRED FIND-WORD-DEFINITION
LOOKING FOR SENSE OF "gold"
     1.550   PROCEDURAL            CLEAR-BUFFER WN-LEXICAL
     1.550   WN-LEXICAL            RETRIEVE-WN-CHUNKS
     1.550   WN-LEXICAL            SELECT-FROM-RETRIEVED-WN-CHUNKS SET OF
SIZE 7
     1.550   PROCEDURAL            CONFLICT-RESOLUTION
     1.600   PROCEDURAL            PRODUCTION-FIRED DO-NOT-KNOW-WORD
I KNOW NOTHING ABOUT "gold"
     1.600   PROCEDURAL            CLEAR-BUFFER IMAGINAL
     1.600   PROCEDURAL            CLEAR-BUFFER WN-LEXICAL
     1.600   PROCEDURAL            CLEAR-BUFFER RETRIEVAL
     1.600   DECLARATIVE           START-RETRIEVAL
     1.600   DECLARATIVE           RETRIEVED-CHUNK SYN-GOAL1-0
     1.600   DECLARATIVE           SET-BUFFER-CHUNK RETRIEVAL SYN-GOAL1-0
     1.600   PROCEDURAL            CONFLICT-RESOLUTION
     1.650   PROCEDURAL            PRODUCTION-FIRED START-EXPLORING-WORD
     1.650   PROCEDURAL            CLEAR-BUFFER WN-LEXICAL
     1.650   WN-LEXICAL            CLEAR-WN-LEXICAL-SYNSET-CONTEXT
     1.650   PROCEDURAL            CONFLICT-RESOLUTION
     1.700   PROCEDURAL            PRODUCTION-FIRED EXPLORE-WORD
     1.700   PROCEDURAL            CLEAR-BUFFER WN-LEXICAL
     1.700   WN-LEXICAL            RETRIEVE-WN-CHUNKS
     1.700   WN-LEXICAL            SELECT-FROM-RETRIEVED-WN-CHUNKS SET OF
SIZE 5
     1.700   WN-LEXICAL            SET-BUFFER-CHUNK WN-LEXICAL
S-300369941-2
     1.700   PROCEDURAL            CONFLICT-RESOLUTION
     1.750   PROCEDURAL            PRODUCTION-FIRED SAY-WORD
SELECTED WORD "gilded"
SOLUTION "gilded"
     1.750   PROCEDURAL            CLEAR-BUFFER RETRIEVAL
     1.750   DECLARATIVE           START-RETRIEVAL
     1.750   DECLARATIVE           RETRIEVED-CHUNK CLUE-1-0
     1.750   DECLARATIVE           SET-BUFFER-CHUNK RETRIEVAL CLUE-1-0
     1.750   PROCEDURAL            CONFLICT-RESOLUTION
     1.800   PROCEDURAL            PRODUCTION-FIRED UPDATE-CLUE
SOLVED CLUE 1
     1.800   PROCEDURAL            CONFLICT-RESOLUTION
     1.850   PROCEDURAL            PRODUCTION-FIRED
              CLEAR-BUFFER-AND-PICK-UP-NEXT-CLUE
```

```
     1.850   PROCEDURAL            CLEAR-BUFFER WN-LEXICAL
     1.850   PROCEDURAL            CLEAR-BUFFER GOAL
     1.850   PROCEDURAL            CLEAR-BUFFER RETRIEVAL
     1.850   GOAL                  SET-BUFFER-CHUNK GOAL SYN-GOAL8
     1.850   DECLARATIVE           START-RETRIEVAL
     1.850   DECLARATIVE           RETRIEVED-CHUNK CLUE-4
     1.850   DECLARATIVE           SET-BUFFER-CHUNK RETRIEVAL CLUE-4
     1.850   PROCEDURAL            CONFLICT-RESOLUTION
     1.900   PROCEDURAL            PRODUCTION-FIRED
UPDATE-GOAL-TO-GET-SYNONYMS
     1.900   PROCEDURAL            CLEAR-BUFFER WN-LEXICAL
     1.900   WN-LEXICAL            CLEAR-WN-LEXICAL-SYNSET-CONTEXT
     1.900   PROCEDURAL            CONFLICT-RESOLUTION
     1.950   PROCEDURAL            PRODUCTION-FIRED FIND-WORD-DEFINITION
LOOKING FOR SENSE OF "ljad;"
     1.950   PROCEDURAL            CLEAR-BUFFER WN-LEXICAL
     1.950   WN-LEXICAL            RETRIEVE-WN-CHUNKS
     1.950   PROCEDURAL            CONFLICT-RESOLUTION
     2.000   PROCEDURAL            PRODUCTION-FIRED DO-NOT-KNOW-WORD
I KNOW NOTHING ABOUT "ljad;"
     2.000   PROCEDURAL            CLEAR-BUFFER IMAGINAL
     2.000   PROCEDURAL            CLEAR-BUFFER WN-LEXICAL
     2.000   PROCEDURAL            CLEAR-BUFFER RETRIEVAL
     2.000   DECLARATIVE           START-RETRIEVAL
     2.000   DECLARATIVE           RETRIEVED-CHUNK SYN-GOAL2-0
     2.000   DECLARATIVE           SET-BUFFER-CHUNK RETRIEVAL SYN-GOAL2-0
     2.000   PROCEDURAL            CONFLICT-RESOLUTION
     2.050   PROCEDURAL            PRODUCTION-FIRED START-EXPLORING-WORD
     2.050   PROCEDURAL            CLEAR-BUFFER WN-LEXICAL
     2.050   WN-LEXICAL            CLEAR-WN-LEXICAL-SYNSET-CONTEXT
     2.050   PROCEDURAL            CONFLICT-RESOLUTION
     2.100   PROCEDURAL            PRODUCTION-FIRED EXPLORE-WORD
     2.100   PROCEDURAL            CLEAR-BUFFER WN-LEXICAL
     2.100   WN-LEXICAL            RETRIEVE-WN-CHUNKS
     2.100   WN-LEXICAL            SELECT-FROM-RETRIEVED-WN-CHUNKS SET OF
SIZE 3
     2.100   WN-LEXICAL            SET-BUFFER-CHUNK WN-LEXICAL
S-114638799-3
     2.100   PROCEDURAL            CONFLICT-RESOLUTION
     2.150   PROCEDURAL            PRODUCTION-FIRED SAY-WORD-1
WORD "atomic number 79"
     2.150   PROCEDURAL            CONFLICT-RESOLUTION
     2.200   PROCEDURAL            PRODUCTION-FIRED EXPLORE-WORD
```

```
     2.200   PROCEDURAL            CLEAR-BUFFER WN-LEXICAL
     2.200   WN-LEXICAL            RETRIEVE-WN-CHUNKS
     2.200   WN-LEXICAL            SELECT-FROM-RETRIEVED-WN-CHUNKS SET OF
SIZE 3
     2.200   WN-LEXICAL            SET-BUFFER-CHUNK WN-LEXICAL
S-114638799-2
     2.200   PROCEDURAL            CONFLICT-RESOLUTION
     2.250   PROCEDURAL            PRODUCTION-FIRED SAY-WORD-1
WORD "Au"
     2.250   PROCEDURAL            CONFLICT-RESOLUTION
     2.300   PROCEDURAL            PRODUCTION-FIRED EXPLORE-WORD
     2.300   PROCEDURAL            CLEAR-BUFFER WN-LEXICAL
     2.300   WN-LEXICAL            RETRIEVE-WN-CHUNKS
     2.300   WN-LEXICAL            SELECT-FROM-RETRIEVED-WN-CHUNKS SET OF
SIZE 3
     2.300   WN-LEXICAL            SET-BUFFER-CHUNK WN-LEXICAL
S-114638799-1
     2.300   PROCEDURAL            CONFLICT-RESOLUTION
     2.350   PROCEDURAL            PRODUCTION-FIRED SAY-WORD-1
WORD "gold"
     2.350   PROCEDURAL            CONFLICT-RESOLUTION
     2.400   PROCEDURAL            PRODUCTION-FIRED EXPLORE-WORD
     2.400   PROCEDURAL            CLEAR-BUFFER WN-LEXICAL
     2.400   WN-LEXICAL            RETRIEVE-WN-CHUNKS
     2.400   WN-LEXICAL            SELECT-FROM-RETRIEVED-WN-CHUNKS SET OF
SIZE 3
     2.400   PROCEDURAL            CONFLICT-RESOLUTION
     2.450   PROCEDURAL            PRODUCTION-FIRED
                                            NO-MORE-WORDS-IN-THIS-SENSE
PICKING UP NEXT WORD
     2.450   PROCEDURAL            CLEAR-BUFFER RETRIEVAL
     2.450   DECLARATIVE           START-RETRIEVAL
     2.450   DECLARATIVE           RETRIEVED-CHUNK SYN-GOAL3-0
     2.450   DECLARATIVE           SET-BUFFER-CHUNK RETRIEVAL SYN-GOAL3-0
     2.450   PROCEDURAL            CONFLICT-RESOLUTION
     2.500   PROCEDURAL            PRODUCTION-FIRED TEMP-PROD
     2.500   PROCEDURAL            CONFLICT-RESOLUTION
     2.550   PROCEDURAL            PRODUCTION-FIRED START-EXPLORING-WORD
     2.550   PROCEDURAL            CLEAR-BUFFER WN-LEXICAL
     2.550   WN-LEXICAL            CLEAR-WN-LEXICAL-SYNSET-CONTEXT
     2.550   PROCEDURAL            CONFLICT-RESOLUTION
     2.600   PROCEDURAL            PRODUCTION-FIRED EXPLORE-WORD
     2.600   PROCEDURAL            CLEAR-BUFFER WN-LEXICAL
```

```
     2.600   WN-LEXICAL            RETRIEVE-WN-CHUNKS
     2.600   WN-LEXICAL            SELECT-FROM-RETRIEVED-WN-CHUNKS SET OF
SIZE 1
     2.600   WN-LEXICAL            SET-BUFFER-CHUNK WN-LEXICAL
S-113371760-1
     2.600   PROCEDURAL            CONFLICT-RESOLUTION
     2.650   PROCEDURAL            PRODUCTION-FIRED SAY-WORD-1
WORD "gold"
     2.650   PROCEDURAL            CONFLICT-RESOLUTION
     2.700   PROCEDURAL            PRODUCTION-FIRED EXPLORE-WORD
     2.700   PROCEDURAL            CLEAR-BUFFER WN-LEXICAL
     2.700   WN-LEXICAL            RETRIEVE-WN-CHUNKS
     2.700   WN-LEXICAL            SELECT-FROM-RETRIEVED-WN-CHUNKS SET OF
SIZE 1
     2.700   PROCEDURAL            CONFLICT-RESOLUTION
     2.750   PROCEDURAL            PRODUCTION-FIRED
                                       NO-MORE-WORDS-IN-THIS-SENSE
PICKING UP NEXT WORD
     2.750   PROCEDURAL            CLEAR-BUFFER RETRIEVAL
     2.750   DECLARATIVE           START-RETRIEVAL
     2.750   DECLARATIVE           RETRIEVED-CHUNK SYN-GOAL4-0
     2.750   DECLARATIVE           SET-BUFFER-CHUNK RETRIEVAL SYN-GOAL4-0
     2.750   PROCEDURAL            CONFLICT-RESOLUTION
     2.800   PROCEDURAL            PRODUCTION-FIRED TEMP-PROD
     2.800   PROCEDURAL            CONFLICT-RESOLUTION
     2.850   PROCEDURAL            PRODUCTION-FIRED START-EXPLORING-WORD
     2.850   PROCEDURAL            CLEAR-BUFFER WN-LEXICAL
     2.850   WN-LEXICAL            CLEAR-WN-LEXICAL-SYNSET-CONTEXT
     2.850   PROCEDURAL            CONFLICT-RESOLUTION
     2.900   PROCEDURAL            PRODUCTION-FIRED EXPLORE-WORD
     2.900   PROCEDURAL            CLEAR-BUFFER WN-LEXICAL
     2.900   WN-LEXICAL            RETRIEVE-WN-CHUNKS
     2.900   WN-LEXICAL            SELECT-FROM-RETRIEVED-WN-CHUNKS SET OF
SIZE 1
     2.900   WN-LEXICAL            SET-BUFFER-CHUNK WN-LEXICAL
S-113353446-1
     2.900   PROCEDURAL            CONFLICT-RESOLUTION
     2.950   PROCEDURAL            PRODUCTION-FIRED SAY-WORD-1
WORD "gold"
     2.950   PROCEDURAL            CONFLICT-RESOLUTION
     3.000   PROCEDURAL            PRODUCTION-FIRED EXPLORE-WORD
     3.000   PROCEDURAL            CLEAR-BUFFER WN-LEXICAL
     3.000   WN-LEXICAL            RETRIEVE-WN-CHUNKS
```

```
     3.000   WN-LEXICAL            SELECT-FROM-RETRIEVED-WN-CHUNKS SET OF
SIZE 1
     3.000   PROCEDURAL            CONFLICT-RESOLUTION
     3.050   PROCEDURAL            PRODUCTION-FIRED
                                       NO-MORE-WORDS-IN-THIS-SENSE
PICKING UP NEXT WORD
     3.050   PROCEDURAL            CLEAR-BUFFER RETRIEVAL
     3.050   DECLARATIVE           START-RETRIEVAL
     3.050   DECLARATIVE           RETRIEVED-CHUNK SYN-GOAL5-0
     3.050   DECLARATIVE           SET-BUFFER-CHUNK RETRIEVAL SYN-GOAL5-0
     3.050   PROCEDURAL            CONFLICT-RESOLUTION
     3.100   PROCEDURAL            PRODUCTION-FIRED TEMP-PROD
     3.100   PROCEDURAL            CONFLICT-RESOLUTION
     3.150   PROCEDURAL            PRODUCTION-FIRED START-EXPLORING-WORD
     3.150   PROCEDURAL            CLEAR-BUFFER WN-LEXICAL
     3.150   WN-LEXICAL            CLEAR-WN-LEXICAL-SYNSET-CONTEXT
     3.150   PROCEDURAL            CONFLICT-RESOLUTION
     3.200   PROCEDURAL            PRODUCTION-FIRED EXPLORE-WORD
     3.200   PROCEDURAL            CLEAR-BUFFER WN-LEXICAL
     3.200   WN-LEXICAL            RETRIEVE-WN-CHUNKS
     3.200   WN-LEXICAL            SELECT-FROM-RETRIEVED-WN-CHUNKS SET OF
SIZE 2
     3.200   WN-LEXICAL            SET-BUFFER-CHUNK WN-LEXICAL
S-104966240-2
     3.200   PROCEDURAL            CONFLICT-RESOLUTION
     3.250   PROCEDURAL            PRODUCTION-FIRED SAY-WORD-1
WORD "gold"
     3.250   PROCEDURAL            CONFLICT-RESOLUTION
     3.300   PROCEDURAL            PRODUCTION-FIRED EXPLORE-WORD
     3.300   PROCEDURAL            CLEAR-BUFFER WN-LEXICAL
     3.300   WN-LEXICAL            RETRIEVE-WN-CHUNKS
     3.300   WN-LEXICAL            SELECT-FROM-RETRIEVED-WN-CHUNKS SET OF
SIZE 2
     3.300   WN-LEXICAL            SET-BUFFER-CHUNK WN-LEXICAL
S-104966240-1
     3.300   PROCEDURAL            CONFLICT-RESOLUTION
     3.350   PROCEDURAL            PRODUCTION-FIRED SAY-WORD-1
WORD "amber"
     3.350   PROCEDURAL            CONFLICT-RESOLUTION
     3.400   PROCEDURAL            PRODUCTION-FIRED EXPLORE-WORD
     3.400   PROCEDURAL            CLEAR-BUFFER WN-LEXICAL
     3.400   WN-LEXICAL            RETRIEVE-WN-CHUNKS
     3.400   WN-LEXICAL            SELECT-FROM-RETRIEVED-WN-CHUNKS SET OF
```

```
SIZE 2
     3.400   PROCEDURAL            CONFLICT-RESOLUTION
     3.450   PROCEDURAL            PRODUCTION-FIRED
                                     NO-MORE-WORDS-IN-THIS-SENSE
PICKING UP NEXT WORD
     3.450   PROCEDURAL            CLEAR-BUFFER RETRIEVAL
     3.450   DECLARATIVE           START-RETRIEVAL
     3.450   DECLARATIVE           RETRIEVED-CHUNK SYN-GOAL6-0
     3.450   DECLARATIVE           SET-BUFFER-CHUNK RETRIEVAL SYN-GOAL6-0
     3.450   PROCEDURAL            CONFLICT-RESOLUTION
     3.500   PROCEDURAL            PRODUCTION-FIRED TEMP-PROD
     3.500   PROCEDURAL            CONFLICT-RESOLUTION
     3.550   PROCEDURAL            PRODUCTION-FIRED START-EXPLORING-WORD
     3.550   PROCEDURAL            CLEAR-BUFFER WN-LEXICAL
     3.550   WN-LEXICAL            CLEAR-WN-LEXICAL-SYNSET-CONTEXT
     3.550   PROCEDURAL            CONFLICT-RESOLUTION
     3.600   PROCEDURAL            PRODUCTION-FIRED EXPLORE-WORD
     3.600   PROCEDURAL            CLEAR-BUFFER WN-LEXICAL
     3.600   WN-LEXICAL            RETRIEVE-WN-CHUNKS
     3.600   WN-LEXICAL            SELECT-FROM-RETRIEVED-WN-CHUNKS SET OF
SIZE 3
     3.600   WN-LEXICAL            SET-BUFFER-CHUNK WN-LEXICAL
S-301528730-3
     3.600   PROCEDURAL            CONFLICT-RESOLUTION
     3.650   PROCEDURAL            PRODUCTION-FIRED SAY-WORD-1
WORD "gilded"
     3.650   PROCEDURAL            CONFLICT-RESOLUTION
     3.700   PROCEDURAL            PRODUCTION-FIRED EXPLORE-WORD
     3.700   PROCEDURAL            CLEAR-BUFFER WN-LEXICAL
     3.700   WN-LEXICAL            RETRIEVE-WN-CHUNKS
     3.700   WN-LEXICAL            SELECT-FROM-RETRIEVED-WN-CHUNKS SET OF
SIZE 3
     3.700   WN-LEXICAL            SET-BUFFER-CHUNK WN-LEXICAL
S-301528730-2
     3.700   PROCEDURAL            CONFLICT-RESOLUTION
     3.750   PROCEDURAL            PRODUCTION-FIRED SAY-WORD-1
WORD "golden"
     3.750   PROCEDURAL            CONFLICT-RESOLUTION
     3.800   PROCEDURAL            PRODUCTION-FIRED EXPLORE-WORD
     3.800   PROCEDURAL            CLEAR-BUFFER WN-LEXICAL
     3.800   WN-LEXICAL            RETRIEVE-WN-CHUNKS
     3.800   WN-LEXICAL            SELECT-FROM-RETRIEVED-WN-CHUNKS SET OF
SIZE 3
```

```
     3.800   WN-LEXICAL            SET-BUFFER-CHUNK WN-LEXICAL
S-301528730-1
     3.800   PROCEDURAL            CONFLICT-RESOLUTION
     3.850   PROCEDURAL            PRODUCTION-FIRED SAY-WORD-1
WORD "gold"
     3.850   PROCEDURAL            CONFLICT-RESOLUTION
     3.900   PROCEDURAL            PRODUCTION-FIRED EXPLORE-WORD
     3.900   PROCEDURAL            CLEAR-BUFFER WN-LEXICAL
     3.900   WN-LEXICAL            RETRIEVE-WN-CHUNKS
     3.900   WN-LEXICAL            SELECT-FROM-RETRIEVED-WN-CHUNKS SET OF
SIZE 3
     3.900   PROCEDURAL            CONFLICT-RESOLUTION
     3.950   PROCEDURAL            PRODUCTION-FIRED
NO-MORE-WORDS-IN-THIS-SENSE
PICKING UP NEXT WORD
     3.950   PROCEDURAL            CLEAR-BUFFER RETRIEVAL
     3.950   DECLARATIVE           START-RETRIEVAL
     3.950   DECLARATIVE           RETRIEVED-CHUNK SYN-GOAL7-0
     3.950   DECLARATIVE           SET-BUFFER-CHUNK RETRIEVAL SYN-GOAL7-0
     3.950   PROCEDURAL            CONFLICT-RESOLUTION
     4.000   PROCEDURAL            PRODUCTION-FIRED TEMP-PROD
     4.000   PROCEDURAL            CONFLICT-RESOLUTION
     4.050   PROCEDURAL            PRODUCTION-FIRED START-EXPLORING-WORD
     4.050   PROCEDURAL            CLEAR-BUFFER WN-LEXICAL
     4.050   WN-LEXICAL            CLEAR-WN-LEXICAL-SYNSET-CONTEXT
     4.050   PROCEDURAL            CONFLICT-RESOLUTION
     4.100   PROCEDURAL            PRODUCTION-FIRED EXPLORE-WORD
     4.100   PROCEDURAL            CLEAR-BUFFER WN-LEXICAL
     4.100   WN-LEXICAL            RETRIEVE-WN-CHUNKS
     4.100   WN-LEXICAL            SELECT-FROM-RETRIEVED-WN-CHUNKS SET OF
SIZE 1
     4.100   WN-LEXICAL            SET-BUFFER-CHUNK WN-LEXICAL
S-105141492-1
     4.100   PROCEDURAL            CONFLICT-RESOLUTION
     4.150   PROCEDURAL            PRODUCTION-FIRED SAY-WORD-1
WORD "gold"
     4.150   PROCEDURAL            CONFLICT-RESOLUTION
     4.200   PROCEDURAL            PRODUCTION-FIRED EXPLORE-WORD
     4.200   PROCEDURAL            CLEAR-BUFFER WN-LEXICAL
     4.200   WN-LEXICAL            RETRIEVE-WN-CHUNKS
     4.200   WN-LEXICAL            SELECT-FROM-RETRIEVED-WN-CHUNKS SET OF
SIZE 1
     4.200   PROCEDURAL            CONFLICT-RESOLUTION
```

```
     4.250   PROCEDURAL            PRODUCTION-FIRED
NO-MORE-WORDS-IN-THIS-SENSE
PICKING UP NEXT WORD
     4.250   PROCEDURAL            CLEAR-BUFFER RETRIEVAL
     4.250   DECLARATIVE           START-RETRIEVAL
     4.250   DECLARATIVE           RETRIEVAL-FAILURE
     4.250   PROCEDURAL            CONFLICT-RESOLUTION
     4.300   PROCEDURAL            PRODUCTION-FIRED TEMP-PROD
     4.300   PROCEDURAL            CONFLICT-RESOLUTION
     4.350   PROCEDURAL            PRODUCTION-FIRED EXPLORED-ALL-SENSES
COULD NOT SOLVE CLUE 4
EXPLORED ALL SENSES
     4.350   PROCEDURAL            CLEAR-BUFFER WN-LEXICAL
     4.350   PROCEDURAL            CLEAR-BUFFER GOAL
     4.350   PROCEDURAL            CLEAR-BUFFER RETRIEVAL
     4.350   GOAL                  SET-BUFFER-CHUNK GOAL SYN-GOAL9
     4.350   DECLARATIVE           START-RETRIEVAL
     4.350   DECLARATIVE           RETRIEVAL-FAILURE
     4.350   PROCEDURAL            CONFLICT-RESOLUTION
     4.350   ------                Stopped because no events left to
process
```