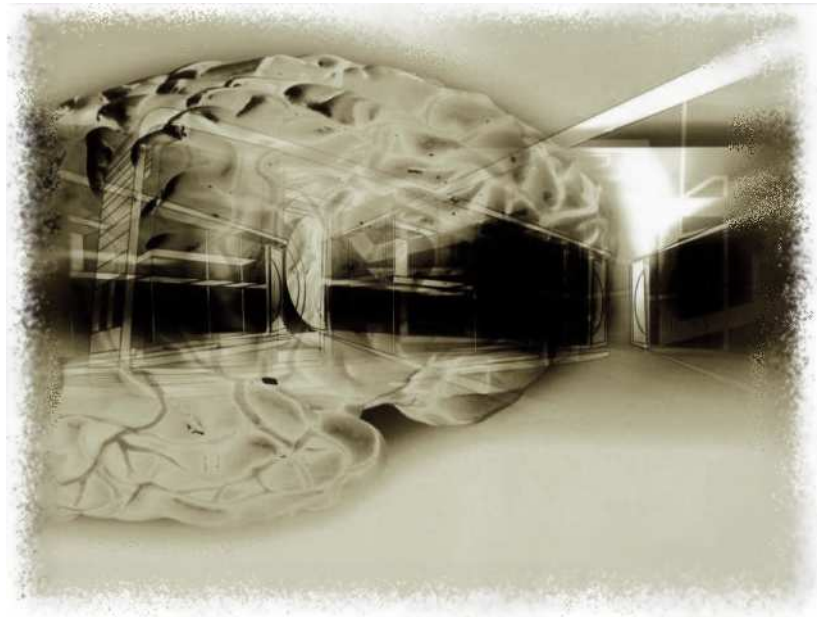


# Exploring the Intersection of Decisions and Architectures

Eric Dimperio



Qualifying examination essay 3 of 3.

Advisory committee:

Professor Jerome Busemeyer, Ph.D.

Professor Robert Goldstone, Ph. D.

Professor John K. Kruschke, Ph. D.

Professor Larry Yaeger

## **Exploring the Intersection of Decisions and Architectures**

Eric Dimperio

### **Foreword**

This article serves as an exploration of the meeting of two fields. Its original purpose was to serve simply as an exploration of cognitive architectures. However, since Allen Newell's call to arms for such architectures in 1990, the field has grown so large, that a high level overview would fail to capture some of the intriguing challenges faced by these complex systems. For this reason, I decided to explore how models from a specific line of research are being incorporated into architectures. The selection of decision making was motivated mostly by the convenience of working with others who have extensive knowledge on the subject. My own fascination with the subject has certainly grown throughout the writing of this article.

### **Intro to cognitive architectures**

In the field of cognitive modeling, mathematical or well-specified computational systems are used to enhance knowledge of cognitive functions. These models represent a formal version of some theory of cognition. These models can be used to describe the underlying nature of the process, to predict future behaviors, and/or to suggest courses of action necessary to alter behaviors toward some goal. Most models are of unique behaviors observed in specific environments. In 1973, Allen Newell wrote an article entitled "You can't play 20 questions with nature and win" that challenged the current practice of exploring theories by forming strict dichotomies such as serial versus parallel processing and massed versus distributed practice. Not only did he make such criticisms,

but he made three suggestions for the future. First, he suggested a need for complete processing models. Second, he noted a desire to see specific tasks thoroughly analyzed to identify all of the psychological phenomena involved. Third, he recommended the application of models of a particular psychological phenomenon be applied to a diverse set of circumstances where that phenomenon arises (Newell, 1973).

These ideas were the seeds that led to the call for what he labeled *unified theories of cognition* (Newell, 1990). A unified theory of cognition would not be a vague theory replacing existing work, but instead a formalization that brings together what has already been learned from modeling many aspects of cognition. According to Newell, such a theory could be used to integrate models of problem solving, decision making, memory, learning, perception, motor behavior, language, motivation, emotion, and eventually dreaming, imagining, and daydreaming. He recognized that some of those goals were a bit lofty, but at present, unifying models in some of these areas was possible.

For what reason do we need a unifying theory? For one, it can make things more convenient if multiple researchers develop models under the same unifying theory. They can communicate ideas directly and integrate models into larger cohesive systems. Beyond such practical reasons, a unified theory is more comparable to human cognition where a single mind controls behavior. Although we may be able to break the mind into component parts, we should expect interactions and dependencies between such components (Newell, 1990). These dependencies can only be explored under a grand theory where all aspects of cognition can be modeled as one. Newell suggests that this new holistic perspective as well as the practical effects of standardizing will speed the progress of science (1990). Also, since unified models can include explanations of a

diverse set of psychological phenomena, they can serve in the study of real-world problems (Anderson, Bothell, Byrne, Douglass, Lebiere, & Qin, 2004).

A hierarchy exists to describe the levels of analysis commonly seen in the modeling of cognition (Sun, 2006a). This includes the social or cultural level, the psychological level, the componential level, and the physiological level. The social level includes multiagent models for the study of collective processes. The psychological level utilizes models of agents as a whole, whereas the componential level studies intra-agent processes through the use of modular constructions of agent models. Finally the physiological level utilizes models of the biological substrates used to compose the modules in the componential level. A unified theory can aid in cross-level analysis. Existing models at the component level can be studied at the psychological level. Competing models of memory may both accurately predict performance on a list recall task, but may lead to drastically different behaviors when combined with perceptual and motor systems in a maze exploration task. More recently, the growth of knowledge in the field of neuroscience has been shaping the development of cognitive models. Unified theories allow the integration of knowledge from cognitive science and neuroscience at a system-wide level (Anderson et al., 2004).

Such a unified theory could take on many forms, however, in order to be testable and specified at an appropriate level, such a theory should exist as a sort of cognitive architecture. An architecture specifies the fixed parts of a system. Humans may gather different knowledge that influences how they go about solving problems, but at some level, the mechanisms by which humans receive information, store it, and retrieve it for later use remain constant. Such mechanisms are specified by a cognitive architecture.

These mechanisms should also remain constant along the entire spectrum of cognitive activities. An architecture can in turn run a model. A model is specified by the pre-existing knowledge (and possibly the incoming knowledge during a dynamic task) contained within a system. The architecture specifies what is done with that knowledge to produce behaviors (Taatgen, 1999). Several systems / processes commonly studied in fields such as psychology and artificial intelligence have potential to serve as cognitive architectures. Connectionist / neural network systems, production systems, dynamic systems, and random walk models have all been used to specify over-arching concepts that apply to a wide variety of cognitive functions (Newell, 1990). Nonetheless, thoroughly formalized cognitive architectures have primarily been specified using symbol systems (Newell; 1990; Meyer & Kieras 1997) or hybrid symbolic / connectionist systems (Anderson & Lebiere, 1998; Just & Carpenter, 1992; Sun, 2006b). See Taatgen (1999) for a detailed explanation of the hurdles that must be overcome by other systems before they can be utilized as cognitive architectures.

Symbol systems use well specified tokens to define specific entities (goals, knowledge, concepts, actions, etc.) involved in cognitive processes. The architecture defines how such symbols are manipulated based on the presence of other symbols. **A truly symbolic system does not impose any meaning to concepts outside of relationships and attributes defined in the model. This means that computational processes must be given a symbolic level description that gives meaning to numerical values (See appendix A for a sample ACT-R model description the process of adding numbers). In many models of cognition, computations are assumed to exist at a subconscious level inherent to underlying processes and cognitive mechanisms.** Hybrid systems use symbols as the

primary knowledge structure, but the manipulation and activation of those structures are defined by sub-symbolic processes. The selection of symbolic knowledge as goals, as active memories, and as actions depends on a variety of decision rules that act as one of the core functions of cognitive architectures.

The remainder of this article shall highlight the decision processes occurring within several popular architectures. This will include decisions made at the architectural level and how they influence decisions made at the behavioral level. A review of significant research in the field of decision making and preferential choice will provide a context for the appraisal of decisions in cognitive architectures. Although there seems to be an apparent disconnect between the levels at which decisions are studied in the field of cognitive architectures (decisions within cognitive processes) and decision making (choice behavior), several relationships shall be explored. First, how do choice behavior phenomena extend into the lower levels of cognition, and how might current decision models influence architectures. Second, a brief review will be presented to show how behavioral choices are made in a variety of contexts within various architectures. These contexts have generally fallen outside of the traditionally economically inspired choices in decision theory and instead deal with problem solving, decision heuristics, and dynamic decision making. Third, I hope to explore how current models of behavioral decision making can be expressed within existing architectures. Finally, I shall conclude with some remarks on how decision making research may constrain future versions of cognitive architectures.

### **Decision making: a brief history**

Decision theory has its roots in expected utility theory (Von Neumann & Morgenstern 1947; Savage 1954). According to EUT, all actions have some utility associated with them. This utility represents a personal valuation where a higher utility has greater value. Within EUT, decision makers are rational and always choose actions that maximize utility. The theory is somewhat limited and does not explain what might happen should two decisions have equal utility. This paradox is commonly known as *Buriden's ass* or the donkey paradox after 14<sup>th</sup> century philosopher Jean Buriden pointed out that a perfectly rational donkey placed exactly between two equivalent stacks of hay would starve since it would be unable to make a decision between the two (Belavkin, 2006).

One of the original ways of dealing with problems stemming from using a maximum decision rule was to utilize Luce's choice axiom (Luce, 1959). The Luce choice rule can deal with the fact that people do not make consistent choices in repetitive tasks by allowing stochastic choices where the probability of choosing some alternative  $i$  is determined by

$$p(i) = \frac{w_i}{\sum_j w_j}$$

where  $w$  is some measure of the saliency of the alternative. This measure can be a utility or some transformation of a utility. Luce's choice rule implies a property called independence from irrelevant alternatives (IIA). The IIA property says that if X and Y are options in a set of alternatives and  $p(X) > p(Y)$ , then adding another option Z to the set of alternative should not change the preference of X over Y. However, people are regularly shown to violate this property (Tversky, 1972). This violation can be seen in what is

commonly referred to as the similarity effect (Roe, Busemeyer, & Townsend, 2001). As an example of the similarity effect, imagine a music listener who will choose to listen to a song by country artist X over rock 'n roll artist Y about 60% of the time when presented with a binary choice<sup>1</sup>. However, when presented with those choices along with an additional song by country artist S, the listener chooses X about 30% of the time, S 30% of the time, and Y about 40% of the time. Country artists are still preferred as a group, but within the group, choice probabilities are split among similar items.

The elimination by aspects (EBA) model is a heuristic choice model created to account for such results (Tversky, 1972). According to EBA, all choice alternatives can be described by a set of aspects or properties. During the decision process, a single aspect is randomly selected and all options that do not possess that particular aspect are eliminated from the set of possible alternatives. The process continues until a single option is left.

People also display certain seemingly irrational behaviors when making choices under risky conditions (Kahneman & Tversky, 1979). They tend to give added weight to events that are certain to occur over those that are only probable. They also tend to ignore components that are shared between two choices with uncertain outcomes. As an attempt to explain such violations, Kahneman & Tversky (1979) proposed their prospect theory for which Daniel Kahneman won the Nobel Prize in economics. Prospect theory replaces utilities with values that are based on a reference point from which losses are weighted greater than gains. Also, probabilities are transformed such that low probabilities are overestimated while moderate and high probabilities are under

---

<sup>1</sup> I love rock'n roll. put another dime in the jukebox baby.

estimated. Prospect theory has gained popularity in economics as a method of predicting choice probabilities.

More recently, several new decision models have gained in popularity. In addition to being able to explain more qualitative effects observed in experiments of human choice behavior, these new models are dynamic process models that also make predictions of response times. The multi-attribute decision field theory model (MDFT) and the leaky accumulator model (LCA) of decision making are two such models (Roe, Busemeyer, & Townsend 2001; Usher & McClelland 2004). Both are variations of an Ornstein-Uhlenbeck diffusion process. The diffusion process is a continuous time Gaussian stochastic process that allows values representing choice alternatives to stochastically accumulate over time until one value reaches a threshold. Both models can be estimated using discrete time Markov processes.

As mentioned earlier, Tversky's EBA model is able to explain the similarity effect. Unfortunately, the EBA entails a property called regularity that has been shown to be violated with the attraction effect (Roe et al.). Regularity says that if an additional option is added to a set of choice alternatives, all choice probabilities of the original options can only be reduced or stay the same. The attraction effect occurs when a clearly dominated item is added and actually boosts the probability of selecting the dominating item. If in the previous music example, we add a country music artist A who is not liked very much, the original country artist X looks much better in comparison to the similar but not as desirable option. Artist X will actually be chosen more often than it was when presented in the binary choice. Another effect that these new models are capable of

explaining is the compromise effect. According to the compromise effect, when three items that are shown to be indifferent in binary choices are presented all together, the option that represents a compromise will have a higher probability of being selected. Going along with the music example, we may have a situation where in addition to songs by country artist X and rock n' roll artist Y, we also have songs by country-rock fusion artist C. When presented in pairs (X-Y, X-C, C-Y) there exists no preference for any of the artists. However, if given the choice to pick a song when all three artists are available, the compromise C will be chosen significantly more often.

The models that we have looked at have been developed for very limited types of preferential decision tasks. They are meant to explain results taken from individual decision paradigms that often do not include interaction with an environment, learning over time, or making decisions as part of larger cognitive tasks. These are more the types of tasks that have been studied in the development of cognitive architectures. It would be prudent to begin to explore how decision making is approached within large scale cognitive architectures

### **Decision making within architectures**

Let us first address how decision making occurs within some existing cognitive architectures. In order to do this, I will provide a brief description of several architectures. The atomic components will be identified as well as how choices are made between how those components are applied at any given time. It was stated earlier that a model built in an architecture is specified by a certain set of knowledge and the architecture defines how the systems operates on that knowledge. Human knowledge

often contains seemingly contradictory elements that may all apply at some given time. A person may hate birds, love penguins and know that penguins are birds all at the same time. It is an architecture's job to settle such conflicts. This represents the choice behavior that is internal to the architecture. Models of choice behavior expressed in an architecture may directly utilize these internal choice rules, or they may use them to guide a larger meta-choice process.

### Soar

When Allen Newell proposed the development of unified theories of cognition, he had developed an architecture to serve as a candidate for such a theory (Newell, 1990). That architecture is called Soar (States, Operators, And Reasoning) and was developed by Laird, Rosenbloom, and Newell (1987). Soar is a purely symbolic production system. Production systems utilizes productions - which are rules with an "if-then" type structure - to modify the internal state as well as to take action. Soar's primary memory is its production memory which is the main store of all knowledge known to the system. Soar also uses a working memory to represent the knowledge applicable to the current situation. Knowledge in working memory is contained in objects that provide the current state of the system, the current goal to be achieved, the current applicable operators that can update the system, and preferences on those operators. Soar's implementation represents a special type of problem-space computational model which treats problem-solving as a search through problem-space. Essentially, when a goal is present, knowledge is retrieved from procedural memory about what operator to apply that satisfies the goal given the current state. If that knowledge is not directly available, or it is

not immediately clear what operator should be used because of conflicting knowledge, an *impasse* is said to occur. A new sub-goal to solve the impasse is placed on the goal stack and the system follows the same methodology to solve the sub-problem.

When retrieving knowledge, all applicable procedures are activated and brought into working memory in parallel. In addition to directly suggesting operators, some procedures define preferences among operators. These preferences are the simple relationships: *acceptable*, *reject*, *better (worse)*, *best (worst)*, and *indifferent* (Newell, 1990). These operators can lead to an impasse if there is a tie in the preferences of two operators, or a conflict over which operator should be chosen. (Laird & Congdon, 2006).

Since an impasse becomes a new problem to be solved, almost all decisions made by the system are the direct result of knowledge embedded in the procedural memory. The only decision not a direct consequence of production rules occurs when there are multiple operators that are *indifferent* in which case one is chosen randomly (Laird & Congdon, 2006).

Soar is capable of learning over time using a process referred to as data chunking. The chunking mechanism adds productions to memory once the results of a subgoal have been found. When encountering the same situation in the future, preference is given to the newly formed chunk such that an impasse does not occur and processing the subgoal is avoided. With experience, Soar gains more knowledge and is able to solve problems quicker.

Soar's knowledge system approach leaves it with very rigid, rational decision making abilities. The preferences between operators give them a sort of value on an ordinal scale that has some analogy to utility values. The system logically explores its

knowledge to determine which operator has the greatest value and then chooses that operator. Although the knowledge in a model can cause the system to enter into more complicated decision processes at a higher level, within the system, Soar uses a very limited type of expected utility decision rule.

### ACT-R

The ACT-R (Adaptive Character of Thought – Rational) architecture is also a production system, but it is not a purely symbolic architecture. Unlike Soar, it uses sub-symbolic mechanisms to resolve conflicts within the system (Anderson, 1993; Anderson & Lebiere, 1998). ACT-R uses two different types of knowledge stores, each with its own style of conflict resolution. The first is the procedural memory. Procedural memory is stored as production rules. The rules define actions that can update goals, and fetch facts from declarative memory based. These actions are selected based on current goals and current known facts. Declarative memory (the second type of memory store) contains ‘chunks’ of information that contain facts about the world. As of ACT-R 5.0 (Anderson et al., 2004), productions can also send and receive chunks from modules other than declarative memory such as sensory and motor systems (Taatgen, Lebiere, & Anderson, 2006).

Procedural conflict resolution methods are a direct derivation of expected utility theory (Belavkin, 2006). Utility of some production  $i$  is calculated as

$U_i = P_i G - C + noise$  where  $P$  is an estimate of the probability of successful completion of an action based on past experience,  $G$  is the value gained by successful completion of

the production, and  $C$  is the cost of executing the production. The production with the highest utility is always selected. Although the noise in the system leads to some human-like errors, the probability of selecting any production  $i$  is:

$$p(i) = \frac{e^{u_i/t}}{\sum_j e^{u_j/t}} \text{ (Taatgen, Lebiere, & Anderson, 2006).}$$

Since this is Luce's choice rule, the conflict resolution mechanism in ACT-R cannot explain violations of independence of irrelevant alternative.

Another choice the system has to make occurs when multiple declarative chunks match a request. Selection of a chunk is based on an activation level where chunks with an activation level exceeding some threshold will be retrieved. Activation is determined by several things. The first is a baseline activation that is set by how recently and frequently the chunk was used. The second is a matching of individual elements in a chunk. For example, if I was searching for a subtraction fact to answer the question "8-2=?" then the addition fact "2+6=8" would get some activation based on the matching of the 2 and 8 elements. Elements are also weighted by a measure of how useful they have been in the past when that particular chunk was needed. If multiple chunks are retrieved, Luce's choice axiom governs the probability that any particular chunk will be selected. Unlike Soar, which would always fail in the presence of missing information, ACT-R can infer knowledge from facts that are similar to what is necessary so that it can attempt to meet the current goal.

ACT-R allows learning to occur at both the sub-symbolic and at the symbolic level. At the sub-symbolic level, successes and failures of productions influence their

utilities. With experience, ACT-R learns to favor productions that have proved useful in the past. The declarative parameters for base rate and associative strengths for individual elements also get learned based on the usage of chunks to satisfy goals. At the symbolic level, new declarative chunks and new productions can be learned. Achieved goals are stored as new chunks in declarative memory. A process named *production compilation* allows commonly paired productions to be combined into individual productions as well as the incorporation of declarative knowledge directly into production rules. For example suppose the following productions were used during a task involving adding two numbers:

```
IF asked to add addend1 and addend2
THEN retrieve the sum from an addition-fact with addend1 and addend2

IF adding and a sum has been retrieved
THEN type sum
```

In a specific situation where the system was asked to add the numbers 2 and 6, it could retrieve the matching chunk declaring the sum to be 8. It would then type the answer 8.

All of that can be compiled into a single new production rule:

```
IF asked to add 2 and 6
THEN type 8
```

When this production is created, it has zero utility and will not be chosen over the original first production (both productions would be valid in a situation where the system

is asked to add 2 and 6). However, if this same rule is compiled again, the system will see that it already exists and its estimated probability of success will be increased. For this reason, when ACT-R does repetitive tasks, it learns to combine steps. The learning processes in ACT-R all serve to speed up response times with repeated trials.

### EPIC

Executive-Process Interactive Control (EPIC) is a model that was developed primarily out of the need to model situations where people are involved in multiple concurrent tasks (Meyer & Kieras, 1997). The specific tasks of interest required significant interaction with the environment, such as talking on a cell phone and driving at the same time. EPIC is a production system, but differs from SOAR and ACT-R by emphasizing the role of perceptual and motor processing. EPIC is addressed here because of its significant role as a cognitive architecture, however, its internal decision making abilities are sparse. This is because EPIC does not introduce any limits on the number of production cognitive control can handle. Many productions may become active in parallel, and there is no conflict resolution. All limits are placed in the physical restrictions of peripheral controls. This places the burden on the modeler to define productions that include mechanisms for making decisions if they are necessary.

### CLARION

One of the more recent architectures to come about is Ron Sun's CLARION (2004, 2006). It was developed under a strong desire to increase the modularity of cognitive architectures. For this reason, CLARION consists of many more discreet

processing units and memory stores than other architectures. This also leads to greater complexity. The primary modules include an action-centered subsystem (ACS), non-action-centered subsystem (NACS), motivational subsystem (MS), and metacognitive subsystem (MCS). The ACS is responsible for selecting both external actions and internal changes. The NACS stores and maintains general knowledge. The MS controls the driving forces and the feedback to perceptual, action and cognitive components. The MCS has the role of monitoring and dynamically modifying operations of the other subsystems. Another point for modularization in CLARION is the dichotomy of implicit versus explicit memory. All subsystems are built using separate mechanisms for the encoding, storage, and retrieval of explicit memories and implicit memories. The explicit memory systems are symbolic in nature and are accessible to both the observer and the system itself. The implicit memory systems utilize neural network structures that distribute stored knowledge in an inaccessible fashion.

Choices are made in the architecture when determining what action the ACS will take. Both the explicit and implicit systems provide action suggestions which are “combined” using several methods to make a final action choice. The explicit memory store is made up of chunks similar to production rules, and are able to suggest external actions, working memory actions and goal actions. Inputs from the current goal, working memory, and sensory systems are matched to chunks to provide a similarity measure. The similarity measure provides a determination of how appropriate the rule is to the current situation. For all appropriate rules, a utility value is calculated that measures the difference between expected benefits and costs of the action. The utility is used to stochastically choose an action where the probability of selection action  $i$  is given by:

$$p(i) = \frac{e^{u_i/\tau}}{\sum_j e^{u_j/\tau}} \quad (\text{Sun 2003})$$

where  $\tau$  is an indicator of noise. This is the same rule used to determine choice probabilities of productions in ACT-R.

The implicit memory module uses separate networks to evaluate goal, working memory, and external actions. Using the same inputs as the explicit memory system, the networks generate a measure  $Q(a)$  of the quality of an action. The same choice axiom is used to suggest an action from the implicit memory system

$$p(a) = \frac{e^{Q(a)/\tau}}{\sum_i e^{Q(a_i)/\tau}}$$

Several methods can be used to determine what action to take given the two suggestions depending on the type of task being accomplished (Sun, 2004). One possible method is to stochastically decide which action to take. Another is to combine the utility and  $Q$  values across all possible actions and use this new value to make a choice. All suggested methods utilize the Luce's choice axiom to make a decision. It is unclear when it is appropriate to use one method over the other. CLARION is still fairly new and has not been tested extensively across many types of tasks the was the other architectures have.

The learning process in CLARION is unique and adds a new capability to cognitive architectures as a whole. Implicit memories are able to update weights using straight backpropagation when being trained on known input/output pairs, but it can also utilize reinforcement learning using a single numerical reward signal (Sun, 2006; Sutton & Barto, 1998). The implicit memory store can learn from interacting with the world. A form of bottom-up learning can identify actions suggested by the implicit networks and for explicit rules to use. Unlike, any other architectures, CLARION can allow an agent without any pre-programmed knowledge to not only learn to act in its environment according to some reward signal, but it can generate a “program” coded as explicit rules about what it has learned.

In addition, to bottom-up learning, CLARION can also take advantage of top-down learning processes. These top down learning processes update weights in the implicit memory to support the successful operation of rules in explicit memory. The more typical case for a CLARION agent is to begin operating based mostly using rules. However, the more it acts and uses those rules, the more that knowledge become encoded as implicit knowledge and the system becomes more reliant on the bottom level.

### **Models of Decision tasks**

We can get a sense of what decision making models look like when built in an architecture by looking at current models of decision heuristics, problem solving, and dynamic decision making within an architecture. For continuity, a single architecture will be explored. ACT-R has several features that make it best suited to serve as the exemplar architecture. First of all, it is the most popular in the psychological sciences

and its large body of contributors provides a nice selection of models to study. Second of all, it provides a nice middle ground between some of its competitors. It does not have the parsimony of Soar, but it is certainly less complex than CLARION. It is a hybrid system that has both symbolic and sub-symbolic mechanisms to manipulate. Also, the most recent versions have adapted EPIC's modular peripheral structure to expand it to include cognitive models involving sensory and motor operations.

Simple models of choice have been built in ACT-R that directly utilize the choice mechanisms built into the architecture. Both the production memory and the declarative memory provide such mechanisms. Both have been used to explain learning behaviors when making simple choices. One of the simplest models was built by Lovett (1998) to explain probability matching in a two alternative forced choice task where each alternative has a certain probability of being correct. If option A is the correct choice 80% of the time, then the optimal behavior would be to always choose A and be rewarded for 80% of the trials. However, people tend to show probability matching and choose A about 80% of the time. Lovett modeled this simple task with two productions CHOOSE-A and CHOOSE-B that both match the conditions at the beginning of a trial. No declarative knowledge is used, and so the choice is based solely on the learned utilities of the productions. This simple model is able to learn the probability matching at a rate similar to humans.

The declarative memory system can also be utilized to make decisions in an equally simple case. In a baseball simulation, players (batters) had to make a decision of how to swing based on whether they anticipated a fast ball or a slow ball from a pitcher (Lebiere, Gray, Salvucci, & West, 2003). The model had a single production instructing

the player to anticipate a pitch, while declarative memory consisted of two chunks: one for fast ball and one for slow ball. The activation of chunks in this case was dominated by the base-level activation which was highly influenced by past experience. For this reason, the model was able to quickly learn to anticipate a pitch based on the previous few pitches observed. When looking at the mean temporal error based on the previous one, two, or three pitches, the model closely matches data collected from human baseball players. A variation of that model was adapted to the trinary choice in a competitive rock-paper scissors game (West, Lebiere, & Bothell, 2006). Again, the simple model was able to match qualitative data regarding the choices of human subjects<sup>2</sup>.

A more complex decision process was used by Altmann and Burns (2005) to explain streak biases when trying to predict successive coin flips. They used three productions that were possible valid at the beginning of a trial: GUESS-HEADS, GUESS-TAILS and HEURSTIC-GUESS. If the HEURSTIC-GUESS production becomes active the GUESS-HEADS and GUESS-TAILS get locked out and a series of secondary productions are used to make decisions according to a heuristic rule. The rules states that if a streak (continuous heads or continuous tail) with length one greater than the current streak can be recalled, then guess against the streak. If a streak of greater

---

<sup>2</sup> For these simple models, no guidelines have been set to determine when a choice of actions should be dependent on procedural learning or declarative learning. Although it is clear that procedural utilities are sensitive to raw percentages and declarative activations are sensitive to temporal aspects of experience, how does one know to store information in a particular fashion. It is not in the scope of this article to answer such a question, but it is important to bring up because this is the first of many question that come up when evaluating models of behavior developed within cognitive architectures. How much does the success of a model depend on the design of the architecture versus the design of the model.

An architecture with very little restrictions can be capable of universal computation. This means that it would be capable of reproducing any data set given the right model. An architectures limitations are really what help it stand out as a respectable model of cognition. Unfortunately, most publications regarding models built within architectures focus on how they match the data, regardless of whether or not the model utilizes the strengths and weaknesses of a particular architecture so as to conform to some psychological theory.

length cannot be recalled then guess with the streak (it is assumed that a change in the environment has made this particular value more likely). Human data shows a quadratic pattern of streak biases depending on streak length. The learned production utilities account for the overall probability matching of guessing heads versus tails. The heuristic leads to an accurate prediction of how people respond to streak biases. The heuristic choice is a good representation of how a choice process can be modeled within the ACT-R system.

One of the best examples of generating a unique decision strategy within the framework of ACT-R was performed by Gonzalez, Lerch & Lebiere (2003). Instead of using the built in utility mechanism, past decisions are stored as chunks in situation, decision, utility (SDU) triplets. The decision process evaluates these triplets one at a time and after each evaluation, procedures decide with some probability to use the decision with the best utility so far or to select another alternative to evaluate. The probability of continuing the search is a function of the time taken to make a decision. This represents a model of choice where a person really has to *think* about what they are to do. Choices using built in ACT-R mechanisms happen very quickly (Anderson & Lebiere, 1998), and apply to situations where quick action is needed (i.e. baseball). When one has a chance to deliberate, other mechanisms that allow you to recall knowledge about the situation are more appropriate. ACT-R's symbolic nature makes it fairly easy to implement certain heuristic choice rules. However, the preferential choice models discussed previously have been quite successful at describing choices between alternatives that can be described by attributes that are differentiated along a continuous scale.

### **The best of both worlds**

How might we incorporate concepts used in the newer dynamic models of judgment and decision making into ACT-R? Both the MDFT and LCA models are very similar in their operation, but we shall look at the details of the MDFT model to see how its decision mechanism may be incorporated into a cognitive architecture.

We will see that ACT-R allows three different levels at which we can insert comparable mechanisms, each with their own strengths and weaknesses. The first would be to insert an MDFT mechanism within the architecture itself. This could be at either the procedural or at the declarative memory level. This would depend on whether a choice is interpreted as a choice between actions, or a choice between knowledge. It is beyond the scope of this article to argue which is more appropriate. The elements that currently are used to calculate procedural utility (past successes and failure) and declarative activations (recency of use, associations to goal) could be utilized to determine the valuations of attributes for each choice in the MDFT. The benefit of this change would be that decision tasks could be simply encoded such that individual procedures or chunks could represent the choice to be made. However, this would come at a great cost. This would involve drastically changing the underlying architecture of ACT-R. This may have a large impact on successes that have already been established within this framework. In fact, incorporating such a mechanism at the level of the architecture would essentially require the development of a new architecture.

A second level at which we could incorporate MDFT into ACT-R is at the model level. We might follow the lead of Bettman, Johnson, & Payne (1990) who reduced decision rules to a set of elementary information processes (EIPs) which were to be

actions used in a production system. The rules they suggested would need to be broken down into even lower level elementary units to be implemented in a purely symbolic system. The EIPs representing operations such as ADD and PRODUCT do not have any comparable operation in a symbolic system like ACT-R. However, this does suggest a hierarchical approach to implement decision models. In addition, these EIPs represent versatile productions that could be utilized in multiple cognitive processes including other decision strategies.

Because we can conceptualize MDFT in several ways, we could build such a model using several designs in ACT-R. Purely symbolic systems attach no predetermined meaning or value to any symbols, so it is specifically the *judgments* part of the decision making process that would be difficult to represent. Specific preference levels would need to be coded symbolically as chunks and switching between them would be handled by procedures. This type of modeling might be appropriate for modeling a single choice task, but it has some obvious problems. If we are utilizing the symbolic level of ACT-R to code preference levels between  $n$  alternatives, each set of valid preference values would need to be stored in declarative memory. A set of  $n$  preferences would be valid if all values sum to zero and the maximum preference is equal or less than the decision threshold. This type of representation is just not feasible considering the many permutations of choices we make each and every day (See appendix B for an exploration of how such a system might be coded). It seems that decision making is a fundamental element of cognition and should not be recoded for every particular decision that needs to be made.

This brings us to the third level at which we could implement a dynamic decision making strategy. As of ACT-R 5.0 (Anderson, Bothell, Byrne, Douglass, Lebiere, & Qin, 2004) the architecture has been designed as a modular system connecting declarative memory, sensory systems, and motor systems, to a central cognitive core. All modules communicate with the cognitive core (where procedural memory is stored) by passing chunks of information through buffers. A decision module could be created to handle certain decision making problems. At first, it may seem inappropriate to place an obviously cognitive function outside of the cognitive core. However, this level of modularity works well with many of the current standards for encoding strategies in problem solving and heuristics in decision making (Nellen, 2003). A cognitive process at the procedural level may evaluate goals and known knowledge to decide when there is an obvious heuristic available, but may pass the choice off to the decision module when the individual attributes need to be weighed. Also, seeing as how MDFT has been used to independently account for reaction times in decision tasks, there would be no need to adapt timing mechanisms to match ACT-R's established timing mechanisms.

### **Final Remarks**

In his call for unified theories of cognition, Allen Newell (1990) suggested that such a theory could take on many forms. Symbolic systems have been very successfully in this arena for two reasons. First, the roots of these symbolic knowledge systems have a long history deeply entwined with our understanding of logic systems. The artificial intelligence community greatly explored how symbolic logic can act 'intelligently' in an environment. We have not committed as many resources to understanding the general

application of neural networks, dynamic systems, or random walk models to many problem solving domains. The focus of these efforts may be highly influenced by the second reason for the success of symbol systems. Even though we know we live in a world with a lot of gray, we interact with it and with each other as if it were black and white. We comment on whether the weather is hot or cold without knowing when one turns to the other. A simple search for images a 'chairs' on the internet will show a wide spectrum of forms with many functions, yet we understand how each is a chair. We use a language and often think in terms of symbolic representations. For the mere fact that we need to provide input to a cognitive architecture and we would like to understand its behavior, symbol systems will have a role to play. We will be able to rely on a cognitive system without symbolic representations until researchers make the commitment to an artificial agent similar to the commitment made to our best example of intelligent neural networks. This commitment would include spending hours a day for several years before it begins to make coherent responses, providing care and instruction for another decade until the agent understands basic social concepts, spending another five years hoping it does not get mixed up with the wrong crowd, and finally letting it be completely autonomous only to see it on holidays. Until then, we will need to inject the knowledge of our world into artificial agents in a way we understand. New cognitive architectures such as CLARION try to take advantage of both a symbolic system and a neural network by closely integrating them into a coherent architecture. This type of integration is necessary for the next generation of systems attempting to implement a unified theory of cognition.



References

- Altmann, E. M., & Burns, B.D. (2005). Streak biases in decision making: data and a memory model. *Cognitive Systems Research*, 6(1), 5-16.
- Anderson, J. R. (1993). *Rules of the Mind*. Hillsdale, NJ: Erlbaum.
- Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of mind. *Psychological Review*, 111, 1036–1060.
- Anderson, J. R. & Lebiere, C. (1998). *The atomic components of thought*. Mahwah, NJ: Erlbaum.
- Belavkin, R. V. (2006). Towards a Theory of Decision-Making without Paradoxes. In D. Fum, F. D. Missier & A. Stocco (Eds.) *Proceedings of the Seventh International Conference on Cognitive Modeling* (pp. 38-43).
- Gonzalez, C., Lerch, F. J., & Lebiere, C. (2003). Instance-based learning in dynamic decision making. *Cognitive Science*, 27, 591–635.
- Just, M. A. & Carpenter, P. A. (1992). A capacity theory of comprehension: individual differences in working memory. *Psychological Review*, 1, 122-149.
- Kahneman, D., & Tversky, A. (1979). Prospect Theory: An Analysis of Decision Under Risk. *Econometrica*, 47, 263-291.
- Laird, J. E., & Congdon, C. B. (2006). *The Soar User's Manual Version 8.6*: University of Michigan.
- Laird, J. E., Newell, A., & Rosenbloom, P. S. (1987). SOAR: An architecture for general intelligence. *Artificial Intelligence*, 33, 1-64.
- Lebiere, C., Gray, R., Salvucci, D., & West, R. (2003). Choice and learning under uncertainty: A case study in baseball batting. In *Proceedings of the 25th Annual*

*Conference of the Cognitive Science Society* (pp. 704–709). Mahwah, NJ: Lawrence Erlbaum Associates, Inc.

Lovett, M. C. (1998). Choice. In J. R. Anderson, & C. Lebiere (Eds.). (pp. 255-296). *The atomic components of thought*. Mahwah, NJ: Erlbaum.

Luce, R. D. (1959) *Individual choice behaviour: A theoretical analysis*. New York: Wiley.

Meyer, D. E. & Kieras, D. E. (1997). A computational theory of executive cognitive

processes and multiple-task performance: I. Basic mechanisms. *Psychological Review*, 104(1), 3-65.

Nellen, S. (2003). The use of the “take-the-best” heuristic under different conditions, modeled with ACT-R. In F. Detje, D. Dörner, & H. Schaub (Eds.), *Proceedings of the fifth international conference on cognitive modeling* (pp. 171–176). Germany: Universitätsverlag Bamberg.

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

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

Roe, R. M., Busemeyer, J. R., & Townsend, J. T. (2001). Multi-alternative decision field theory: A dynamic connectionist model of decision-making. *Psychological Review*, 108, 370-392.

Savage, L. J. (1954). *Foundations of Statistics*. Oxford: Wiley.

- Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning : an introduction*. Cambridge, MA: MIT Press.
- Sun, R. (2003). *A Tutorial on CLARION 5.0*.  
<http://www.cogsci.rpi.edu/~rsun/sun.tutorial.pdf>
- Sun, R. (2004) Desiderata for cognitive architectures. *Philosophical Psychology*, 17(3), 341-373.
- Sun, R. (2006a). Prolegomena to Integrating Cognitive Modeling and Social Simulation. In R. Sun (Ed.), *Cognition and Multi-Agent Interaction: From Cognitive Modeling to Social Simulation* (pp. 3-26). New York: Cambridge University.
- Sun, R. (2006b). The CLARION Cognitive Architecture: Extending Cognitive Modeling to Social Simulation. In R. Sun (Ed.), *Cognition and Multi-Agent Interaction: From Cognitive Modeling to Social Simulation* (pp. 79-99). New York: Cambridge University.
- Taatgen, N. A. (1999). *Learning without limits: From problem solving towards a unified theory of learning*. Doctoral Dissertation, Department of Artificial Intelligence, University of Groningen, Groningen, the Netherlands.
- Taatgen, N. A., Lebiere, C., & Anderson, J. (2006). Modeling Paradigms in ACT-R. In R. Sun (Ed.), *Cognition and Multi-Agent Interaction: From Cognitive Modeling to Social Simulation* (pp. 29-52). New York: Cambridge University.
- Tversky, A. (1972). Elimination by aspects: A theory of choice. *Psychological Review*, 79(4), 281-299.
- Usher, M., & McClelland, J. L. (2004). Loss aversion and inhibition in dynamical models of multialternative choice. *Psychological Review*, 111(3), 757-769.

von Neumann, J., & Morgenstern, O. (1947) *Theory of Games and Economic Behavior*. Princeton, NJ: Princeton University Press.

West, R. L., Lebiere, C., & Bothell, D. J. (2006). Cognitive Architectures, Game Playing, and Human Evolution. In R. Sun (Ed.), *Cognition and Multi-Agent Interaction: From Cognitive Modeling to Social Simulation* (pp. 103-123). New York: Cambridge University.

Appendix A

The following is a representation of an ACT-R model for adding the numbers 5 and 2. Since this essay is not concerned with the particular syntax of the ACT-R system, productions are given in a more readable fashion.

Declarative Memory:

Chunks are defined as having a type, a name, and a list of labeled arguments. There are two different types of chunks in declarative memory.

1) count-order chunks

name	first	second
A	0	1
B	1	2
C	2	3
D	3	4
E	4	5
F	5	6
G	6	7
H	7	8
I	8	9
J	9	10

2) add chunks

name	Arg1	Arg2
goal	5	2

**INITIALIZE-ADDITION**

IF the goal is to add a pair of numbers  
 but the sum slot is **nil**  
 THEN set the sum slot of the goal equal to the "arg1" value  
 and set the count slot to 0  
 and request a retrieval of the a count-order chunk where  
 the value of the argument "first" matches the "arg1" value  
 in the goal.

**TERMINATE-ADDITION**

IF the goal is to add a pair of numbers  
 and the count slot matches the "arg2" value  
 THEN set the count slot of the goal to **nil**

**INCREMENT-SUM**

IF the goal is to add a pair of numbers  
and the count slot does not match the "arg2" value  
and a count-order chunk is in the buffer where the  
"first" value matches the sum slot  
THEN set the sum slot of the goal equal to the "second"  
value from the buffered chunk  
and set the count slot to 0  
and request a retrieval of the a count-order chunk where  
the value of the argument "first" matches the count slot in  
the goal.

**INCREMENT-COUNT**

IF the goal is to add a pair of numbers  
and the count slot matches the sum slot  
and a count-order chunk is in the buffer where the  
"first" value matches the count slot  
THEN set the count slot of the goal equal to the "second"  
value from the buffered chunk  
and request a retrieval of the a count-order chunk where  
the value of the argument "first" matches the sum slot in  
the goal.

## Appendix B

There are several approaches that can be taken in trying to implement a dynamic computational model of decision making using production rules in the ACT-R system. The following will outline 3 general approaches and discuss the benefits and weaknesses of each method.

## Method 1

The first method operates as a Markov chain. The probabilities for changing states can exist in either the utilities of declarative statements or in the utilities of procedural rules. In ACT-R, the declarative memory modules is supposed to represent explicit knowledge that can be consciously recalled. Dynamic process models of decision making are, however, representative of implicit processes that occur at a less than conscious level. For this reason, it makes more sense to utilize the utilities of procedures. Because the probability of shifting in a particular direction along the chain is dependant on one's position in the chain, a different procedure is required to represent every transition. In a binary choice situation using a Markov chain of length 20, we will need 40 separate procedures. The Markov model becomes quite unruly when scaling up beyond binary choices. The following represents a system relying on a chain of length 5.

**INCREMENT-CHOICE-A**

```
IF the goal is to decide between alternatives X and Y
  and the preference slot is -1
THEN set the preference slot of the goal equal 0
```

**DECREMENT-CHOICE-A**

```
IF the goal is to decide between alternatives X and Y
  and the preference slot is -1
THEN set the preference slot of the goal equal -2
```

**INCREMENT-CHOICE-B**

```
IF the goal is to decide between alternatives X and Y
  and the preference slot is 0
THEN set the preference slot of the goal equal 1
```

**DECREMENT-CHOICE-B**

```
IF the goal is to decide between alternatives X and Y
  and the preference slot is 0
THEN set the preference slot of the goal equal -1
```

**INCREMENT-CHOICE-C**

```
IF the goal is to decide between alternatives X and Y
  and the preference slot is 1
THEN set the preference slot of the goal equal 2
```

**DECREMENT-CHOICE-C**

```
IF the goal is to decide between alternatives X and Y
  and the preference slot is 1
```

THEN set the preference slot of the goal equal 0

**SELECT-ALTERNATIVE-X**

IF the goal is to decide between alternatives X and Y  
and the preference slot is 2  
THEN choose X

**SELECT-ALTERNATIVE-Y**

IF the goal is to decide between alternatives X and Y  
and the preference slot is -2  
THEN choose Y

This method depends on the utilities remaining static. If ACT-R is set to allow learning, then the utilities will change depending on the successes and failures of the productions. This predicts that the decisions would get faster the more often this exact choice is made. This method clearly has problems when trying to scale to more complex choice situations. For one, each set of productions would need to be repeated for every possible decision to be made. It seems as if the many procedures should be generalized into a single process. Unfortunately the only stochastic processes in ACT-R are the conflict resolution systems in procedural and declarative memory structures. One solution would be to abandon the Markov chain representation and rely on an attention mechanism to support probabilities.

**Method 2 –**

In order to capture the acceleration of preference changes near the threshold, we can no longer rely on separate probabilities. In the DFT model, this acceleration comes from larger shifts in preference depending on the distance from threshold. This is not a problem for a system that easily handles real numbers. Unfortunately, due to the discrete, symbolic nature of ACT-R, each possible preference values must be explicitly represented. We run into a similar problem with representations that we had in Method 1. Here we have to code a detailed mathematical system within ACT-R. That may be appropriate to explain how humans cognitively process numbers and mathematical objects, but it is not an appropriate way to develop a mathematical model of underlying cognitive sub-processes.