2005

Thinking as a Production System

Marsha C. Lovett
*Carnegie Mellon University, lovett@cmu.edu*

John R. Anderson
*Carnegie Mellon University, ja@cmu.edu*

Follow this and additional works at: [http://repository.cmu.edu/psychology](http://repository.cmu.edu/psychology)

*Part of the Psychology Commons*

Published In


This Book Chapter is brought to you for free and open access by the Dietrich College of Humanities and Social Sciences at Research Showcase @ CMU. It has been accepted for inclusion in Department of Psychology by an authorized administrator of Research Showcase @ CMU. For more information, please contact research-showcase@andrew.cmu.edu.
Thinking as a Production System

Marsha C. Lovett and John R. Anderson

Corresponding author address:

Marsha C. Lovett
Department of Psychology
Carnegie Mellon University
Pittsburgh, PA 15213-3890
412-268-3499
Thinking as a Production System

Since their birth (ca. late 1960s), production systems have been developed as a formal tool not only for describing but for explaining how humans think. Indeed, “to advance our understanding of how humans think” is the stated goal of Newell and Simon’s classic (1972) book *Human Problem Solving*, where the first body of work on production-system models of human thought was presented. The main goal for production systems in psychological research has changed little in the intervening years, and yet the state of the art has advanced dramatically. The aim of the current chapter is to present a contemporary production-systems approach to open questions in problem solving, reasoning, analogy, and language. We will highlight the ways in which today’s production systems allow for more flexibility, stochasticity, and sensitivity than their predecessors. Besides demonstrating that production systems can offer insight into current questions and add to our understanding of human thinking, we will discuss our view of production systems in future research.

Background on Production Systems

A production system is a set of *production rules* – each of which represents a contingency for action – and a set of mechanisms for matching and applying production rules. Because the production rule is the fundamental unit of this formalism, it is worth giving a few examples. Table 1 presents four sample production rules written in English. Note that each is divided into two parts by the word “then”: the first part of each production rule (before the “then”) specifies the conditions under which that production rule is applicable, and the second part specifies the actions to be applied. Conditions may reflect an aspect of the external world (e.g., it is dark) or an internal, mental state (e.g., my current goal is to get to a particular location, or I can retrieve a
particular fact). Likewise, actions may transform a feature in the real world (e.g., flip the light switch) or an internal, mental state (e.g., change my current goal, or add a fact to memory).

To operate, a production system requires a *dynamic memory* that represents the current state of the system and is used to match against production rules’ conditions. For example, when dynamic memory includes the goal “to get to San Francisco,” the second production rule in Table 1 would match for someone in Pittsburgh, Pennsylvania. This *pattern matching* of production rules to dynamic memory leads to a set of potentially applicable production rules, called the *conflict set*. However, not all production rules in the conflict set are applied. The process of *conflict resolution* specifies which production rules from the conflict set will *execute* their actions or *fire*. These actions are likely to change the external and/or internal state of the system, reflected in a change to dynamic memory. Then, a potentially different set of production rules may comprise the conflict set, and the cycle continues.

One way to view how production rules operate is by analogy to stimulus-response associations, i.e., when a particular stimulus is present, an associated response is triggered. This fits with the notion that a production rule is not directly verbalizable but rather observable through behavior. This analogy to stimulus-response associations emphasizes the fact that production systems do not operate via a homunculus interpreting production rules as programming code. Instead, each production rule – when it matches dynamic memory – has the potential to fire and change the current state, thus setting other production rules into action.
This discussion leads to the question of what it means to model thinking as a production system: What are the theoretical implications associated with representing knowledge as production rules? Below are four features commonly attributed to production-rule representations.

(1) *Production rules are modular.* Each production rule represents a well-circumscribed unit of knowledge such that any production rule can be added, refined or deleted independently of other production rules in the system. Moreover, each production rule is atomic such that it would be added, refined, and deleted as whole unit. It is important to note, however, that this modularity does *not* preclude production rules from interacting with each other extensively in a running system. Indeed, adding a new production rule to an existing set can — and often does — completely change the functioning of the system because of the way production rules’ actions impact each others’ firing. Early production-system modelers (Klahr & Wallace, 1976; Young and O’Shea, 1981) took advantage of this feature by adding or deleting production rules to explicitly test how that change would impact the system’s behavior. More recently, production systems have been developed with autonomous learning mechanisms that enable the system’s production rules to change based on experience. In these systems, modularity is achieved because these learning mechanisms create and modify individual production rules independently of other rules.

(2) *Production rules are asymmetric.* Each production rule is a unidirectional contingency for action. This means that the production rule “When I want to type the letter ‘j’, then I punch my right index finger” is different from “When I punch my right index finger, then I type the letter ‘j’”. Moreover, asymmetry and modularity imply that, if these two production rules were in the same system, adding, deleting or refining the former would not directly change the latter. That is, practicing typing would exercise the first production rule, strengthening the
index-finger response when “j” is the desired letter, but it would not strengthen one’s knowledge that “j” appears when touch-typing with that finger. For expert touch-typists, this asymmetry is quite noticeable: Without looking at a keyboard, try to identify the letter that is typed with your left index finger. Tough, isn’t it? Typing the word “frog” would have been easier. Such asymmetry has been documented in many contexts (see Singley & Anderson, 1989, for a review).

(3) *Production rules can be abstract.* That is, production rules allow for generalization because their conditions may be represented as templates that match to a wide range of patterns. These conditions specify the relationship(s) between items without specifying the items themselves (e.g. “When A is taller than B and B is taller than C, then say A is taller than C” is true for any values of A, B, and C). The capability to represent abstract relationships allows for transfer of learning across different situations as long as they fit within the conditions of the given production rule. For example, the first production rule in Table 1 could match to a dark dining room, living room, or office, meaning that experience at flipping the light switch in any of these rooms would transfer to the others. Likewise, the third production rule in Table 1 could match to any two-addend addition problem.

(4) *Production rules are not directly verbalizable.* This feature is based on the notion that each production rule represents knowledge about a contingency for action that is not directly accessible to verbalization. A good example of this occurs when someone knows how to drive a standard transmission car but cannot explain it verbally. It is important to note that, while this feature implies that knowledge represented in production-rule form cannot be accessed directly, it does not imply that one cannot use other techniques to talk about performance knowledge. For example, when changing gears in a standard transmission car, it
is possible to observe one’s own performance and verbally describe these observations.

Also, knowledge about how to perform a task may be represented in multiple forms, some verballizeable and some not.

This last point confronts a common misconception about production systems, namely, that knowledge about rules or procedures is necessarily represented as production rules. While knowledge about rules and procedures can be represented in production-rule form, it is not the content of knowledge that determines how it is represented. Instead, the four features listed above serve as a set of necessary conditions for knowledge to be considered represented in production-rule form. To illustrate the distinction between knowledge contents and representational form, Table 2 shows that the same knowledge content (either column) can be represented in a production-rule form (top entry) or not (bottom entry as a declarative fact). So, when considering what it means for knowledge to be represented in production-rule form, the key is not in what knowledge is being represented but rather in how.

Production Systems, Then and Now

The first production systems set out to establish a mechanistic account of how human adults perform relatively short, moderately difficult, symbolic tasks (Newell & Simon, 1972). Besides demonstrating that production systems could solve these tasks, the main goal was to connect the system’s processing steps to human problem-solving steps. Several features distinguish these early production systems from their current-day progeny. First, early production systems tended to focus on demonstrating human-like performance; current models rely heavily on learning mechanisms to derive predictions about learning and performance across time. Second, early
models focused on reproducing qualitatively the processing steps of individual problem solvers, whereas more recent models have been submitted to both quantitative analyses of fit to aggregate data (e.g., average reaction times for various conditions) and qualitative analyses (e.g., whether the model demonstrates the same errors as people). Third, the role of noise processes has drastically increased from early models that avoided stochastic processes completely to current day models where stochasticity plays an important role (Lebiere, Anderson, & Bothell, 2002; Lebiere, Gray, Salvucci, & West, 2003). Fourth, early models focused on the “cognitive” layer of processing and eschewed integrating receptors and effectors into models. In contrast, current production systems incorporate and emphasize perception and action in their frameworks (Anderson & Lebiere, 1998; Meyer & Kieras, 1997). Finally, the fifth feature that distinguishes early and recent production systems is so strongly linked to the early models, it has sometimes been considered a defining feature of production systems. This is the symbolic nature of early production systems. However, almost all modern production systems take hybrid view, by positing symbolic representations as important conceptual units and acknowledging graded representations as a valuable additional layer (e.g. associating continuously valued quantities with each production rule).

Current Production Systems in Context

This section provides a brief overview of four production systems currently being used in a variety of cognitive modeling situations. The systems to be described are ACT-R (Anderson & Lebiere, 1998), EPIC (Meyer & Kieras, 1997), Soar (Laird, Newell, & Rosenbloom, 1991), and 4-CAPS (Just, Carpenter, & Varma, 1999). ACT-R emphasizes the notion of a cognitive modeling architecture, in which the same set of mechanisms and representational schemes are used to capture human learning and performance across tasks. Recently, this has been extended
To map various ACT-R mechanisms and modules to particular brain regions for comparison with neuroimaging data. EPIC has focused on capturing the connections between the cognitive, perceptual and motor systems. Recently, EPIC has been used to make quantitative predictions about perception-to-action loops in multiple-task situations and across the adult age span. Soar was originally developed to address issues in both psychology and artificial intelligence. Recently, it has been particularly successful in simulating multi-agent, dynamic interactions with real-world application (e.g., Jones, Laird, Nielsen, Kenney, & Koss, 1999). The 4-CAPS architecture, like its predecessor 3-CAPS (Just & Carpenter, 1992), focuses on individual differences.

To delineate the space of current production systems, we next highlight the dimensions along which these systems differ. First, they differ with regard to their degree of processing parallelism. Toward one end of the spectrum, ACT-R posits that only a single production rule can fire at a time. However, ACT-R allows for parallelism in other ways: asynchronous parallelism among its perceptual and motor modules; parallel retrieval of information from declarative memory; and parallel production-rule matching and selection. Soar similarly posits serial processing in that a single operator is chosen in each decision phase, but this is preceded by an elaboration phase that allows parallel production firing. 4-CAPS allows parallel firing of production rules for all cycles, but this parallelism is subject to a capacity limitation such that the more production rules firing, the less rapidly each of them is executed. EPIC is the only system with fully parallel production-rule firing. To manage its multiply threaded central cognition, EPIC uses task-coordination strategies that impose ordering constraints when necessary.
Another dimension along which the systems differ is the degree of modularity they propose. Soar is at one end of this spectrum because of its unitary structure – a single set of production rules representing long-term memory. 4-CAPS posits a number of distinct sets of production rules, connected to each other. In ACT-R and EPIC, multiple modules correspond to separate perceptual and motor modalities and to “central cognition.” These modules are considered encapsulated, independent processors, with their interactions handled by the production system.

While all four systems produce quantitative predictions that match well to performance data, ACT-R and Soar have particularly focused on production-rule learning as well. Yet another dimension on which these architectures differ is their commitment to hybridization, with Soar committed to a purely symbolic account while ACT-R and 4-CAPS postulate continuously varying quantities that drive the processing of symbolic units. EPIC does have continuously varying parameters associated with various modules but does not appear to have information-laden nonsymbolic quantities in its theory of central cognition.

Finally, production systems differ in the role that noise processes play in their processing. In Soar, their role is minimal (i.e., when a “tie” between production rules arises, one of them is chosen at random). In ACT-R and 4-CAPS, noise processes are assumed added to the various continuously varying computations that influence system performance. In EPIC, noise is used more to represent variability in system parameters (e.g., rate parameter in Fitt’s Law governing motor movements) than to represent a generic non-determinism of the system.

Organization of the Remainder of the Chapter
Our own research has involved the ACT-R system and slight variants. Below we describe six ACT-R models, with which we are familiar, that address different aspects of cognition. We will not focus on the ACT-R details of these models but rather on how they illustrate the general trends in production-system models towards softer, more flexible, and highly detailed characterizations of human cognition. We will try to place each model in the multi-dimensional space described above by highlighting the following features: Does the model include both performance and learning mechanisms? Does the model make use of symbolic (rule-based) and other continuously varying computations? Does the model draw upon multiple processing modules beyond a central production-rule memory? We will use the template in Table 3 to summarize how each model fits into this 3-dimensional space in terms of its use of various ACT-R representations and mechanisms. In addition, we will comment on how each model makes use of parallelism and noise processes, as appropriate.

It is worth noting that, in this chapter, the term “subsymbolic” refers to the numerical values and computations associated with each symbolic unit. In this sense, the prefix “sub” refers to a level of description below the symbolic units and that determines those symbolic units’ access in competition with other symbols. The use of the term “subsymbolic” from a connectionist perspective often refers to the fact that symbols may be represented in a distributed fashion, with the prefix “sub” referring to the pieces of the pattern that comprise a symbol. For instance, Smolensky (1988) writes “The name subsymbolic paradigm is intended to suggest cognitive descriptions built up of entities that correspond to constituents of the symbols used in the symbolic paradigm; these fine-grained constituents could be called subsymbols, and they are the activities of individual processing units in connectionist networks” (p. 3). It is an interesting question whether these two views are really in contradiction. The subsymbolic values discussed
in this chapter are updated and used only locally while at the same time having a global impact on the system’s processing like activations of units in a connectionist system. As an example of this, consider the utility values associated with production rules: When multiple production rules match the current situation, the one with the highest utility value succeeds in firing. This competition occurs among the individual units themselves without any explicit selection by a controlling homunculus and without any conscious access to the utility values. Another important kind of numerical quantity in our subsymbolic representation are similarities between symbols. With these quantities, a production rule can partially match against a symbol similar to what is specified in its condition, allowing the system to realize soft constraints. This fact further blurs the difference between the two senses of subsymbolic. Work exploring a connectionist implementation of the ACT-R architecture (Lebiere & Anderson, 1993) suggests that symbolic units represented in a distributed fashion can yield the behavior of a symbolic system that has continuously valued quantities influencing the access and use of its symbols.
Choice

One of the perennial questions in problem-solving research involves how solvers make choices: choices of the next step, of an appropriate solution strategy, of whether to use weak (domain-general) vs. strong (domain-specific) methods. Indeed, around the time when production systems were first developed, Newell and Simon introduced the idea that the very process of problem solving could be viewed as search in a problem space, which equates problem solving with a series of choices. Research then addressed the question “How do solvers make choices?” by focusing on cases where solvers have little or no domain knowledge. Production-rule models representing various problem-solving heuristics predicted performance and established links between heuristics and human data. Current research asks “How do solvers make choices?” but focuses on cases where solvers have prior, relevant experience. This is, at its heart, a question about learning, so production systems that learn from their experience may offer additional insight.

In a set of studies by Lovett (Lovett & Anderson, 1996; Lovett, 1998), participants’ choice learning in the Building Sticks Task (BST) was studied and modeled within ACT-R. The BST is an isomorph of the water jars task (Luchins, 1942) such that, in each problem, solvers must add and subtract the lengths of three building sticks to equal the length of a goal stick (see top of Figure 1). Solvers face a choice between two strategies (see bottom row of Figure 1): overshoot, which involves starting with the longest building stick and shortening it by the others, and undershoot, which involves starting with the short or medium building stick and then lengthening it. In these studies, participants encountered the BST for the first time and solved a sequence of problems in which the proportion of problems that could be solved by each of the
two strategies was manipulated (e.g., 30% overshoot-only problems and 70% undershoot-only problems or vice versa). The results can be summarized in three main findings:

1. Participants’ choices initially followed a hill-climbing heuristic, with little bias toward undershoot or overshoot.
2. With experience, participants gradually learned to prefer the more successful strategy for their condition.
3. Changes in strategy choice were sensitive to recent experiences in that participants were more likely to choose the strategy that had been successful on the previous (or even second previous) problem.

The model that was built for this task has since been applied in various forms to account for choice learning in several other tasks (see Lovett, 1998). Here we describe the BST model specifically. The model was initially endowed with production rules that implement two domain-general heuristics, hill-climbing and guessing, for the particulars of this task. For example, the guess-overshoot production rule makes the first overshoot move regardless of the details of the problem, and guess-undershoot does this for undershoot. These productions represent an uninformed guess that their action will lead to a solution and match to any BST problem. In addition, the hillclimb-overshoot production makes the first overshoot move but only matches when this move takes the initial state closest to the goal state; hillclimb-undershoot does the same for undershoot. These productions represent knowledge for taking the action that looks best according to a hill-climbing metric. Note that three of these four production rules will match to each BST problem’s initial state: both guess production rules and one hillclimb production rule – whichever one matches the stick lengths of the problem (e.g., hillclimb-undershoot in Figure 1).
Note also that, while three production rules match to an initial “stimulus”, two of them produce the same “response” but on the basis of different knowledge (i.e., two separate production rules). This emphasizes the fact that production rules are not simply stimulus-response associations but represent additional information in their conditions, which defines the (potentially different) scopes over which they apply.

Beyond the task-specific composition of its production rules, this model’s most important features come from ACT-R’s general, subsymbolic computations for production-rule utility values. Each production rule has an associated utility – learned by experience – that represents a combined estimate of how successful and costly that production rule is likely to be. Whenever the model is faced with multiple matching production rules, there is a noisy selection process that fires the production rule with the highest subsymbolic utility value. This noise process serves to lead the model generally to choose the production rule that has been most useful in past experience but to do so a proportion of the time consistent with that production rule’s utility relative to the competing production rules’ utility (e.g., competing production rules with very close utility values are selected virtually at random). These utility values are learned from experience, according to a pre-specified mechanism: Specifically, each production rule’s utility is computed arithmetically as a time-weighted average of its past success rate combined with a (negated) time-weighted average of its past costs (where cost is measured in time the production rule “spends” when fired).

In the case of the BST model, learned utility values “average in” new experiences of success and failure across trials, thus allowing the model to gradually increase the utility value for production
rules that have had greater success (and lower cost) and hence to gradually increase the
likelihood of firing more useful production rules. Thus, the model shows the same gradual
preference for the more successful BST strategy as do participants. In addition, because this
updating mechanism includes a time-weighted decay, the impact of recent successes (and costs)
on a production rule’s overall utility value is greater, thus leading the model – like participants –
to change strategy choice with greater sensitivity to recent experiences.

Summary
This production-system model of problem-solving choice specifies a set of fairly generic
production rules to represent the heuristics of guessing and hill-climbing and then draws on
ACT-R’s pre-existing production-rule mechanisms to learn to solve problems by experience. The
major claim, then, is that strategy-choice learning is strongly guided by problem-solving
experiences of success and cost associated with using those strategies, and that strategies are
effectively represented as production rules. More specifically, the model posits that choices in
problem solving are governed by an implicit competition among production rules based on their
utility values (a subsymbolic performance mechanism) and that these utilities are updated
naturally based on experience (a subsymbolic learning mechanism). The corresponding two
subsymbolic, production-rule cells have checks in Table 4. Although this model does not address
how production rules specific to this task are acquired (i.e., there is no symbolic production-rule
learning), its initial production rule set is mainly composed of general heuristics that have only
been adapted slightly to the context of the particular task. (That is, for this relatively knowledge-
lean task, it is reasonable to suspect that participants and the model can get by without acquiring
a lot of new, task-specific production rules.) It is also interesting that, in this task, production
rules – with their somewhat broad conditions of applicability – largely determine the behavior of the system; while declarative knowledge is involved, it is not involved critically in the explanation of the phenomena. This representational bias is supported by the relatively broad, within-task transfer that problem solvers show in carrying over their strategic preferences from trained BST problems to novel BST problems.
Analogy

Analogy, the process of finding and using correspondences between concepts, plays a fundamental and ubiquitous role in human cognition (see Holyoak, this volume). From mathematical problem solving (Novick & Holyoak, 1991) to computer programming (Anderson & Thompson, 1989) to creative discovery (Holyoak & Thagard, 1995), analogy facilitates better understanding of old knowledge and the formation and inference of new knowledge. The critical step in analogy is finding a mapping from objects and relations in the source or “known” domain (where pre-existing knowledge forms the base of the analogy) to objects and relations in the target or “novel” domain (where knowledge from the source domain will be applied). Numerous researchers have proposed theories that describe how analogical mapping takes place (Gentner, 1983, 1989; Hofstadter & Mitchell, 1994; Holyoak & Thagard, 1989; Hummel & Holyoak, 1997; Keane, Ledgeway, & Duff, 1994; Kokinov, 1998). A common feature of these theories is that they require a mixture of symbolic and subsymbolic processes. The symbolic processes are required to reason about the structure of the domains, but the softness of subsymbolic processes is required to stretch the analogy in semantically plausible ways.

Given the requirement of a mixture of symbolic and subsymbolic processes, modern production systems would seem well designed to model analogy. Salvucci & Anderson (2001) describe a relatively successful application of the ACT-R theory to modeling results in the analogy literature. Before reviewing it, we would like to highlight the value added of such a theory. While the model incorporates many of the insights of the other theories, it is not just a matter of implementing these theories in ACT-R. As a complete theory of cognition, the model contributes three things that are lacking in these other models. First, it naturally maps these
processes onto precise predictions about real-world metrics of latency and correctness, rather than the more qualitative and ordinal predictions that have typified other theories. Second, it integrates the process of analogy with the rest of cognition and thus makes predictions about how processes such as eye movements are interleaved with the analogy process. Third, it shows that the mechanisms underlying analogy are the same as the mechanisms underlying other aspects of cognitive processing.

Figure 2 illustrates the representation of the famous solar system analogy (Gentner, 1983) in the Salvucci and Anderson system. Analogs are represented as higher-order structures built up of three components: objects, relations, and roles. The first two components, objects and relations (represented as ovals in Figure 2) serve the same purpose as in other theories of analogy: Objects are the semantic primitives of the analogs, while relations link objects or relations together according to their function. The solar-system domain contains the two objects ss-sun and ss-planet, along with the three relations ss-causes, ss-attracts, and ss-revolves. Similarly, the atom domain contains the two objects at-nucleus and at-electron and the three relations at-causes, at-attracts, and at-revolves. The boxes in Figure 2 represent the third component of an analog structure – roles – which serve to link objects and relations to form higher-order conceptual structures. Each role comprises five components:

- **parent**: a pointer to the parent relation
- **parent-type**: the semantic type of the parent relation
- **slot**: the relation slot that the object fills in the relation
- **child**: a pointer to the child object or relation
- **child-type**: the semantic type of the child object or relation
For example, in the case of the **ss-attractor** role, **ss-attracts** is the **parent**, **attracts** is the **parent-type**, **attractor** is the **slot**, **ss-sun** is the **child**, and **sun** is the **child-type**.

Salvucci and Anderson (2001) describe a path-mapping process by which the structure in the source is made to correspond to the structure in the analog. This mapping process is achieved by production rules that essentially walk through these graphs looking for correspondences. The critical step in this mapping is retrieving roles from the target domain to map onto roles in the source domain. This is achieved by the partial matching process in ACT-R that selects the most similar role. Similarity between the source and target role is determined based on the similarities between the parent-type, slot, and child-type components of the roles. One of the consequences is that the model can be misled to select inappropriate analogs on the basis of surface similarity between the components of a source and target. For instance, the model successfully simulated the results from Ross (1989) that showed role confusions in probability problems based on surface similarities between examples. One limitation of the path-mapping process built into this model is that it only considers one proposition at a time. For that reason, the model cannot solve analogies that require the consideration of multiple propositions in parallel, whereas people and other models (e.g., Hummel & Holyoak, 1997) can.

On the other hand, the production system control structure leads to other predictions. Since the model goes from the source to the target, it has a preference for many-to-one mappings over one-to-many mappings. This enables the model to successfully predict the results of Experiment 2 in Spellman and Holyoak (1996). They gave subjects two stories involving countries on different
planets and asked subjects to map countries on one planet to those on the other. The story relations can be summarized as follows:

<table>
<thead>
<tr>
<th>Story 1</th>
<th>Story 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>richer (Afflu, Barebrute)</td>
<td>richer (Grainwell, Hungerall)</td>
</tr>
<tr>
<td>stronger (Barebrute, Compak)</td>
<td>stronger (Millpower, Mightless)</td>
</tr>
</tbody>
</table>

The relations include an ambiguous mapping -- namely, the mapping of Barebrute to either Hungerall or Millpower. Subjects were divided into two conditions: In the 1-2 condition, subjects mapped objects from story 1 to those in story 2; in the 2-1 condition, subjects mapped objects from story 2 to story 1. In both conditions, subjects had the option of including any, all, or no objects in their mapping, thus allowing the possibility of a one-to-one, one-to-many, or many-to-one mapping, if so desired. Spellman and Holyoak found that subjects rarely produced one-to-many mappings (less than 2% of subjects), whereas they frequently produced many-to-one mappings (more than 30% of subjects).

In addition to reproducing these results in the literature, Salvucci and Anderson had subjects try to determine the analogies between two stories and collected their eye movements while they were doing so. The data showed that subjects moved their eyes back and forth between the two stories as they read them and searched for the analogs. The Salvucci and Anderson model was able to predict the eye movement transitions. This is a critical study because it shows how analogy is dynamically integrated with cognition and how it can control – and be determined by – processes like eye movements.
Summary

This production-system model of analogy specifies a set of production rules that implement a path-mapping process through declaratively represented source and target structures. That is, the model posits that analogies are made and used via an explicit process of path mapping that is influenced by the relative activation levels of the elements to be mapped. The subsymbolic mechanisms governing declarative retrieval specify which parts of those declarative structures will be retrieved and when. In this way, the model makes specific, quantitative predictions about the results of analogy making and its time-course (as observed through eye movement data).

Although analogy making is a process that produces new knowledge – the mapping, which in turn can be used to produce new inferences – the process of analogy usually occurs in a single trial, without much learning. Thus, Table 5 highlights that this model of analogy making draws on three of the four performance mechanisms in ACT-R.
Working Memory

Just as the previous section’s model of analogy makes heavy use of declarative knowledge and corresponding mechanisms, so does this section’s model of working memory. Working memory has been implicated in the performance of such diverse tasks as verbal reasoning and prose comprehension (Baddeley & Hitch, 1974), sentence processing (Just & Carpenter, 1992), free recall learning (Baddeley & Hitch, 1977), prospective memory (Marsh & Hicks, 1998), and note-taking and writing (Engle, 1994). (See also Morrison chapter, this volume.) This research has suggested that working memory resources are limited because, as working memory demands of a task increase, participants’ performance declines. Moreover, working-memory limitations appear to differ across people, such that some people show a more striking decrease in performance as a function of task demands than others.

Each of the four production systems discussed thus far has an account for the impact of working memory demands on cognitive processing (see Miyake & Shah, 1999). EPIC implements Baddeley’s articulatory loop via production rules acting on the auditory store and vocal/motor processor. These production rules implement strategies for rehearsal and recall and are constrained by the processing features of the modules they engage (e.g., all-or-none decay of items from the auditory store and time to re-read an item by the vocal/motor processor). In contrast, Soar assumes no a priori limit to working memory through its dynamic memory. Rather, limitations arise when multiple levels of processing are necessary to establish multiple subgoals to handle a sequence of impasses. In the CAPS architecture(s), working memory limitations are captured through a limitation in the amount of activation that can propagate through the system: When less activation is available, production-rule firing takes more
processing cycles. CAPS has been used to model different patterns of sensitivity to working
memory demands among groups of individuals with low, medium and high capacity (e.g., Just &
Carpenter, 1992).

In ACT-R, working memory limitations are imposed via a limitation to the amount of attention
that can be focused on the current goal. This attentional activation (also called source activation)
serves to (1) maintain elements of the goal as highly active, (2) activate above their resting levels
any goal-relevant facts in declarative memory, and (3) suppress below their resting levels any
facts negatively associated with the current goal. Although they sound similar, the CAPS and
ACT-R limitations in activation are quite different. CAPS directly limits total activation in the
system, whereas ACT-R limits the ability to differentially activate goal-relevant information
above goal-irrelevant information. In other words, greater source activation in ACT-R is akin to
having a better signal-to-noise ratio for retrieving facts. It is worth noting that the working
memory limitations in both ACT-R and CAPS are imposed as constraints on a particular model
parameter, whereas in other working memory accounts, e.g., SOAR, the connectionist system
LISA (Hummel & Holyoak, 1997), and to some degree EPIC, these limitations emerge as a
natural consequence of general processing.

In this section, we demonstrate how ACT-R’s implementation of working memory can be used
to estimate individuals’ working-memory capacity from performance on one task (call it Task A)
and then make accurate, zero-parameter predictions of those individuals’ performance on other
tasks B, C, etc.. Task A is a Modified Digit Span task (MODS), designed as an isomorph of the
reading span (Daneman & Carpenter, 1980) and the operation span (Turner & Engle, 1989). In
this task, participants perform dual tasks of reading various presented characters and memorizing the exact order of digits only. Figure 3 shows a sample MODS trial in which the participant would read “a j 2 b i e 6 c f 8” and then recall the digits in order (“2 6 8”). Because the task is fast paced (there is little time for idiosyncratic strategies), it draws on skills that are highly practiced (there is little chance for skill or knowledge differences), and both aspects of the task are monitored (there is little opportunity for different levels of task compliance), most of the variation in performance on this task should be due to differences in participants’ fundamental processing capacities. For our modeling purposes, we take this to be variation in source activation.

An ACT-R model of the MODS task successfully fits individual participant’s data as a function of set size (Figure 4) and as a function of serial position for the set size 6 trials (Figure 5), by only varying the source-activation parameter (Daily, Lovett, & Reder, 2001). This suggests that source activation presents a reasonable implementation of working memory that can explain the variation in individuals’ MODS performance. Moreover, because source activation plays the same role in all ACT-R models, this allows for predictions to be made for the same participants on other tasks, by plugging each participant’s estimated source-activation parameter into the other task models. In Lovett et al. (2000), this is accomplished for the n-back task. Specifically, individual participant estimates of source activation were derived from their MODS task performance and then used to make zero-parameter, individual participant predictions on the n-back task.
The n-back task is a continuous trial paradigm in which, for a given block of trials, participants are asked to respond to whether each letter stimulus is a repeat of the stimulus “n” trials back (e.g., Braver et al., 1997; Cohen et al., 1994). For example, suppose the participant saw the stimuli “U E E R E K L L”. In a “1-back” block, a participant should say “yes” to the third and last stimulus and “no” elsewhere, whereas in a “2-back” block, the participant should say “yes” to the fifth stimulus and “no” elsewhere. As “n” increases, the working memory demands of the task increase and, not surprisingly, performance degrades. Figure 6 shows high fidelity modeling fits at the individual participant level in the n-back task, by using the individualized source activation parameter values that were estimated from the same participants’ MODS performance.

Summary

This model of working memory includes production rules to perform the various tasks studied. Across all tasks, the ACT-R architecture provides a single theory of working memory in which working memory limitations are represented by a fixed amount of source activation, propagated from the current focus of attention to increase the activation of goal-relevant items and to decrease the activation of goal-irrelevant items. The larger this source activation for a given individual, the greater the degree of facilitation/suppression of goal-relevant/irrelevant items. This leads to direct performance implications as a function of source activation, plus there are indirect effects in the model (e.g., more rehearsals are possible because of faster retrievals with high source activation) that can further the implications. In sum, this working memory model relies most heavily on the relative activation levels of declarative chunks (both those that are part of the initial model and those that are newly acquired as part of task performance); this is highlighted by the check marks filling the declarative chunks row in Table 6.
Categorization

Research on human category learning has a history that extends back at least to Hull's (1920) study of learning to categorize Chinese symbols and his conclusions in favor of an associative learning proposal. It was an important domain early in the cognitive revolution where theorists argued for various hypothesis-testing theories (e.g., Trabasso & Bower, 1964, Levine, 1975). The hypothesis-testing theories were based on research with stimuli that had a very simple, often one-dimensional categorical structure. The 1970's saw a renewed interest in more complex, fuzzy categories and proposals for prototype theories (Reed, 1972; Rosch, 1975) and exemplar theories (e.g., Medin & Schaffer, 1978). The rise of connectionist models resulted in the proposal of associative theories (e.g., Gluck & Bower, 1988) not that different than the original Hull proposal. While the original research focused on accuracy data, there has been a new emphasis on latency data to help choose among theories (e.g., Lamberts, 1998; Nosofsky & Palmeri, 1997). Recently, neuro-imaging and other cognitive neuroscience data have been recruited to try to decide among alternative theories (e.g., Ashby, Alfonso-Reese, Turken & Waldron, 1998; Smith, Patalano & Jonides, 1998). There has been an impressive growth in the characterizations of the phenomena in category learning (see Medin & Rips, this volume). However, the field does not seem any closer to coming to consensus as to what "the" mechanism of category learning is.

Anderson and Betz (2001) produced a production system model that reflected the belief that this contest of theories was misplaced and that different mechanisms were being used to different degrees in different experiments. In particular, they implemented in ACT-R two alternative models that have been advanced for categorization, Nosofsky, Palmeri, and McKinley’s (1994)
rule-plus-exception (RULEX) model and Nosofsky and Palmeri’s (1997) exemplar-based random walk (EBRW) model. The first model proposes that subjects store explicit rules for category membership and possible exceptions. The EBRW model proposes that subjects retrieve instances that are similar to the test stimulus and assign the stimulus to the category that has the most retrieved exemplars, after exceeding a particular threshold. While the original models are mathematical characterizations of participants’ behavior, the ACT-R model is a computational system that actually performs the task. Production rules provide the control structure for how the ACT-R model approaches the task (e.g., whether it employs a RULEX- or EBRW-based approach), while declarative memory stores the rules, exceptions, and examples used and strengthened by each approach. The subsymbolic components of the architecture determine which production rules and declarative structures are retrieved at any given point in time.

The component of the model incorporating an EBRW approach retrieves past instances from memory as a function of their similarity to the current stimulus. This depends critically on the ability of the ACT-R system to retrieve partially matching traces. Specifically, the probability of retrieving a memory in ACT-R is a function of how similar it is to the memory probe. Anderson and Betz (2001) show that this retrieval function yields a similar, but not identical, selection rule to that used in the original Nosfosky and Palmeri formulation. In addition, the ACT-R mechanism for chunk strengthening favors the retrieval of more frequently presented items and hence produces a speed up similar to the speed up in EBRW (which uses multiple traces and a Logan (1988) race process). While the original EBRW and the ACT-R implementation are not identical, they prove largely indistinguishable in their predictions. This near equivalence is strongly dependent on the pre-existing subsymbolic processes built into ACT-R.
The component of the ACT-R model implementing a RULEX approach depends more on the symbolic production-level system, as the actual logic of hypothesis testing in RULEX is quite complex (e.g., different rules specify when to settle on a hypothesis, when to switch from single dimension to multiple dimension rules, and when and how to form exceptions). Nevertheless, the subsymbolic level of ACT-R, which governs the selection among production rules based on their ever-changing utility values, is essential for this model component to capture the randomness of RULEX. Indeed, this noisy selection process enables this model component to reproduce the wide variety of hypotheses that subjects display.

The Anderson and Betz effort is a relatively successful integration of the two models. Moreover, the effort adds value over the two original models. First, it establishes that the two theories are not necessarily in opposition and in fact reflect the same underlying subsymbolic processes but with different symbolic control. Moreover, those subsymbolic processes are the same ones as can be used to model other, very different domains of human cognition. Also, because both categorization mechanisms are able to sit within the same architecture, Anderson and Betz were able to address the issue of choice between the two mechanisms. This depends on the relative utility of these two mechanisms. Anderson and Betz show that the mixture of the two strategies is able to account for phenomena that neither strategy alone can account for. They also show that there is a natural tendency for this mixture of strategies to evolve from being dominated by rule-based classification to being dominated by instance-based classification because the latter is more efficient. Figure 7 shows the tendency for exemplar use to increase in two of the models.
reported in Anderson & Betz. This increased exemplar use is consistent with reported results of a strategy shift with extensive practice (Johansen & Palmeri, 2002).

Summary

This contemporary production-system model of categorization integrates two approaches (implemented as different sets of co-habitating production rules) and chooses between them (based on the production rules’ learned utility values). In one approach, production rules are the conduit for creating and accessing exemplars (implemented as declarative chunks) in a context-sensitive and frequency-sensitive way. In the other approach, production rules create and manipulate declarative rules for categorizing items. In all cases, the ACT-R subsymbolic learning mechanisms for production rules and declarative chunks govern how these kinds of knowledge are used. Table 7 highlights this (see checks in the right column) as well as the fact that this model employs ACT-R’s symbolic learning mechanism for declarative chunks.
Skill Learning

Research into skill learning can be roughly divided into two categories. One category focuses on how skills are learned in the first place (e.g., Catrambone, 1996; Chi et al., 1989; VanLehn & Jones, 1993). The other focuses on how skills are refined to achieve domain expertise. (See also, Chi and Ohlsson, this volume.) Research in the former category has addressed issues of learning from instruction, transfer, and induction. Research in the latter category has addressed issues of generalization, specialization, and automaticity. A unified approach merges these issues into a single explanation. Production-systems models – particularly those that address the question of production-rule learning – hold the promise of offering such an explanation.

Among production-systems models, Soar holds the most parsimonious view of skill learning with its single mechanism, chunking. Chunking is invoked whenever the system encounters an impasse (i.e., when existing production rules do not directly specify the next step). At this point, the system creates a subgoal to solve the impasse by applying domain-general production rules. Solving the impasse creates a new rule, specialized for that situation. A similar rule-learning process is employed by Cascade, a model of skill acquisition that incorporates both the impasse-repair-reflect cycle and analogical problem solving (VanLehn, 1999). After the new rule is learned, when Cascade subsequently encounters the same (or a related) situation, it can apply the new rule directly and avoid the extra processing. These models employ specialization – making a new rule that is a specific version of its parents – and composition – combining multiple production rules into one new rule.
ACT-R also has a production-rule learning mechanism. This mechanism combines *composition* – merging two production rules that fire in sequence – and *proceduralization* – creating a new version of an existing production rule, where the new version avoids fact retrieval by instantiating necessary information directly into the new rule. For example, consider a pair of production rules that solve addition problems of the form ‘x+y=?’ by first retrieving the relevant addition fact from memory and then using this fact to make a response (A and B in Table 8). When these production rules are applied to the problem ‘2+5=?’, a single production rule is learned (C in Table 8) that combines the two steps into one but is specific to the case of “2+5”. This mechanism treats skill learning as a ubiquitous process of building more specific, more powerful, and less explicit problem-solving knowledge. Greater power comes from the knowledge being faster, no longer subject to retrieval failures, and incurring lower working-memory load. Less explicitness comes from the fact that the new rule transforms a fully inspectable, declarative fact into the body of a production rule, where knowledge is not open to inspection.

We exemplify ACT-R’s production-rule learning in the context of an experimental paradigm where rule-like knowledge is learned in many different forms (Anderson & Fincham, 1994; Anderson, Fincham, and Douglass, 1997). This paradigm involves teaching participants a number of *sports facts*, e.g., “Hockey was played on Saturday at 3 and then on Monday at 1”. After committing these sports facts to memory, participants are told that each one conveys a particular pattern or rule for the game times for that sport (e.g., Hockey’s second game time is always two days later and two hours earlier than its first). Participants are then given practice at using these sports facts to solve new problems where either the first or second time is given and the other must be predicted. Figure 8a shows the speed-up in
performance from Anderson & Fincham (1994) as participants practiced this over three days (each “bump” occurred at the beginning of a new day). Figure 8a also shows the predictions of an ACT-R simulation (Taatgen and Wallach, 2002) that involves four representations of the sports-fact knowledge.

Figure 8b tracks the contribution of these four sources of knowledge over the three days. The initial representation was simply the set of eight studied sports facts represented as declarative facts (see first row of Table 9). Specifically, each sports fact was represented in terms of four inter-related chunks to capture the two days and two times for that sport (e.g., “Hockey’s first event day was Saturday” “Hockey’s first event time was 3” “Hockey’s second event day was Monday” “Hockey’s second event time was 1”). To solve problems using these facts, the model was endowed with a set of production rules representing the weak methods of direct retrieval (applicable for the original facts) and analogy.

From this initial knowledge base, the model generated the other three representations of the sports-fact knowledge. The first of these represents the rule-like relationships of each original sports fact as two declarative chunks (e.g., “Hockey’s day relationship is + 2” “Hockey’s time relationship is –2”). The model produces this declaratively represented generalization as a byproduct of the analogizing process (see second row of Table 9). Once these generalized relationships are derived, applying them to a new problem is much simpler than solving by analogy. The second new representation of knowledge comes in true production-rule form. Specifically, a new production rule is learned which merges the steps involved in applying the declarative generalizations just mentioned. Note that this production rule is specialized to the sport and direction (time 1 -> time 2 or vice versa) under which it was generated. Such a “directional” production rule should show faster performance for problems in the practiced
direction, and Anderson and Fincham showed that such asymmetry develops with extensive practice.

The third new representation is a specific instance representing the solution to a particular previous (and often repeated) problem. This knowledge can complete a new problem in just two steps (one each for the day and time). However, it is specific to a particular problem and is only generated after the preceding forms of knowledge have paved its way. It predicts that participants will be faster on frequently repeated problems, and Anderson, and Fincham, and Douglass (1997) provide evidence for such item-specific learning.

Summary

The most noteworthy aspect of this production-systems model of skill learning is that it posits multiple, overlapping stages to the development of a new skill, some of which represent the new “skill” knowledge in production-rule form and some of which do not. Because of the acquisition of new production rules and new declarative chunks, the model relies on both symbolic learning mechanisms in ACT-R. In addition, these new knowledge representations are refined and strengthened through experience, drawing on ACT-R’s subsymbolic learning mechanisms. Finally, the model chooses among the different knowledge representations via the subsymbolic performance mechanisms: As declarative representations are strengthened through use, those with higher activation will tend to get retrieved; and as new production rules are used and are successful, those with higher utilities will tend to get chosen (over more generic production rules that employ declarative representations). In sum, this model draws on all eight mechanisms presented in Table 10.
Language Learning: Past Tense

The learning of the English past tense is another domain where symbolic and subsymbolic models have clashed. The appearance of over-regularization errors in children’s past tense (e.g., go-goed as opposed to go-went) had been originally taken as evidence (e.g., Brown, 1973) that children were acquiring abstract rules. However, Rumelhart and McClelland (1987) showed that by learning associations between the phonological representations of stems and past tense it was possible to produce a model that made overgeneralizations without building any rules into it. It was able to account for the U-shaped learning function demonstrated by children by which they first do not produce such overgeneralization, then do, and finally gradually eliminate the overgeneralizations. This attracted a great many critiques and, while the fundamental demonstration of generalization without rules stands, it is acknowledged by all to be seriously flawed as a model of the process of past-tense generation by children. Many more recent and more adequate connectionist models (some reviewed in Elman, et al. 1996) have been proposed and many of these have tried to use the backpropogation learning algorithm.

This would seem like an appropriate domain for production system models, and Taatgen and Anderson (2002) have produced a successful model of these phenomena. Significantly, they show that one can account for past tense learning with a similar dual mechanism model like that of Anderson and Betz (2001). The model posits that children initially approach the task of past-tense generation with two strategies. Given a particular word like "give" they can either try to retrieve the past tense for that word or they can try to retrieve some other example of a past tense (e.g. "live" - "lived") and try to apply this by analogy to the current case. (In the case of analogy, previously encountered present-past tense pairs serve as potential sources, and a source whose present tense form is similar to the target’s present tense form will be retrieved. Then, the
transformation driving the past tense form in the retrieved source is applied to the target.)

Eventually, through the production-rule learning mechanisms in ACT-R, the analogy process will be converted into a production rule that generatively applies the past-tense rule. Once the past-tense rule is learned, the generation of past tenses will largely be determined by a competition between the general rule and retrieval of specific cases. Thus, ACT-R has basically a dual-route model of past-tense generation where both routes are implemented by production rules. The rule-based approach depends on general production rules while the exemplar approach depends on the retrieval of declarative chunks by production rules that implement an instance-based strategy.

Figure 9 graphically displays the variety of ways this model can generate the past tense. Although all of these options are implemented in ACT-R production rules, only the two rightmost options represent the application of general past-tense rules (e.g., add “ed”). The second and third options initiate procedures for retrieving a memory trace that can then be applied directly or by analogy to the current situation.

The general past-tense rule, once discovered by analogy, gradually enters the competition as the system learns that this new rule is widely applicable. This gradual entry, which depends on ACT-R’s subsymbolic utility-learning mechanisms, is responsible for the onset of overgeneralization. While this onset is not all-or-none in either the model or the data, it is a relatively rapid transition in both model and data and corresponds to the first turn in the U-shaped function. However, as this is happening, the ACT-R model is encountering and strengthening the declarative representations of exceptions to the general rule. Retrieval of the exceptions comes to counteract the overgeneralizations. Retrieval of exceptions is preferred
because they tend to be shorter and phonetically more regular (Burzio, 2002) than regular past
tenses. Growth in this retrieval process corresponds to the second turn in the U-shaped function
and is much more gradual—again both in model and data.

Note that the Taatgen model, unlike many other past-tense models, does not make artificial
assumptions about frequency of exposure but learns given a presentation schedule of words (both
from the environment and its own generations) like that actually encountered by children. Its
ability to reproduce the relatively rapid onset of overgeneralization and slow extinction depends
critically on both its symbolic and subsymbolic learning mechanisms. Symbolically, it is
learning general production rules and declarative representations of exceptions.
Subsymbolically, it is learning the utility of these production rules and the activation strengths of
the declarative chunks.

Beyond just reproducing the U-shaped function, the ACT-R model explains why exceptions
should be high-frequency words. There are two aspects to this explanation. First, only high-
frequency words develop enough base-level activation to be retrieved. Indeed the theory predicts
how frequent a word has to be in order to maintain an exception. Less obviously, the model
explains why so many high-frequency words actually end up as exceptions. This is because the
greater phonological efficiency of the irregular form promotes its adoption according to the
utility calculations of ACT-R. Indeed, in another model that basically invents its own past-tense
grammar without input from the environment, Taatgen showed that it will develop one or more
past-tense rules for low-frequency words but tend to adopt more efficient irregular forms for
high-frequency words. In the ACT-R economy the greater phonological efficiency of the
irregular form justifies its maintenance in declarative memory if it is of sufficiently high frequency.

Note that the model receives no feedback on the past tenses it generates, unlike most models but in apparent correspondence with the facts about child language learning. However, it receives input from the environment in the form of the past tenses it hears and this input influences the base-level activation of the past-tense forms in declarative memory. The model also uses its own past-tense generations as input to declarative memory and can learn its own errors (a phenomenon also noted in cognitive arithmetic—Siegler, 1988). The amount of overgeneralization displayed by the model is sensitive to the ratio of input it receives from the environment to its own past-tense generations.

Summary

While this model of past-tense generation fully depends on the existence (and emergence) of rules and symbols it also critically depends on the subsymbolic properties of ACT-R to produce the observed graded effects. Table 11 highlights the fact that this model relies on learning of both declarative and procedural knowledge at both the symbolic and sub-symbolic level. This eclectic position enables the model to achieve a number of other features not achieved by many other models:

- It does not have to rely on artificial assumptions about presentation frequency.
- It does not need corrective feedback on its own generations.
- It explains why irregular forms tend to be high frequency and why high-frequency words tend to be irregular.
- It correctly predicts that novel words will receive regular past tenses.
It predicts the gradual onset of overgeneralization and its much more gradual extinction.
Conclusions and Future Directions

This chapter describes six production-systems models accounting for six different areas of cognition: problem-solving choice, analogy making, working memory, categorization, skill learning, and past-tense learning. In some cases, an important contribution of the model lies in specifying a production system that implements a fairly general reasoning strategy (e.g., analogy making and categorization). The analogy model specifies a path-mapping process as a set of production rules. The categorization model specifies two processes for categorization — by rules (with exceptions) and by retrieving multiple category exemplars — both implemented as sets of production rules that cohabit a single production system. In both models, it is not only the production rules that govern model behavior but also subsymbolic quantities that influence how the production rules do their work. In the analogy model, subsymbolic activation levels associated with different declarative chunks influence which parts of the analogy will get mapped and when; in the categorization model, subsymbolic utility levels associated with different production rules influence which categorization approach will be chosen and when.

Another contribution made by several of the models is specifying how multiple strategic approaches to a given task can be integrated. Indeed, a common but often underemphasized feature of high-level cognitive tasks is that people can approach them in so many ways. The problem-solving model addresses this issue of choice directly and illustrates a modern interpretation of production-rule conflict resolution. Specifically, this model (along with the categorization, skill learning, and past-tense learning models) demonstrate that a noisy selection of the production rule with highest utility (where utility is naturally learned through experience by the system) works well to choose among different strategies.
A related contribution made by some of these models is making clear that rule-like thinking is not always best represented in terms of production rules. The categorization, skill learning, and past-tense learning models all use multiple strategic approaches; in the latter two models, one of the approaches is based on production-rule representations of knowledge and another is not based on production-rule representations of knowledge. Together, the two representational forms complement each other in a way that accounts for the variability in people’s behavior.

Accounting for variability is probably the most notable contribution of the working memory model given that it posits a theory of working-memory limitations that can be used to estimate individuals’ working memory capacities and then predict other task performance on that basis.

What is most striking about these models as a whole, however, is that they make use of the same set of mechanisms for learning and using knowledge across such a disparate set of tasks and that they use the same two kinds of knowledge representations – production rules and declarative chunks. While each model emphasizes a somewhat different subset of mechanisms (cf. Tables 4, 5, 6, 7, 10, and 11), they all fit together in a unified architecture, just as the many processes of human cognition all must fit together in the human brain. Likewise, modern productions systems offer an understanding of how the many components of cognition are integrated.

Production systems into the future
Given the progress represented by the relatively few models presented here, it is worthwhile to speculate how production systems will continue to be involved in future research on cognition. Two areas where production systems have ventured in the past few years are already showing initial levels of success and promise to play a large role in future developments in modeling.
One of these areas involves the development of production-system models that can handle complex tasks. Complexity can arise in many dimensions, but one involves the dynamic qualities of the task. Air-traffic control is a dynamic task in that it requires continuous attention to changing stimuli and changing task demands. It is complex in that it requires the integration of multiple areas of knowledge (e.g., different skills to handle the different situations) and the integration of perceptual, motor and cognitive processing (e.g., actually working with a graphical and keyboard interface akin to what real air-traffic controllers use). A modeling competition to account for human data of various sorts on an air-traffic-control task called AMBR (Agent-Based Modeling and Behavior Representation) set the bar high with regard to modeling a complex, dynamic task (Gluck & Pew, 2001). Several production-system models including one built within ACT-R (Lebiere, Anderson, & Bothell, 2001) and one built within an EPIC-Soar combination (Chong & Wray, 2002) took on the challenge and demonstrated success in accounting for the number and type of errors of human controllers and, in the case of ACT-R to account for similar levels of variability in performance among human controllers. Similar successes are beginning to arise in real-world applications of production-systems models. For instance, there is the Soar model that managed to fly 50 Air Force training missions (Jones, Laird, Nielsen, Kenney, & Koss, 1999), and other examples of production-systems models used in industry and military applications will likely become more the rule than the exception (pardon the pun!). Some of these will likely come in the form of cognitive agents (e.g., Best, Lebiere, & Scarpinatto, 2002) that act in virtual worlds (e.g., for training purposes with humans) or real environments (e.g., in coordination with robotic systems).

Another area of current growth in production-systems modeling that promises to expand is the integration of production-systems models (i.e., their components and their predictions) with
neuroimaging work. With the growth of functional neuroimaging as a means of studying cognition (see Goel, this volume), the field of cognitive modeling has another dependent measure for testing models’ predictions. To take advantage of this additional constraint, however, the model must posit some mapping between model output and neuroimaging results. A direct approach is to map brain locations to model functions and then predict localization of activation on the basis of model functions. This basic approach can be elaborated to account for the time course of brain activity either at a coarse grain size (e.g., predicting differential localization of activity early vs late in the task or between conditions) or at a fine grain size (e.g., within a single trial). Both of these approaches have been used in tasks ranging from language processing (Just, Carpenter, & Varma, 1999) to equation solving (Anderson, Qin, Soh, Stenger, & Carter, in press), to planning (Fincham, Carter, vanVeen, Stenger, & Anderson, 2002) to task-switching (Sohn, Ursu, Anderson, Stenger, & Carter, 2000). The goal of this work is to improve our understanding of brain function and also of the mapping to production-system models of thought. While production systems have tended to describe cognitive processes at a high level of abstraction, the trend has been toward more and more fine-grained models, so it is now becoming appropriate to consider the neural processing implications of many of the issues in production-system models.
References


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


Author Notes

The authors thanks Stacey Becker, Norma Chang, and Niels Taatgen for comments on an earlier draft.
Endnotes

1 Interestingly, recent production-system models have returned to embrace an individual participant approach with quantitative analyses (e.g. Daily, Lovett, & Reder, 2000; Lovett, Daily, & Reder, 2001).

2 Asynchronous parallelism means that each perceptual/motor module can work in parallel with others in such a way that the actions of each need not be synchronized with the others’.

3 However, Young and Lewis (1999) have posited that no more than two items with the same type of coding (e.g., phonological or syntactic) can be stored in dynamic memory at a time.
Table 1

Illustrative examples of production rules, written in English

<table>
<thead>
<tr>
<th>Number</th>
<th>Specification of production rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>When my current goal involves navigating in a dark room, then I flip the light switch in that room</td>
</tr>
<tr>
<td>2</td>
<td>When my current goal is to go to a location that is more than 300 miles away, then I set a subgoal to go to the local airport.</td>
</tr>
<tr>
<td>3</td>
<td>When my current goal is to answer an arithmetic problem of the form $D_1 + D_2$, then I change the goal to try retrieving the sum of $D_1$ and $D_2$ from memory.</td>
</tr>
<tr>
<td>4</td>
<td>When my current goal is to answer an arithmetic problem of the form $D_1 + D_2$, then I hold up $D_2$ fingers and change the goal to count them starting with the number after $D_1$.</td>
</tr>
</tbody>
</table>
Table 2

Examples illustrating that the form (rows) and content (columns) of knowledge are independent

<table>
<thead>
<tr>
<th>Representational form</th>
<th>Knowledge contents</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Rule-like</td>
</tr>
<tr>
<td>Production rule</td>
<td>When I want to type a letter &amp; I know its finger move, then make that move</td>
</tr>
<tr>
<td>Declarative fact</td>
<td>To touch-type, one must make the finger move corresponding to the currently desired letter</td>
</tr>
</tbody>
</table>
Table 3

Template for describing the knowledge structures and mechanisms involved in each ACT-R model.

<table>
<thead>
<tr>
<th></th>
<th>Performance Mechanisms</th>
<th>Learning Mechanisms</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Symbolic</td>
<td>Subsymbolic</td>
</tr>
<tr>
<td>Declarative Knowledge</td>
<td>Knowledge (usually facts) that can be directly verbalized</td>
<td>Relative activation of declarative chunks affects retrieval</td>
</tr>
<tr>
<td>Production Rules</td>
<td>Knowledge for taking particular actions in particular situations</td>
<td>Relative utility of production rules affects choice</td>
</tr>
</tbody>
</table>
Table 4

Knowledge structures and mechanisms used in a production-systems model of choice

<table>
<thead>
<tr>
<th></th>
<th>Performance Mechanisms</th>
<th>Learning Mechanisms</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Symbolic</td>
<td>Subsymbolic</td>
</tr>
<tr>
<td>Declarative</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chunks</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Production</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rules</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Table 5

Knowledge structures and mechanisms used in a production-systems model of analogy making

<table>
<thead>
<tr>
<th></th>
<th>Performance Mechanisms</th>
<th>Learning Mechanisms</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Symbolic</td>
<td>Subsymbolic</td>
</tr>
<tr>
<td>Declarative Chunks</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Production Rules</td>
<td>√</td>
<td></td>
</tr>
</tbody>
</table>
Table 6

Knowledge structures and mechanisms used in a production-systems model of working memory

<table>
<thead>
<tr>
<th></th>
<th>Performance Mechanisms</th>
<th>Learning Mechanisms</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Symbolic</td>
<td>Subsymbolic</td>
</tr>
<tr>
<td>Declarative Chunks</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Production Rules</td>
<td>√</td>
<td>√</td>
</tr>
</tbody>
</table>
This model of categorization relies on three out of four of ACT-R’s learning mechanisms.

<table>
<thead>
<tr>
<th>Performance Mechanisms</th>
<th>Learning Mechanisms</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Symbolic</td>
</tr>
<tr>
<td>Declarative</td>
<td>√</td>
</tr>
<tr>
<td>Chunks</td>
<td></td>
</tr>
<tr>
<td>Production</td>
<td>√</td>
</tr>
<tr>
<td>Rules</td>
<td></td>
</tr>
</tbody>
</table>
Table 8. Two parent production rules and the learned child production rule.

<table>
<thead>
<tr>
<th>Production A</th>
<th>Production B</th>
<th>Production C</th>
</tr>
</thead>
<tbody>
<tr>
<td>When the goal is to add the numbers ( x ) and ( y ), then try to retrieve the sum of ( x ) and ( y ).</td>
<td>When the goal is to add the numbers ( x ) and ( y ) and the sum of ( x ) and ( y ) has been retrieved as ( z ), then update the goal with ( z ) as the answer.</td>
<td>When the goal is to add 2 to 5, then update the goal with 7 as the answer.</td>
</tr>
</tbody>
</table>
Table 9
Model’s different representations of the sports facts from Anderson & Fincham (1994)

<table>
<thead>
<tr>
<th>Knowledge type</th>
<th>Decl vs. Prod</th>
<th>How generated</th>
<th>#/sport</th>
<th># steps required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original sports fact components</td>
<td>Declarative</td>
<td>Original study</td>
<td>4</td>
<td>~20</td>
</tr>
<tr>
<td>General relationships</td>
<td>Declarative</td>
<td>Analogy on original sports fact components</td>
<td>2</td>
<td>~10</td>
</tr>
<tr>
<td>Procedural relation</td>
<td>Production</td>
<td>Production compilation on relationships</td>
<td>4</td>
<td>~6</td>
</tr>
<tr>
<td>Studied instance</td>
<td>Declarative</td>
<td>Result of previous (&amp; often repeated) example</td>
<td>2 for each example</td>
<td>2</td>
</tr>
</tbody>
</table>
Table 10.

Knowledge structures and mechanisms used in production-systems model of skill learning

<table>
<thead>
<tr>
<th></th>
<th>Performance Mechanisms</th>
<th>Learning Mechanisms</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Symbolic</td>
<td>Subsymbolic</td>
</tr>
<tr>
<td>Declarative</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Chunks</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Production</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Rules</td>
<td>√</td>
<td>√</td>
</tr>
</tbody>
</table>
This model of past tense generation relies on all four of ACT-R’s learning mechanisms

<table>
<thead>
<tr>
<th></th>
<th>Performance Mechanisms</th>
<th>Learning Mechanisms</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Symbolic</td>
<td>Subsymbolic</td>
</tr>
<tr>
<td>Declarative Chunks</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Production Rules</td>
<td>√</td>
<td>√</td>
</tr>
</tbody>
</table>
Figure Captions

Figure 1. Initial state and three possible subsequent problem states from the Building Sticks Task.

Figure 2. Sample analogs for the solar-system and atom domains.

Figure 3. Graphic illustration of a MODS trial with a memory set of size 3. The differences in the positions of the characters on-screen have been exaggerated for clarity.

Figure 4. Model fits for four representative subjects from Daily et al. (1999). Filled symbols are subject data; open symbols are the model’s predictions.

Figure 5. Fits to the serial position data (largest set size only) for four typical subjects. Filled symbols are subject data; open symbols are the model’s predictions.

Figure 6. N-back performance and model predictions for individual participants where parameter estimates of individuals’ working memory capacities were derived from performance on the MODS task.

Figure 7. Proportion exemplar use across blocks in two data sets modeled in Anderson and Betz (2001).

Figure 8. Latency to respond across trials in each session in Anderson & Fincham, 1994, (panel a), and proportion of simulation runs in which particular knowledge representations were used across trials in Taatgen and Wallach, 2002, (panel b).

Figure 9. Different choices the model can make in generating the past tense. Each option is executed by the firing of a production rule, but only the two rightmost options actually implement a generalized rule. ACT-R’s production-rule competition and learning mechanisms govern the model’s selection among these options.
Figure 1
Figure 2

**SOURCE**

```
<table>
<thead>
<tr>
<th>ss-cause</th>
<th>ss-effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>causes</td>
<td>causes</td>
</tr>
<tr>
<td>cause</td>
<td>effect</td>
</tr>
<tr>
<td>attracts</td>
<td>revolves</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ss-attractor</th>
<th>ss-attracted</th>
<th>ss-revolver</th>
<th>ss-center</th>
</tr>
</thead>
<tbody>
<tr>
<td>attracts</td>
<td>attracts</td>
<td>revolves</td>
<td>revolves</td>
</tr>
<tr>
<td>attractor</td>
<td>attracted</td>
<td>revolver</td>
<td>center</td>
</tr>
<tr>
<td>sun</td>
<td>planet</td>
<td>planet</td>
<td>sun</td>
</tr>
</tbody>
</table>

ss-causes

ss-attracts

ss-revolves

ss-sun

ss-planet
```

**TARGET**

```
<table>
<thead>
<tr>
<th>at-cause</th>
<th>at-effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>causes</td>
<td>causes</td>
</tr>
<tr>
<td>cause</td>
<td>effect</td>
</tr>
<tr>
<td>attracts</td>
<td>revolves</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>at-attractor</th>
<th>at-attracted</th>
<th>at-revolver</th>
<th>at-center</th>
</tr>
</thead>
<tbody>
<tr>
<td>attracts</td>
<td>attracts</td>
<td>revolves</td>
<td>revolves</td>
</tr>
<tr>
<td>attractor</td>
<td>attracted</td>
<td>revolver</td>
<td>center</td>
</tr>
<tr>
<td>nucleus</td>
<td>electron</td>
<td>electron</td>
<td>nucleus</td>
</tr>
</tbody>
</table>

at-causes

at-attracts

at-revolves

at-nucleus

at-electron
```
Figure 3.
**Figure 4.** Model fits for four representative subjects from Daily et al. (1999). Filled symbols are subject data, open symbols are the model's predictions.
Figure 5. Fits to the serial position data for 4 typical subjects (largest set size only). Filled symbols are subject data, open symbols are the model’s predictions.
Figure 6

Subject 610
W = 0.8

Subject 619
W = 0.9

Subject 613
W = 1.0

Subject 623
W = 1.1
Figure 7

Graph showing the comparison of Data Set 2 (blue line) and Data Set 3 (pink line) across different blocks.

The x-axis represents the block number, and the y-axis represents the data values. The graph illustrates the progression of data sets over the blocks.
Figure 8a
Figure 8b
Figure 9

- Do nothing
- Retrieve exact exemplar
- Analogize from distinct exemplar
- Apply general rule: Add “ed”
- Other rules (phonetically defined)