Stances: Production systems

Christian D. Schunn and **David Klahr**
Department of Psychology
Carnegie Mellon University

To appear in W. Bechtel and G. Graham (Eds.), *A Companion to Cognitive Science*. Blackwell.

Cognitive Universals

"Fiiirre!" If someone were to shout that while you were in the midst of reading this essay, you would, like most people, stop reading and look around the room for the source of the shout, or the fire itself. You would also consider whether or not the likelihood of a fire was sufficiently high to cause you to take appropriate action e.g., locate a fire extinguisher, call the fire department, or leave the room. Of course you have not been sitting around waiting for someone to yell "fire!". You were engaged in some task (unrelated to fires) and yet were able to react to that stimulus. In contrast, if you were a police officer who was being trained at a firing range, you would be doing exactly that (i.e., waiting for someone to yell "fire!"), and when you heard "fire" you would pull the trigger on your weapon, rather than look around for the source of the cry.

In both situations, two kinds of knowledge would be brought to bear in responding. On the one hand you would have facts (e.g., that someone yelled "fire", that you are in the library or on a firing range, etc.), and on the other, you would have skills for how to respond (e.g., how to find an exit, how to fire a gun, etc.). Furthermore, responding could involve both pre-existing knowledge of both types (e.g., that you are on a firing range, or how to fire a gun), and newly acquired knowledge of both types (e.g., that someone yelled fire, or where to look for the exit in this particular room).

This example illustrates the cognitive universals that builders of production system models believe to be fundamentally important aspects of intelligent behavior (both human and artificial). First, at any point in time there are always many possible actions from which one must be

selected, and the selection happens very quickly. For example, the reaction to the shout "Fire!" involves quickly selecting among many possible responses. Second, actions are taken sequentially. For example, leaving the room involves a series of actions (locating the door, moving towards the door, opening the door, etc.). Third, there are two very different kinds of knowledge, *knowing that* and *knowing how*. In our example, you could know that you are currently on a firing range or know that someone yelled "fire". In contrast, you know how to fire a weapon and you know how to exit a room. Your knowing that would determine which of your knowings how would be appropriate in this instance. Fourth, the process of behaving is fully interleaved with the process of learning. In our example, you would learn both things about the current situation and how to act while you were acting, and this learning would influence subsequent behavior. These cognitive universals are summarized in Table 1.

Table 1. Cognitive universals deemed important by production system modelers.

1. Parallel consideration of possible actions
2. Serial execution of actions
3. Distinction between *knowing that* and *knowing how*
4. Simultaneous performance and learning

These universal aspects of behavior led Allen Newell, one of the founding fathers of cognitive science, to propose production systems as an information processing theory of human learning nearly 25 years ago (Newell, 1973). In this chapter, we will present an overview of what production systems are and how they achieve these cognitive universals. Then we will consider what the basic theoretical assertions of production systems are, illustrating these assertions with more detailed discussions of production system components.

What is a production system?

Production systems consist of two basic entities, one for representing objects and features in the world, and one for representing skills or procedures for interacting with the world. Mostly for

historical reasons, the set of entities used to represent objects and features is usually called working memory, but it is more accurate to call it declarative memory (since what psychologists call working memory is typically much more limited in size). Declarative memory contains statements and beliefs about the world. These include things known for a long time (e.g., rooms have red fire exit signs), and more recent knowledge such as assertations about the immediate context, both "external" (e.g., someone just yelled "fire") and "internal" (e.g., my subgoal to get to the exit).

By contrast, productions are rules stated in an 'if-then' form (where the 'if' side is called the *condition* and the 'then' side is called the *action*). For example, one might have the production, "if you hear the word 'Fire!', then look to find the source of the yell". The condition side of the production is a list of entities that must appear in declarative memory (e.g., hearing the word 'Fire!', being in certain place, etc.). When the conditions of a production are true of the current state of declarative memory, then the production is said to *match*. The action side of a production can refer to either behavioral actions (e.g., leaving a room) or new declarative memory elements (e.g., a new current location or a new goal). Thus, declarative memory is both used 1) to decide which productions to apply and 2) to store the results of production actions.

Cognitive Universals and the Components of Production Systems that Implement Them

How do production systems attain the cognitive universals described in the preceding section? For each of the four universals, there is a corresponding major component of productions systems. In the following section, we will discuss those components and show how they attain each of the cognitive universals.

Parallel selection among options—conflict resolution

It is the productions that give a production system the power to be reactive to changes in the environment, and to consider large numbers of responses simultaneously. At any given point in

time, all productions are considered simultaneously to determine which productions are currently applicable (i.e., which productions match the items currently in declarative memory).

What happens when more than one production's conditions are currently satisfied? For example, in the case of someone yelling "Fire!" while a person is trying to read an essay, how does the person know whether to continue reading or to pay attention to the shout? The productions relevant to reading are still applicable, even though the productions relevant to dealing with the shout have also become applicable. The process by which a production system decides among potentially-applicable productions is called *conflict resolution*. An important thing to note about conflict resolution is that it involves a theory of decision-making: how does one decide among different potential actions? For production systems, these decisions are viewed as an integral part of the computational framework.

A number of conflict resolution schemes have been used as production systems have evolved over the past 25 years. The early schemes used various combinations of the following approaches: 1) favoring productions whose conditions refer to declarative memory elements that have been most recently added or changed; 2) favoring productions with many conditions (i.e., more specific) over productions with few conditions (i.e., more general); or 3) simply setting a rank ordering among productions (e.g., fire-attending productions are always considered first). These schemes are difficult to implement because the ordering of productions frequently needs to change from one task to another, and this change in ordering is not well-predicted by recency or specificity. Nonetheless, these are usually the approaches that are tried first in building a production system model because they are fairly simple.

Alternative schemes order productions according to how often they have been used in the past (with preference given to those that have been used most often) and how often the productions have been used successfully. In this way, production order can be adapted to different tasks via experience with each task. We will return to this issue later, when we discuss learning in production systems.

A more radical approach to conflict resolution is to not do it at all: instead apply all rules that could apply to the current situation in parallel. In one such scheme, the productions only make suggestions about what to do (applying knowledge from past experiences), and then some other conflict resolution scheme has to decide among those suggestions. In another, all productions fire, and re-allocate a limited pool of activation "resources" to the declarative memory elements on their action sides. But, in effect, all such schemes merely defer the conflict resolution decision and delegate it to some other part of the architecture. The basic issue of conflict resolution and the ultimate sequential selection of motor actions cannot be avoided in a production system framework.

Serial execution of actions—subgoaling

How are production systems able to follow a sequence of steps in service of some goal? There are two answers to this question. First, productions can refer to declarative memory items that are the result of a previous production action. For example, in adding 15 and 19, the production for setting the carry in the tens column can only occur after the production that retrieves the sum of nine and five.

The other answer is that the conditions of productions can also refer to goals and subgoals. For example, one might have the production, "if my goal is to leave the room then set a subgoal to locate the exit". With a number of such productions, a production system can thus also follow a long sequence of steps by setting a sequence of subgoals.

Knowing that vs. knowing how—declarative memory vs. productions

Production systems also easily capture the distinction between *knowing how* and *knowing that* —the distinction between items in declarative memory and productions corresponds to the distinction between *knowing that* and *knowing how*. There is considerable psychological and neurophysiological evidence suggesting that the human brain treats these kinds of knowledge separately. For example, some brain damaged patients can learn new cognitive skills yet are completely unable to learn new facts (see also DEFICITS AND PATHOLOGIES). This distinction also explains how people can behave in a rule-like fashion without explicitly knowing

the rules. For example, people usually have little difficulty applying the grammatical rules of their native language (procedural knowledge) without being able to explain that grammar (declarative knowledge) .

Simultaneous performance and learning—production rule learning and tuning

A fundamental challenge for both builders of AI systems and cognitive modelers is to produce a model that can both perform some task and learn (i.e., improve with experience). Production systems provide important insights into, and strong theoretical assertions about, how learning might occur in a performance system, and so we shall explore this issue in some depth.

Learning in production systems. Most of the production systems of early 1970s were incapable of learning. In the few systems that did learn, the mechanisms for controlling when learning occurred were not especially powerful and rarely produced any serious accounts of human learning. At that time, cognitive scientists built models of a particular task at a particular level of competence. That is, they built models of how a person (or machine) might perform some task but did not address the question of how the person (or machine) might acquire those abilities. The modelers simply assumed that some learning mechanism was capable of producing the models that they built, and that the (eventual) addition of the learning mechanism would not interfere or interact with the performance of their models.

When cognitive scientists first became interested in learning and developmental issues in the late 1970s, they built a series of models, each of which described performance at a particular level of competence (or developmental stage). However, they did not offer computational accounts of how a model of one stage might evolve into the next stage. The belief at that time was that the learning mechanisms could be induced from a solid understanding of performance at the different stages. Further, it was assumed that the learning mechanisms, once developed, could simply be applied to a model of one performance level to produce a model of the next performance level. However, the efficacy of this approach has not yet been demonstrated, and the emergent view is that the early assumptions were probably incorrect.

As workable mechanisms of learning were developed in the 1980s, modelers discovered that there was a problem: under some circumstances, the addition of learning mechanisms radically altered the functioning of the performance model. That is, when learning was "turned on," some models either ceased to function because learning immediately moved them into an unworkable state, or they had to be rewritten from scratch to produce learning that noticeably improved system performance. Moreover, the workable learning mechanisms produced a more limited set of kinds of productions and memory elements. That is, many of the productions that were built into the early, static models were simply not learnable by the learning mechanisms. Of course, one might always believe that learning algorithms developed in the future might make these unlearnable productions learnable. However, at the very least, these problems undermined the early notion that one could build a competence model first, and then the learning model second. Nonetheless, this remains an unresolved issue, and many productions system models are still being built without an active learning mechanism.

Mechanisms for learning new productions. Production systems are based on the assumption that the fundamental unit of thought is the production, and that competence undergoes discrete jumps as new productions are learned. Most importantly, the learning is assumed to be specific to a particular production rather than general improvement across numerous productions at once. Therefore, specifying how new productions are learned is a central theoretical and practical feature of a production system.

One production-learning mechanism is compilation, in which a new production is produced that does, in one step, the action of several productions. An important variant of compilation is the chunking algorithm used in the Soar production system (Newell, 1990). The chunking algorithm determines which pieces of declarative knowledge were used by a recently successful sequence of productions and then creates a new production that looks for those declarative memory elements and directly produces the desired conclusion. For example, if the system determines through trial and error (i.e., many production firings) that a certain key opens a

particular door, then it will create a production that recognizes the relevant features of the key and door and directly selects the correct key.

Another mechanism is analogy. This mechanism, used in the ACT-R production system (Anderson, 1993), converts examples in declarative memory into productions. In particular, the system tries to make an analogy between the current goal and the corresponding goal in an example, and creates productions that achieve the current goal using steps analogous to the ones used to achieve the example goal. For example, if the system was trying to solve an addition problem and had recently processed an example, it would try to make analogies between the corresponding steps and states to get the answer to the current problem. As a result of this analogy process, new productions are created that then can be used the next time in lieu of the analogy process.

Other production learning mechanisms create productions by combining or mutating existing productions. For example, one might add or delete conditions to a production, thereby making it more or less situation-specific. Classifiers are a special case of this process (Holland, 1986). They consist of a set of rules for classifying instances into different categories. New rules are created by randomly mutating some of the conditions of existing rules. The new rules are then strengthened or weakened according to their effectiveness in classifying instances.

Although there are several different mechanisms for learning new productions, current production systems that learn use only one production-creation mechanism, and each uses a different one. Why is this? It is not that the learning schemes are incompatible. Indeed, one could easily imagine a system that involves more than one of the production-creation mechanisms.

There are several factors that were important to the decision to construct each of the production system architectures with only one production-creation mechanism. First, using only one mechanism is parsimonious. This feature is desirable in a scientific theory of human behavior because the more components a theory has the more difficult it is to test. Second, having only one mechanism provides a strong test of the power of that mechanism. At this point, the exact limitations and potential of each of the learning mechanisms have not yet been fully

explored. In particular, it is not clear that more than one mechanism is required (which is surprising given how different the production-creation schemes seem, as we have seen). Finally, for computer scientists interested in performance factors, using only one mechanism reduces the computational demands—the learning algorithms tend to be very computationally intensive and using more than one learning mechanism would severely impact system performance.

Mechanisms for tuning productions. A frequent finding in studies of learning is that performance continues to improve with practice. That is, despite the fact that the learning is specific to a particular subskill, one usually finds continual speed-up on that subskill with practice. How might a production system achieve this gradualness in learning if learning new productions creates discrete jumps in performance? There are two solutions to this dilemma that have been incorporated into production system architectures. One answer is to formulate productions at a very fine grained level of detail such that many productions are required to produce each external action. In such a scheme, the addition of each production would produce only a minor improvement in performance. Another solution is to associate with each production performance parameters that cause productions to perform slowly, suboptimally or infrequently when they are first created, and then gradually become faster, more efficient, or more frequent. It is important to note that even with the addition of these more graded features, the fundamental unit of learning and transfer in a production system architecture is the production.

There are two primary mechanisms for tuning production performance parameters. First, productions can be strengthened each time they are used and weakened over time with disuse. This production strength can be used to determine how fast a production will be selected or how likely that production is to be selected. Alternatively, productions can be tuned by keeping track of how often they lead to problem-solving success or failure. Under such a scheme, productions are more likely to be selected if they usually lead to success than if they usually lead to failure.

Meta-Theoretical Assumptions

Although particular production systems involve many different theoretical assumptions, there are some core assumptions that are common across all production systems. They are listed in Table 2. In this section we will discuss meta-theoretical assumptions involved in the claim that there is a correspondence between the cognitive universals in Table 1 and the production system commonalties in Table 2.

The first three assumptions have already been treated in some depth. It is worth noting that psychologists using production systems differ in terms of how strongly they hold these three assumptions. Some view production systems as a loose metaphor for understanding the activities of the mind (i.e., the mind generally acts as if it were a production system). Others use production systems to make an unambiguous theoretical assertion about the nature of the mind: the mind is a production system. The latter position is stated forcefully by John Anderson in his book on the ACT-R theory of human thought (Anderson, 1993, p. 1): "What is happening in the human head to produce human cognition?" he asks rhetorically. His answer is unequivocal: "Cognitive skills are realized by production rules. This is one of the most astounding and important discoveries in psychology...". In the rest of this section, we explain the other four core theoretical assumptions.

Table 2. Core theoretical assumptions of production systems.

1. Procedural knowledge is stored as if-then rules (called productions).
2. The execution of a production is the fundamental unit of thought.
3. The acquisition of a production is the fundamental unit of learning.
4. Declarative memory is represented by frame-like propositions.
5. The results of computation are stored in a (potentially temporary) declarative memory.
6. Knowledge is (mostly) modular.
7. All intelligent behavior can be captured in one cognitive architecture.

The frame-like structure of declarative memory elements

Production systems typically represent declarative memory items in terms of entities called frames or schemas. Each frame is simply a list of attribute-value pairs in which attributes represent dimensions (e.g., color, size, location, etc.) that take on the values of the entity that the memory item denotes. For example, a declarative memory item representing some visual object might have a slot for the object's color, another slot for the object's shape, and yet another slot for the object's position. Different kinds of items can have a different set of slots. One can think of the different combinations of slots as representing different object categories as well as relations between objects.

Such frame-like memory structures provide precise and powerful representations of things in the world, including objects, relations between objects, and relations between relations. This representational power is especially important when one tries to build systems that do complex problem-solving. However, this form of memory representation often has difficulty in situations where the knowledge is more continuous and less hierarchical (e.g., low-level vision).

Interestingly, the particular organization of declarative knowledge in a production system usually does not have immediate consequences for the system's performance. That is, one can get similar behavior from very different organizations of memory items. For example, one could

use a single declarative memory element with many slots representing all that one knew about some individual, or one could have a large number of declarative elements each representing individual facts about that individual. A production system could function equally well with either representation scheme. The reason is that what matters is primarily whether information is contained somewhere in memory, and not so much which information is stored together. If a different organization is selected, the productions are re-written to accommodate the new structure. It is important to note, however, that in the production systems that learn, the organization of declarative memory can have strong influences on performance.

Results of computation are stored in a (potentially temporary) declarative memory

As noted earlier, declarative memory does more than represent objects and features in the environment—it also serves to represent the intermediate results for tasks that cannot be solved all in one step. For example, when mentally multiplying two two-digit numbers, you must mentally store the intermediate products. Thus, a production system for doing this task would contain some declarative memory elements that represented the (external) multiplicands, as well as other declarative memory elements to represent the (internal) partial products. Another way in which declarative memory serves this function is in storing goals and subgoals.

This function of declarative memory raises another important and related question—are these declarative memory elements permanent? In particular, are all the intermediate products of complex tasks erased after the task is complete, or do they leave long-lasting declarative memory elements? The basic problems is that the more information there is sitting around in declarative memory, the more likely it is that many productions will be satisfied simultaneously. This, in turn, complicates the process of conflict resolution. Moreover, this issue relates to a common psychological finding considered to be a basic feature of human cognition: the limited nature of short term or working memory.

Production system designers have proposed a wide range of answers to these questions. At one extreme are systems in which items stay around forever once they are created. At the other extreme are systems in which items are deleted once the system moves onto the next task. The

only way in which such systems can remember facts over long delays is to have productions that re-create the facts into declarative memory when they are required. Intermediate between these two extreme approaches to dealing with the duration of declarative memory elements are those systems in which the elements vary in activation (which in turn determines how available or easily retrieved they are). The activation increases each time the represented facts or items are encountered and decays with time after each encounter. At first blush, it would seem obvious that the vast body of empirical evidence from experimental studies of human memory could be used to select among these approaches. However, it turns out that one can produce the effect of a limited working memory using any of these schemes, and the ultimate answer will require both further experimental evidence and detailed modeling of those experimental results.

Knowledge is (mostly) modular

How does knowledge interact, and how does learning generalize? Production systems provide strong answers to these fundamental questions: (a) learning occurs at the unit of productions; (b) transfer from one situation to another occurs to the extent that the same productions are applicable in both situations. This assumption about the modularity of productions allows production system designers to determine—via a detailed analysis of a task domain or a careful encoding of the verbal and behavioral protocols of human problem solvers, or both—what the individual productions are and simply add them to the system. One does not have to decide "where" to put a production; its conditions define when it will be used.

Because of their modularity, productions systems scale up well to complex tasks. That is, not only do production systems function well on small, simple tasks, but they also function well in more realistic environments involving many subtasks and thousands (or more) of bits of knowledge. For example, there is a production system called TacAir Soar which has tens of thousands of productions, can fly a simulated plane in a dogfight (in real time) while doing language comprehension and production, and is capable of providing a verbal summary of the mission afterwards (Tambe, Johnson, Jones, Koss, Laird, Rosenbloom, & Schwamb, 1995).

It is important to note that this modularity is not perfect. In particular, one often finds that some productions are not entirely independent of all other productions. As noted earlier, this can be particularly troublesome in production systems that learn because whenever one has a sequence of productions that are needed to perform one task, then the removal or addition of a production can affect the function of the other same-task productions. For example, if a newly added production changes the internal goal in the beginning or middle of a sequence of steps, then the production which is to compute the next step might be affected. Moreover, many productions rely on other productions to set up things in a very particular form so they can be accessed in just the right way. Thus, in many complex situations, the modularity of the production system is only partial and groups of productions must be constructed with careful consideration given to their potential interactions. The way in which cognitive scientists deal with this complication is to do careful task analyses of the steps required by a task and of the relationship between them. This is one of the most difficult aspects of building production system models.

For computer scientists, whose goal is to build intelligent systems, rather than explain human intelligence, this partial non-independence is usually a nuisance. For psychologists, however, this aspect reflects known properties of human learning. For example, learning how to play racquet ball can interfere with one's tennis abilities (see also MODULARITY).

All intelligent behavior can be captured in one cognitive architecture

From the perspective of a psychologist trying to develop a scientific theory in the form of a computational model, production systems have a very important advantage: it is possible to see how one underlying computational framework (called a cognitive architecture) could give rise to all cognitive activities. The advantage of using such a unified architecture was clear even in the early days of cognitive science. Without a unified architecture, a modeler had to develop a theory from scratch for each new task. Models developed in such a fashion had a very ad hoc flavor, and it was never clear how such models could be combined with models of other cognitive activities to produce overall behavior. Production systems were first proposed just as this issue of

avoiding a plethora of ad hoc models was raised, and, even today, proponents of the unified architecture idea are for the most part production system modelers (see also COGNITIVE ARCHITECTURES).

Conclusion

In this chapter, we have covered the set of cognitive universals that production systems were designed to capture, the basic components of production systems, and their core meta-theoretical assertions. In addition to those common features, there are components unique to particular production systems, including spreading activation between declarative memory elements via associative links, capacity limitations on activation, and learning in declarative memory. These design parameters represent different theoretical assumptions and they lead to important performance differences. Although we do not have space to discuss them here, we mention them to convey the sense in which production systems are to be conceived as tools under development, rather than fixed theoretical statements.

However, there are a number of areas in which production system models have already done very well, and are arguably the strongest (and occasionally only) models in those areas. These areas are almost exclusively instances of higher-level cognition and generally require the coordination of many kinds of knowledge. They include learning mathematical skills such as algebra and geometry, learning computer programming skills, language comprehension, scientific discovery, and many other forms of high-level, complex action and reasoning. For more detailed descriptions of these examples, we refer interested readers to Anderson (1993), Halford & Simon (1995), Just & Carpenter (1987), and Steier & Mitchell (1996).

Bibliography

Anderson, J. R.: Rules of the Mind (Hillsdale, NJ: Erlbaum, 1993).

Halford, G., & Simon, T., eds: Developing Cognitive Competence: New Approaches to Process Modeling (New York: Academic Press, 1995).

Holland, J. H.: Escaping Brittleness: The Possibilities of General-Purpose Learning Algorithms Applied to Parallel Rule-Based Systems; eds. R. S. Michalski, J. C. Carbonell, & T. M. Mitchell, Machine Learning: An Artificial Intelligence Approach, Volume II (1986), pp. 593-623.

Just, M. A., & Carpenter, P. A.: The Psychology of Reading and Language Comprehension. (Boston, MA: Allyn and Bacon, 1987).

Newell, A.: Production Systems: Models of Control Structures; ed., W. G. Chase, Visual Information Processing (New York: Academic Press, 1973), pp. 463-526.

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

Steier, D., & Mitchell, T., eds.: Mind Matters: A Tribute to Allen Newell. (Mahwah, NJ: Erlbaum, 1996).

Tambe, M., Johnson, W. L., Jones, R. M., Koss, F., Laird, J. E., Rosenbloom, P. S., & Schwamb, K. (1995). Intelligent Agents for Interactive Simulation Environments. AI Magazine, 16(1).