Cognitive Architectures and Principles of Behavior[1]

Pat Langley
Stellan Ohlsson
Robert Thibadeau
Robert Walter
The Robotics Institute
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213 USA

Introduction: Taxonomies and Principles

Taxonomies play an important role in emerging fields of science, since they identify significant dimensions along which the entities studied by those fields can differ. Yet one must eventually move beyond simple taxonomies to formulate *principles* that relate these dimensions of variation to observed behavior. The emerging field of Cognitive Science is concerned with the behavior of intelligent entities, both human and artificial. However, Cognitive Science is notably lacking in both taxonomies for the structure of intelligent systems, and in principles which relate such structures to intelligent behavior. In this paper, we describe an evolving taxonomy of cognitive architectures, and propose some initial principles based on this taxonomy.

Cognitive Science and its sister discipline, Artificial Intelligence, have generally been empirical sciences, in that they have spent considerable time collecting *examples* of intelligent behavior, through experiments with humans and through constructing simple intelligent artifacts. This work has been worthwhile and should continue, but eventually we must begin to develop *theories* of intelligence that cover not only human intelligence, but cognitive behavior in general. Since different intelligent entities may rely on different cognitive mechanisms, Ohlsson [1] has proposed that a general theory of intelligence must be concerned with the *relation* between such mechanisms and the form of intelligent behavior that results.

In science, a researcher often limits his attention to ensure progress, and in this case we have focused on the class of cognitive architectures known as *production systems*. Production systems were first proposed as models of the human information processing architecture by Newell and Simon [2]. Since that time, they have been used to simulate a variety of intelligent behavior, ranging from problem solving to natural language understanding to cognitive development. Production system schemes have a number of features that make them attractive candidates for cognitive architectures, independent of their value as models of human behavior. For instance, they seem to be a viable compromise between the stimulus-response approach of behaviorism, and the goal-driven approach of cognitive psychology. In addition, the relative independence of the condition-action rules making up a production system program lends itself to modeling the learning process, since interaction between new and old components will be minimal.

Dimensions of Production System Architectures

Our research goal has been to identify the significant dimensions along which production system architectures may vary. This has resulted in a formalism, PRISM2, which specifies a set of such dimensions, along with possible values for each of these dimensions. Using this formalism, one may succinctly describe architectures that have been explored by other researchers, as well as architectures that have never before been examined. This allows comparison of the differences between various architectures, and should facilitate communication between researchers in the area.

Since we are concerned with the relation between architectures and intelligent behavior, we have implemented PRISM2 in such a way that one can "run" an architecture in conjunction with a particular production system program. Thus, PRISM2 has some of the characteristics of a programming language,

---

[1]This research was supported by Contract N00014-83-K-0074 from the Office of Naval Research. We would like to thank David Nicholas, Robert Neches, and Rolf Pfeifer for their assistance in the early stages of our work on PRISM2.

though it actually defines an entire *class* of production system languages. In this context, a given architecture can be viewed as providing "free" control structure that need not be specified explicitly in the program itself. Let us examine the dimensions of architectural variation supported by the PRISM2 formalism:

- *The structure of memory.* PRISM2 provides for multiple declarative and production memories, allowing "flat" production system schemes, hierarchically organized systems, or more exotic control structures. Moreover, each memory may have different characteristics. E.g., PRISM2 allows arbitrary numeric attributes (such as strength, activation, and affect) to be associated with each memory.
- *Decay and forgetting.* Production system architectures differ in the manner in which memory elements decay over time, and in the details of the forgetting process. In PRISM2, elements in a given memory may decay by a fixed amount on every cycle, or as a function of the number of elements entering memory. The formalism supports a number of alternate decay and forgetting methods.
- *Retrieval by spreading activation.* Production system architectures differ in the manner by which they retrieve forgotten elements through spreading activation. E.g., activation may decay according to different functions as it spreads through adjacent elements, ceiling effects may occur, and the threshold below which activation may not spread can differ. PRISM2 supports a variety of constraints on spreading activation.
- *The match process.* Production system architectures differ in their matching abilities, and PRISM2 supports a variety of different matching styles. E.g., conditions may match against embedded structures, find sequences of symbols, and match against lists as though they were sets. In addition, the user may require one-to-one mappings between conditions and memory, or allow many-to-one matches to occur.
- *Conflict resolution.* Production system architectures differ in the conflict resolution methods they employ for selecting among competing instantiations of rules. Three relevant dimensions of conflict resolution are:
  - *Ordering strategies.* The architecture orders instantiations of productions along some dimensions, such as recency of matched elements, or specificity of the matched rules.
  - *Selection strategies.* The architecture selects one or more instantiations based on the resulting ordering; e.g., the best instantiation may be selected, or all those above a certain threshold may be chosen.
  - *Refraction strategies.* The architecture may remove some instantiations permanently; e.g., it may remove all instantiations that applied on the last cycle, or all instantiations currently in the conflict set.

  The PRISM2 formalism supports many different combinations of ordering, selection, and refraction strategies for implementing alternate conflict resolution methods.
- *Learning methods.* Production system architectures differ in the processes they use to learn new condition-action rules. Common methods include:
  - *Discrimination learning*, in which errors lead to more specific rules based on differences between positive instances and negative instances of the errorful production.
  - *Generalization learning*, in which specific productions with similar structures lead to more general rules based on features common to the original rules.
  - *Composition*, in which two or more rules that tend to fire in sequence are combined into a more complex production that leads to the same results.
  - *Proceduralization*, in which a specific version of a general rule is based on the current instantiation of that production.

  PRISM2 supports each of these learning methods, as well as providing the ability to modify the detailed characteristics of each method. The learning methods may be used in conjunction or in isolation.

The basic organizing principle for specifying PRISM2 architectures is the architectural *template*. Different templates are available for specifying the characteristics of declarative memories, production memories, and action side functions responsible for adding new elements, decay and forgetting, spreading activation, and learning new rules. Each template has an associated set of *parameters*, whose values determine the exact behavior of that component of the system.

For example, the spreading activation template contains three main parameters. The first of these, **spread-from-element**, contains a list of steps taken when activation spreads out from a memory element; this might include actions such as dividing the available activation by the number of symbols in the element and causing this activation to decay by a certain amount. The second parameter, **spread-through-symbol**, specifies a list of steps taken as activation spreads through a symbol contained in a memory element. This can include actions such as dividing up the activation by the number of other memory elements containing this symbol, leading to a form of the fan effect. Finally, the parameter **spread-to-element** specifies a list of actions carried out when activation spreads to a new memory element. This may include tests for whether to continue spreading activation, as well as constraints on the amount retained by the element, leading to a ceiling effect.

Towards Principles of Intelligent Behavior

The flexibility of the PRISM2 framework has allowed us to experiment with alternate production system architectures. To date, most of our experiments have involved alternate conflict resolution schemes and different methods for learning new productions, and we shall draw our examples of principles from these areas. While we would not claim that the PRISM2 formalism was necessary for generating these principles – in fact, there was a strong analytical component in both cases – it has certainly helped us in clarifying and testing our ideas. In our future work, we hope to examine the relation between other dimensions and behavior, and would welcome any other research with similar goals.

The first example relates conflict resolution strategies to the notion of *search*. In recent years, a number of production system models have been implemented in which the rules play the part of operators for moving through some problem space [3, 4]. Within this framework, the conflict resolution strategy used by the system determines the form of search it carries out. For example, Young [5] has proposed the following principle:

- *Recency-based conflict resolution schemes lead to depth-first search behavior with automatic backtracking.*

To be more specific, depth-first search behavior results when the architecture prefers instantiations matching against elements added to memory more recently, and when the single best instantiation is then selected for application. Automatic backtracking also results, provided that refraction removes applied instantiations from the conflict set. This relation has been known informally among production system users for years, but we believe it is important to note its status as a basic principle of cognitive architectures.

Since other conflict resolution schemes are possible, an obvious question is whether other search strategies arise from alternate architectures. Another popular conflict resolution scheme allows all instantiations to apply in parallel, unless they have been applied on an earlier cycle [6]. Upon reflection, this strategy leads to a second well-known search method – breadth-first search. Thus, we can formulate a second principle relating architecture to behavior:

- *Conflict resolution schemes involving parallel firings lead to breadth-first search behavior.*

In this case, no backtracking is required, and refraction is used only to keep instantiations used earlier from applying again, since there are no other constraints on the selection process. Presumably, other relations between conflict resolution and search methods exist, and these will be discovered as researchers further explore of the space of architectures.

Our second example involves the area of learning methods. In particular, we have been concerned with the distinction between *discrimination* learning, in which one moves from general rules to more specific ones, and *generalization* learning, in which one moves from specific rules to more general ones. The most common form of discrimination involves creating some variant of an existing rule containing additional conditions. Since such a learning system begins with simple rules and generates more complex ones only when errors of commission occur, we arrive at the principle:

- *Given two rules of different complexity, a discrimination-based learning system will master the simpler rule before the more complex one.*

In contrast, generalization involves the opposite process of creating a production with fewer conditions than an existing rule. Since such a learning system begins with complex rules and generates simpler ones only

when errors of omission occur, we arrive at another principle:

- *Given two rules of different complexity, a generalization-based learning system will master the more complex rule before the simpler one.*

These complementary principles relate learning methods to the rate at which rules of different complexity will be mastered. Although details are absent, even such global statements can be very useful. For instance, empirical studies of human language acquisition suggest that more complex function words are mastered later than simpler ones, suggesting that discrimination learning is a more likely explanation than generalization [7].

Discussion

In the preceding pages, we examined some principles that relate characteristics of the architecture – conflict resolution methods and learning mechanisms – to aspects of intelligent behavior – search strategies and rates of mastery. These principles are explanatory in the sense that they account for behavior in terms of underlying components, just as physical principles account for observed phenomena in terms of inferred properties. However, note that one cannot begin to formulate such principles until one has some ideas about the nature of the components underlying behavior. This is the reason we have focused on developing PRISM2, a formalism which allows us to explore the space of cognitive architectures, to represent the differences between architectures explicitly, and to actually test specific production system architectures in particular domains. The dimensions of variation supported by PRISM2 provide us with a taxonomy of architectural types, which we can then use in formulating principles of behavior.

Admittedly, the principles we have examined are only a beginning, and we are far from a complete theory that relates architectural components to aspects of intelligence. Our principles were intended mainly as examples of relations that one can express using the PRISM2 formalism, and as examples of what we believe should be a more common goal in our developing field. In fact, some readers may disagree with the principles themselves [8], and we would welcome suggestions for modifications and improvements. However, we hope to have convinced the reader that Cognitive Science is ready to begin formulating such principles, and that other researchers will join us in identifying the varieties of cognitive architectures, and in the more long range search for a general theory of intelligent behavior.

References

1. Ohlsson, S. "Mechanisms, behaviors, principles: A time for examples." *AISB Quarterly 48* (1983), 24-25.

2. Newell, A. and Simon, H. A.. *Human Problem Solving.* Prentice-Hall, Inc., Englewood Cliffs, N.J., 1972.

3. Ohlsson, S. A constrained mechanism for procedural learning. Proceedings of the Eighth International Joint Conference on Artificial Intelligence, 1983, pp. 426-428.

4. Laird, J. E. and Newell, A. A universal weak method: Summary of results. Proceedings of the Eighth International Joint Conference on Artificial Intelligence, 1983, pp. 771-773.

5. Young, R. M. Architecture-directed processing. Proceedings of the Fourth Annual Conference of the Cognitive Science Society, 1982, pp. 164-166.

6. Thibadeau, R., Just, M. A., and Carpenter, P. A. "A model of the time course and content of reading." *Cognitive Science 6* (1982), 157-203.

7. Langley, P. "Language acquisition through error recovery." *Cognition and Brain Theory 5* (1982), 211-255.

8. Bundy, A. "Superficial principles: An analysis of a behavioural law." *AISB Quarterly 49* (1983), 20-22.