A Cognitive Architecture for Physical Agents

Pat Langley, Dongkyu Choi, and Daniel Shapiro
Computational Learning Laboratory
Center for the Study of Language and Information
Stanford University, Stanford, CA 94305 USA

Abstract

In this paper we describe Icarus, a cognitive architecture for intelligent physical agents. We contrast the framework's assumptions with those of earlier architectures, illustrating our points with examples from an in-city driving task. Key differences include primacy of perception and action over problem solving, separate memories for categories and skills, a hierarchical organization on both memories, strong correspondence between long-term and short-term structures, and use of expected values to guide behavior. We support claims for Icarus' generality by reporting our experience with the driving domain and three other tasks. In closing, we discuss some limitations of the current architecture and propose extensions that would remedy them.

Introduction and Motivation

A cognitive architecture (Newell, 1990) specifies the infrastructure for an intelligent system that remains constant across different domains and knowledge bases. This infrastructure includes a commitment to formalisms for representing knowledge, memories for storing this domain content, and processes that utilize and acquire the knowledge. Research on cognitive architectures has been closely tied to cognitive modeling, in that they often attempt to explain a wide range of human behavior and, at the very least, desire to support the same broad capabilities as human intelligence.

In this paper we describe Icarus, a cognitive architecture that builds on previous work in this area but also has some novel features. Our aim is not to match quantitative data, but rather to reproduce the qualitative characteristics of human behavior, and our discussion will focus on such issues. The best method for evaluating a cognitive architecture remains an open question, but it is clear that this should happen at the systems level rather than in terms of isolated phenomena. We will not claim that Icarus accounts for any one result better than other candidates, but we will argue that it models facets of the human cognitive architecture, and the ways they fit together, that have been downplayed by other researchers in this area.

We discuss the distinguishing features of Icarus in the section that follows, but we should first note that they have resulted from our focus on *physical* agents. We can clarify this concern with an example domain – in-city driving – that involves cognition but in which perception and action also play central roles. In particular, consider the task of a UPS driver who must deliver packages to indicated addresses in an unfamiliar area. The driver must attempt to achieve his multiple delivery goals, which themselves involve a combination of perception, action, and reasoning, while obeying the rules of driving and avoiding collisions with other cars.

To support our research on such complex tasks, we have implemented a simulated environment for in-city driving that simplifies many aspects but remains rich and challenging. Objects take the form of rectangular parallelepipeds that sit on a Euclidean plane. These include vehicles, for which the positions, orientations, and velocities change over time, as well as static objects like road segments, intersections, lane lines, and buildings. Each vehicle can alter its velocity and change its steering wheel angle by setting control variables, which interact with realistic laws to determine each vehicle's state. The physics for collisions is simplified, with vehicles exchanging momentum along their lengthwise axes.

Most vehicles are drones controlled by the simulator, but one vehicle is driven by an Icarus agent, which has access to the same effectors and can only sense objects closer than 60 feet. The system perceives other vehicles (with no occlusion) and the corners of buildings, both described in agent-centered polar coordinates that give the object's distance, angle, relative velocity, and orientation. The Icarus agent also perceives its distance and angle with respect to lane lines, and some of its own properties, like speed and steering wheel angle. To support the package delivery task, the agent can perceive the street, address, and cross street for each package it carries, along with the current street name, the upcoming cross street, and the address associated with visible building corners. We also provide the system with top-level intentions to deliver these packages to their destinations.

Despite the idealized nature of this environment, it forces us to take seriously the goal of integrating cognition with perception and action in ways that are consistent with our knowledge of human behavior. Thus, we will use this task domain as a running example throughout our discussion of ICARUS' features in the next section. However, generality is a key criterion for a successful cognitive architecture, so we follow this with initial results on three additional domains. We conclude with comments on ICARUS' relation to other work in AI and our plans for future research.

Distinctive Characteristics of ICARUS

Our framework shares many features with traditional cognitive architectures, such as Soar (Laird et al., 1987) and ACT-R (Anderson, 1993). These include a commitment to symbolic representation of knowledge, utilization of pattern matching to select relevant knowledge elements, and organization of behavior into a recognize-act cycle. However, ICARUS also has some distinctive characteristics, which we contrast here with the assumptions in these established frameworks.

Primacy of Perception and Action

Most cognitive architectures draw heavily on results from the study of human problem solving. This influence is most apparent in Soar, which makes an explicit commitment to Newell's (1990) hypothesis that all cognitive behavior can be cast as search through a problem space. Anderson's (1993) ACT-R framework does not take as strong a position as Soar on this issue, but most ACT-R models emphasize cognitive over sensory-motor activities, following the paradigm set by early models of human problem solving.[1]

In contrast, ICARUS is concerned centrally with intelligent agents that exist in a physical environment. Our work to date has used only simulated worlds, but they are separate and distinct from the cognitive systems, which must perceive it through sensors and influence it through effectors. At the same time, we do not reject theories of human problem solving, as they reflect important phenomena that deserve explanation. However, we hold that problem-solving activities are not primitive but rather are built on top of, and integrated with, more primitive activities for perception and action.

As noted earlier, our ICARUS program for the in-city driving environment perceives a variety of object types, each described as numeric attributes in agent-centered polar coordinates. The system can affect its own situation through effectors that alter speed, turn the steering wheel, and deposit a package at the current location. Our typical runs involve a three-block by three-block city, with five buildings on each side of each block, which provides a reasonably complex environment. The resulting system exhibits much the same mixture of perception, inference, decision making, and action that humans demonstrate when driving.

[1]Kieras and Meyer's (1997) EPIC gives more emphasis to peripheral processes, but the majority of cognitive architectures are designed with mental processing in mind.

Table 1: Some ICARUS concepts for in-city driving, with variables indicated by question marks.

```
(in-rightmost-lane (?self)
 :positives ((on-right-side-of-road ?self)
             (line-to-left ?self ?line))
 :negatives ((lane-to-right ?self ?anyline)))
(lane-to-right (?self ?line)
 :percepts  ((lane-line ?line dist ?dist))
 :tests     ((> ?dist 0)(< ?dist 10)))
(line-to-left (?self ?line)
 :percepts  ((lane-line ?line dist ?dist))
 :tests     ((< ?dist 0)))
```

Separation of Categories from Skills

Another common feature of cognitive architectures is a commitment to a single representation for long-term knowledge. This typically takes the form of production rules, each of which specifies the conditions under which it will match and the actions to be carried out upon execution. Although production systems have been quite successful in modeling many aspects of human cognition, they borrow key ideas from behaviorist psychology and retain a strong action-oriented flavor, even though the actions are primarily mental ones. Even in ACT-R, which distinguishes between a procedural rule memory and a declarative memory of facts, the latter serves primarily as a source of elements for short-term memory.

However, cognition involves more than execution of mental procedures; it also includes the recognition of *categories* and drawing of associated inferences. One can model categorization using production systems, but we believe that concepts serve a different function than procedures and are best handled separately. We should note that many researchers seem to agree; except for those who start from such a position, few cognitive models of categorization are cast as production systems.

In response, ICARUS incorporates two separate long-term stores. A conceptual memory contains Boolean concepts that encode its knowledge about general classes of objects and relations. Each concept definition specifies the concept's name and arguments, along with a `:percepts` field that describes observed perceptual entities, a `:positives` field that states lower-level concepts it must match, a `:negatives` field that states concepts it must not match, and a `:tests` field that specifies numeric relations it must satisfy. Table 1 shows some concepts from the driving domain.

In contrast, ICARUS' long-term skill memory encodes knowledge about ways to act and achieve goals. Each skill has a name, arguments, and a variety of fields.[2] These include an `:objective` field, which specifies a conjunction of concepts for the situation the

[2]Actually, each skill can have multiple definitions, much as Prolog allows multiple Horn clauses with the same head. Different versions of a skill must have the same name, arguments, and objective, but they can differ in other fields.

Table 2: Some Icarus skills for in-city driving.

```
(drive-to-address (?self ?package)
 :objective ((at-address ?package))
 :ordered   ((drive-to-street ?self ?package)
             (continue-to-address ?self ?package)
             (slow-down-for-stop ?self ?package)))
(continue-to-address (?self ?package)
 :objective ((approaching-address ?package))
 :requires  ((on-street ?package))
 :unordered ((speed-up-for-cruise ?self)
             (turn-around-for ?self ?package)))
(speed-up-for-cruise (?self)
 :percepts  ((self ?self speed ?spd cruise ?cspd)
             (corner ?corner))
 :requires  ((slower-than-cruise-speed ?self)
             (steering-wheel-straight ?self)
             (centered-in-lane ?line))
 :actions   ((*speed-up (- ?cspd ?spd)))
 :features  (?spd)
 :weights   (-0.25)
 :constant  5.0)
```

skill is intended to achieve, a `:start` field that specifies the situation in which one can initiate the skill, and a `:requires` field that must hold throughout the skill's execution. For example, Table 2 shows the skill *continue-to-address*, which has the objective of approaching the delivery address and requires that it already be on the target street. The driving domain lends credibility to the architectural distinction between concepts and skills, which the Icarus interpreter treats in quite different manners, as we will see shortly.

Hierarchical Structure of Memory

Another distinguishing feature of Icarus lies in its commitment to the hierarchical nature of long-term memory. There remains little doubt that human memory has this character. Many natural categories have a componential structure, and timing studies suggest that chunk boundaries remain even in well-practiced procedures. Most cognitive architectures let one model such hierarchical relations, but few raise this notion to a design principle. ACT-R comes the closest by letting productions link goals to subgoals, but the relation remains mediated by working memory elements rather than referring directly to component structures.

Icarus provides direct support for hierarchy at the architectural level. Recall that the fields in a concept definition can refer to other concepts, and thus organize categories into a conceptual lattice, with primitive concepts at the bottom and increasingly complex concepts at higher levels. Each Icarus skill includes fields that specify how to decompose it into subskills, with an `:ordered` field indicating an ordering on component skills and an `:unordered` field allowing choice among them. This hierarchy bottoms out in primitive skills, which specify executable actions in their `:actions` field. Moreover, skills refer to concepts in other fields, thus linking the two memories in a hierarchical manner.

Tables 1 and 2 provide examples of these relations. The concept *in-rightmost-lane* is defined using *lane-to-right* and *line-to-left*, whereas the skill *continue-to-address* refers to *speed-up-for-cruise* and *turn-around-for* in its `:unordered` field, as well as to various concepts in its `:requires`, and `:objective` fields. Note also that higher-level structures refer directly to their components by name, giving more direct indexing than in production system architectures.

Long-Term/Short-Term Correspondence

A cognitive architecture requires more than long-term memory; it must also have a short-term memory that contains dynamic beliefs and intentions. A recurring idea in cognitive science is that this short-term store should simply be the 'active' portion of long-term memory. This relation holds for the declarative memories in ACT-R, but not for its procedural production rules, which are purely long term, and Soar does not support such a mapping in any form. Theories of case-based reasoning come much closer to this theme, but these have seldom been cast as general architectures.

Icarus enforces a strong correspondence by requiring that every short-term element be a specific instance of some long-term structure. In particular, its conceptual short-term memory contains instances of defined concepts which encode specific beliefs about the environment that the agent can infer from its perceptions.[3] For instance, this memory might contain the instance *(lane-to-right self g0037)*, which it can infer from the *lane-to-right* concept shown in Table 1.

The architecture also incorporates a skill short-term memory, which contains instances of skills the agent intends to execute. Each of these literals specifies the skill's name and its concrete arguments, which must be known objects. For example, this memory might contain the skill instance *(continue-to-address self g0019)*, which denotes that the driver has an explicit intention to execute the *continue-to-address* skill with these arguments. Every short-term element must be either a concept instance (belief) or a skill instance (intention), which places psychologically plausible constraints on the structures that an Icarus agent can process.

Value-Driven Nature of Behavior

Early research on cognitive architectures emphasized symbolic decision making over numeric evaluation, and this still holds for some current frameworks like Soar and EPIC. The ACT-R theory diverges from this trend by associating an expected cost and benefit with each production rule, but these have the same value regardless of the current situation. In domains like driving, it seems clear that humans assign different values to certain skills, such as changing lanes and slowing, as

[3]This correspondence does not hold for Icarus' perceptual buffer, which stores the agent's momentary perceptions.

a function of the perceived situation, such as distance and speed relative to another car. Such situation-specific evaluation functions have a long history in game-playing systems, where they are combined with legal move generators to guide search.

Icarus incorporates a similar notion into its architectural design. Each skill decomposition has an associated value function that encodes the utility expected if the skill is executed with this decomposition. This function is defined by a `:percepts` field that matches the values of observed objects' attributes, a `:features` field that lists these values in order, a `:weights` field that states the weights on these quantities, and a `:constant` field that specifies a constant value. The expected utility for a skill decomposition is a linear function of the numeric descriptors matched by that skill. For example, the value for *speed-up-for-cruise* in Table 2 depends on the variable *?spd*, which can vary across cycles.

On each cycle, Icarus applies all matched concepts in a bottom-up manner to draw high-level inferences from its immediate perceptions, whereas skill execution occurs in a more selective, top-down way that is controlled by these value functions. The skill hierarchy defines an AND-OR tree down which the interpreter traverses on each cycle, starting from top-level intentions in short-term skill memory. Each path through this hierarchy terminates in an action, but Icarus must select one to execute on the current cycle. Many paths are rejected because their `:start` or `:requires` fields are not satisfied, or because their `:objective` has been met. Also, the architecture prefers subskills later in an `:ordered` field, as they are closer to the skill's objective.

However, many paths may remain available as alternatives, just as a human driver has choices about when to pass, slow down, and turn a corner. Icarus calculates the value of each acceptable path through the skill hierarchy, computing the value of each skill instance along a path and taking their sum as the total path value. This scheme produces context effects, since the value of taking a low-level action like speeding up or turning the wheel is influenced by the higher-level skills in which it participates. Icarus also multiplies the value of the path from the previous cycle by a *persistence* factor. When set to one, this produces purely reactive behavior with no memory of previous intentions, but higher settings bias the system toward selecting the skill instances it has been pursuing, giving it a form of commitment that purely reactive approaches lack.

Initial Experiences with Icarus

We believe that our design for Icarus is internally consistent and makes contact with a variety of psychological phenomena, but whether it supports embodied intelligent behavior is an empirical question. To obtain an initial answer to this question, we developed models of behavior in four domains, which we report here. Our aim was not to fit the details of human behavior, which is varied enough to require many distinct models, but to demonstrate the architecture's general ability to produce plausible behavior across a range of tasks.

In-City Driving

We described package delivery task earlier, but we should we report our experience with the Icarus model we have developed for it. The system includes 31 primitive concepts and 49 higher-level concepts, which range from two to six levels deep. These are grounded in perceptual descriptions for building corners, lane lines, street signs, packages, other vehicles, and the agent's vehicle. The model also incorporates 24 primitive skills and 34 higher-level skills, organized in a hierarchy that is eight levels deep. These terminate in executable actions for changing speed, altering the wheel angle, and depositing a package.

We have run the system on three variants of the package delivery task, most involving a city with nine square blocks, two other vehicles, and four packages with different target addresses across runs. The program reliably drives on the right-hand side, slows for intersections, and makes successful turns. On occasion, it makes an overly wide turn, but it quickly returns to the proper side. The model slows to avoid collision when it comes behind another vehicle that is driving more slowly, but otherwise keeps going.

When the agent first comes upon the target street or cross street marked on one of the packages it is carrying, it turns right on that street. The system continues on the cross street until reaching the end, in which case it makes a U turn, or until it comes upon the target street, in which case it turns right. Once on the target street, the agent continues if the numbers are changing in the right direction or makes a U turn otherwise. Upon reaching the package's target address, it deposits the package and continues driving. If the system encounters the target or cross street for a second package while delivering the first, it may shift to the new task instead. Whether this occurs depends on the persistence parameter, which produces single-minded behavior at one extreme and indecisive dithering at the other.

The Tower of Hanoi

One of the most heavily studied tasks in cognitive modeling is the Tower of Hanoi. This puzzle, which involves N disks of different sizes and three pegs, typically starts with all disks on one peg and requires moving them to a different target peg. Only one disk can be moved at a time, only the smallest disk on a peg can be moved, and a disk cannot be moved to a peg on which a smaller one already sits. The state space for this puzzle is small, yet it presents considerable difficulty to novices.

Although models of problem solving on the Tower of Hanoi emphasize cognitive processing, the puzzle clearly involves perception and action, and so constitutes a reasonable domain for Icarus. We developed a simulated environment with objects for pegs and disks, each of which has numeric attributes for its x position, y position, height, and width, along with a hand that is empty or full. Actions include grasping and ungrasping a disk, along with moving a grasped disk either vertically or horizontally. Our ICARUS program for this task includes seven concepts for describing states (e.g., recognizing when a disk in on a peg), three primitive skills, and one high-level skill for moving a disk to a target peg that, in some expansions, refers to itself recursively.

The resulting system models behavior at a finer granularity than most studies of this task, in that it treats more seriously the role of perception and action. The program solves the three-disk puzzle in seven disk moves and the four-disk version in 15 moves, but each such move requires three cycles – for grasping and lifting the disk, for shifting it horizontally, and for lowering and ungrasping it. Moreover, the categorization process makes inferences on each cycle that produce the higher-level description used in testing the skills. The system utilizes goal recursion to select the right peg for each disk, but it does this in the context of executing its skills, rather than generating a mental plan.

Pole Balancing

Most problem-solving tasks studied in cognitive science involve goals of achievement, yet many control tasks instead involve maintenance goals. To demonstrate that ICARUS handles problems of this variety, we developed a model of behavior for pole balancing, in which the agent tries to balance an initially upright pole on its end by pushing that end to the left or right. The only object in this environment is the pole, for which the agent can perceive the angle and angular velocity. The agent also has access to actions for pushing the pole to the left and to the right.

We have developed three ICARUS programs for this task. These include a purely logical version in which requirements on six primitive skills are cast as 11 mutually exclusive qualitative states and an alternative in which the value functions for three primitive skills are linear functions of the pole's angle and angular velocity. Runs revealed that the value-based version could balance the pole indefinitely, whereas the logical version could do so for less than two hundred cycles. However, we also developed a hierarchical variant of the value-based system that incorporates two skills with knowledge about the order of primitive skills. This system behaved as robustly as the flat version and, we maintain, encodes more faithfully human skill on this task, which has high-level regularities in action ordering.

Multi-Column Subtraction

A routine but complex task that has received attention by cognitive modelers is multi-column subtraction. To reproduce behavior in this domain, we developed an environment in which perceivable objects correspond to digits, which have an x position, y position, numeric value, and status (clear or crossed out). Primitive actions include writing down a new digit, crossing out a digit, and replacing one value with another.

Our ICARUS program includes concepts for grouping digits into columns, recognizing row adjacencies, and noting when columns have been processed. There are primitive skills for taking a difference, adding ten, and decrementing by one, along with one hierarchical skill. This has one expansion for simple borrowing and another for borrowing across zero, which invokes itself recursively. The system has a similar flavor to VanLehn's (1990) treatment, but we have modeled only correct behavior and not the errors observed in this domain.

Discussion

We argued earlier that ICARUS incorporates a number of features that distinguish it from typical cognitive architectures. However, this does not mean related ideas have not appeared elsewhere in the AI literature under different guises. For instance, our approach has much in common with the 'reactive planning' movement, which often utilizes hierarchical procedures that combine cognition, perception, and action in physical domains. Examples include Georgeoff et al.'s (1985) PRS, Nilsson's (1994) teleoreactive framework, and Bonasso et al.'s (1997) 3T robotic architecture. Howe (1995) reports a system that combines, executes, and revises partial plans in complex environments, whereas Hammond (1993) describes a predecessor that even delivers packages in a simulated driving environment. However, within this paradigm, only Freed's (1998) APEX has been proposed as a candidate architecture for human cognition, and it differs from ICARUS on other fronts.

Our framework also shares ideas with control theory and reinforcement learning, which often invoke linear functions of sensory variables to determine the behavior of reactive agents. However, most work in these paradigms assumes only low-level controllers, with neither higher-level conceptual descriptions or skills. Research on hierarchical reinforcement learning comes closer, but does not make strong assumptions about the architecture for cognition. Our own work on this topic (Shapiro et al., 2001) made commitments to hierarchical skills and value-driven behavior, but did not support conceptual memories, multiple intentions, or the mechanisms to utilize them. One important exception is Albus and Meystel's (2001) RCS architecture, which organizes knowledge hierarchically and also makes a clear distinction between logical structures and value judgments. ICARUS and RCS are perhaps more akin to each other than to other frameworks, but they retain many differences due to their origins in cognitive modeling and control theory, respectively.

Despite its novel characteristics, as a cognitive architecture the current version of ICARUS falls short on a number of fronts. One drawback is the assumption of unlimited perceptual resources, which lets an agent sense each attribute of every object within a certain distance. Clearly, humans have more limited abilities, in that their visual field is relatively narrow and they must focus attention on an object to extract its features. We plan to treat perceptual attention as another action under the skills' control, which will transform it into a constrained resource. However, this change will interact with the current assumption that conceptual inferences are removed from short-term memory when their supporting perceptions disappear. One response would be to retain inferred beliefs (e.g., that a lane is clear) across cycles, but to associate with them expected durations, which in turn would influence attentional decisions.

Another omission relates to Icarus' reliance on prestored skills. Although humans typically prefer to use routine behaviors when possible, there is clear evidence that they can, within limits, combine knowledge elements when needed to solve novel problems. Means-ends analysis has been implicated in such situations, so we plan to incorporate a version of this method into future versions of the architecture. However, it should remain subservient to the primary process of skill execution, rather than being the dominant control scheme, as in Minton's (1990) Prodigy. This also suggests an account for the acquisition of hierarchical skills, which we hypothesize are the cached results of means-ends problem solving. The mechanism for composing existing skills that appear in a novel solution would be similar to chunking in Soar and knowledge compilation in ACT, but would retain the original structures as components of a new hierarchical skill, rather than creating an unstructured chunk or production rule.

Finally, Icarus lacks the key human ability to store its previous experiences in an episodic memory and retrieve them later. However, we have noted that our framework requires elements in short-term memory to be instances of generic concepts or skills. This suggests we can model episodic memory by extending these structures to include time markers that indicate when they entered and left the short-term stores, with elements being indexed through the generic structures of which they are instances. The mechanism responsible for episodic recall is less clear, but Forbus et al.'s (1994) analogical retrieval method is one plausible candidate.

In closing, we should review the distinctive characteristics of our framework and our initial results. Unlike most cognitive architectures, Icarus is concerned centrally with modeling intelligent behavior in physical domains. As a result, processes for perception and action are more basic than ones for inference and decision making, though they interact tightly. The architecture makes a clear separation between concepts and skills, which are stored in distinct but connected long-term memories, and this dichotomy carries over to short-term memories, which contain instances of concepts (beliefs) and skills (intentions). Both conceptual and skill memory are hierarchical, with the latter structuring action selection, which involves computing the expected value of acceptable paths from high-level to primitive skills.

Our experimental studies of Icarus' behavior are in their early stages, and our results to date are qualitative. Nevertheless, we have shown that the architecture supports an interesting mixture of cognition, perception, and action in a complex in-city driving domain, and our experiences with three other domains provide encouraging evidence of generality. We identified some limitations of the current architecture, but these suggested in turn some natural extensions that will give broader coverage of human behavior and that, we believe, are difficult to achieve in traditional approaches.

References

Albus, J. S., & Meystel, A. M. (2001). *Engineering of mind: An introduction to the science of intelligent systems*. New York: John Wiley.

Anderson, J. R. (1993). *Rules of the mind*. Hillsdale, NJ: Lawrence Erlbaum.

Bonasso, R. P., Firby, R. J., Gat, E., Kortenkamp, D., Miller, D., & Slack, M. (1997). Experiences with an architecture for intelligent, reactive agents. *Journal of Experimental and Theoretical Artificial Intelligence*, *9*, 237–256.

Freed, M. (1998). Managing multiple tasks in complex, dynamic environments. *Proceedings of the National Conference on Artificial Intelligence* (pp. 921–927).

Forbus, K., Gentner, D., & Law, K. (1994). MAC/FAC: A model of similarity-based retrieval. *Cognitive Science*, *19*, 141–205.

Georgeff, M., Lansky, A., & Bessiere, P. (1985). A procedural logic. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*. Los Angeles: Morgan Kaufmann.

Hammond, K. (1993). Toward a theory of agency. In S. Minton (Ed.) *Machine learning methods for planning*. San Francisco: Morgan Kaufmann.

Howe, A. E. (1995). Improving the reliability of AI planning systems by analyzing their failure recovery. *IEEE Transactions on Knowledge and Data Engineering*, *7*, 14–25.

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

Laird, J. E., Newell, A., & Rosenbloom, P. S. (1987). Soar: An architecture for general intelligence. *Artificial Intelligence*, *33*, 1–64.

Minton, S. N. (1990). Quantitative results concerning the utility of explanation-based learning. *Artificial Intelligence*, *42*, 363–391.

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

Nilsson, N. (1994). Teleoreactive programs for agent control. *Journal of Artificial Intelligence Research*, *1*, 139–158.

Shapiro, D., Langley, P., & Shachter, R. (2001). Using background knowledge to speed reinforcement learning in physical agents. *Proceedings of the Fifth International Conference on Autonomous Agents* (pp. 254–261). Montreal: ACM Press.

VanLehn, K. (1990). *Mind bugs: The origins of procedural misconceptions*. Cambridge, MA: MIT Press.