A Unified Cognitive Architecture for Physical Agents

Pat Langley and Dongkyu Choi
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 physical agents that integrates ideas from a number of traditions, but that has been especially influenced by results from cognitive psychology. We review ICARUS' commitments to memories and representations, then present its basic processes for performance and learning. We illustrate the architecture's behavior on a task from in-city driving that requires interaction among its various components. In addition, we discuss ICARUS' consistency with qualitative findings about the nature of human cognition. In closing, we consider the framework's relation to other cognitive architectures that have been proposed in the literature.

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 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.

A conventional paper on cognitive architectures would first describe the memories and their contents, then discuss the mechanisms that operate over them. However, ICARUS' processes interact with certain memories but not others, suggesting that we organize the text around these processes and the memories on which they depend. Moreover, some modules build on other components, which suggests a natural progression. Therefore, we first discuss ICARUS' most basic mechanism, conceptual inference, along with the memories it inspects and alters. After this, we present the processes for goal selection and skill execution, which operate over the results of inference. Finally, we consider the architecture's module for problem solving, which builds on both inference and execution, and its associated learning processes, which operate over the results of problem solving.

In each case, we discuss the framework's connection to qualitative results from cognitive psychology. In addition, we illustrate the ideas with examples from the domain of in-city driving, which has played a central role in our research. Briefly, this involves controlling a vehicle in a simulated urban environment with buildings, road segments, street intersections, and other vehicles. This domain, which Langley and Choi (2006) describe at more length, provides a rich setting to study the interplay among different facets of cognition.

Beliefs, Concepts, and Inference

In order to carry out actions that achieve its goals, an agent must understand its current situation. ICARUS includes a module for conceptual inference that is responsible for this cognitive task which operates by matching conceptual structures against percepts and beliefs. This process depends on the contents and representation of elements in short-term and long-term memory.

Because ICARUS is designed to support intelligent agents that operate in some external environment, it requires information about the state of its surroundings. To this end, it incorporates a *perceptual buffer* that describes aspects of the environment the agent perceives directly on a given cycle, after which it is updated. Each element or percept in this ephemeral memory corresponds to a particular object and specifies the object's

type, a unique name, and a set of attribute-value pairs that characterize the object on the current time step.

Although one could create a stimulus-response agent that operates directly off perceptual information, its behavior would not reflect what we normally mean by the term 'intelligent', which requires higher-level cognition. Thus, ICARUS also includes a *belief memory* that contains higher-level inferences about the agent's situation. Whereas percepts describe attributes of specific objects, beliefs describe relations among objects, such as the relative positions of two buildings. Each element in this belief memory consists of a predicate and a set of symbolic arguments, each of which refers to some object, typically one that appears in the perceptual buffer.

ICARUS beliefs are instances of generalized concepts that reside in *conceptual memory*, which contains long-term structures that describe classes of environmental situations. The formalism that expresses these logical concepts is similar to that for PROLOG clauses. Like beliefs, ICARUS concepts are inherently symbolic and relational structures. Each clause in conceptual memory includes a head that gives the concept's name and arguments, along with a body that states the conditions under which the clause should match against the contents of short-term memories.

The architecture's most basic activity is conceptual inference. On each cycle, the environmental simulator returns a set of perceived objects, including their types, names, and descriptions in the format described earlier. ICARUS deposits this set of elements in the perceptual buffer, where they initiate matching against long-term conceptual definitions. The overall effect is that the system adds to its belief memory all elements that are implied deductively by these percepts and concept definitions. ICARUS repeats this process on every cycle, so it constantly updates its beliefs about the environment.

The inference module operates in a bottom-up, data-driven manner that starts from descriptions of perceived objects. The architecture matches these percepts against the bodies of primitive concept clauses and adds any supported beliefs (i.e., concept instances) to belief memory. These trigger matching against higher-level concept clauses, which in turn produces additional beliefs. The process continues until ICARUS has added to memory all beliefs it can infer in this manner. Although this mechanism reasons over structures similar to PROLOG clauses, its operation is closer to the elaboration process in the Soar architecture (Laird et al., 1987).

For example, for the in-city driving domain, we provided ICARUS with 41 conceptual clauses. On each cycle, the simulator deposits a variety of elements in the perceptual buffer, including percepts for the agent itself (*self*), street segments (e.g., *segment2*), lane lines (e.g., *line1*), buildings, and other entities. Based on attributes of the object *self* and one of the segments, the architecture derives the primitive concept instance (*in-segment self segment2*). Similarly, from *self* and the object *line1*, it infers the belief (*in-lane self line1*). These two elements lead ICARUS to deduce two non-primitive beliefs, (*centered-in-lane self segment2 line1*) and (*aligned-with-lane-in-segment self segment2 line1*). Finally, from these two instances and another belief, (*steering-wheel-straight self*), the system draws an even higher-level inference, (*driving-well-in-segment self segment2 line1*). Other beliefs that encode relations among perceived entities also follow from the inference process.

ICARUS' conceptual inference module incorporates a number of key ideas from the psychological literature:

- Concepts are distinct cognitive entities that humans use to describe their environment and goals; moreover, they support both categorization and inference;
- The great majority of human categories are grounded in perception, making reference to physical characteristics of objects they describe (Barsalou, 1999);
- Many human concepts are relational in nature, in that they describe connections or interactions among objects or events (Kotovsky & Gentner, 1996);
- Concepts are organized in a hierarchical manner, with complex categories being defined in terms of simpler structures.

ICARUS reflects each of these claims at the architectural level, which contrasts with most other architectures' treatment of concepts and categorization.

However, we will not claim our treatment is complete. ICARUS currently models concepts as Boolean structures that match in an all-or-none manner, whereas human categories have a graded character (Rosch & Mervis, 1975). Also, retrieval occurs in a purely bottom-up fashion, whereas human categorization and inference exhibits top-down priming effects. Both constitute important directions for extending the framework.

Goals, Skills, and Execution

We have seen that ICARUS can utilize its conceptual knowledge to infer and update beliefs about its surroundings, but an intelligent agent must also take action in the environment. To this end, the architecture includes additional memories that concern goals the agent wants to achieve, skills the agent can execute to reach them, and intentions about which skills to pursue. These are linked by a performance mechanism that executes stored skills, thus changing the environment and, hopefully, taking the agent closer to its goals.

In particular, ICARUS incorporates a *goal memory* that contains the agent's top-level objectives. A goal is some concept instance that the agent wants to satisfy. Thus, goal memory takes much the same form as belief memory, in that each element specifies a predicate defined in conceptual memory followed by its arguments. One important difference is that goals may have as arguments either specific objects, as in *(in-lane self line1)*, pattern-match variables, as in *(in-lane ?x ?y)*, or some mixture, as in *(in-lane self ?line)*. Also, goal memory may contain negated structures, whereas belief memory includes only positive elements. Because goals always refer to predicates defined in the conceptual memory, goal literals can indicate the agent's objectives at dif-

ferent levels of detail. Some goals may refer to primitive concepts stated in terms of percepts, while others may mention predicates far higher in the concept hierarchy.

Because goal memory may contain multiple elements, ICARUS must address the issue of cognitive attention. The architecture makes the common assumption that an agent can focus on only one goal at a time, which means that it must select among those in memory. ICARUS treats the contents of goal memory as an ordered list. On each cycle, it finds the goal with the highest priority that is unsatisfied and makes it the focus of cognitive attention, even if this means dropping a goal that was being pursued on the previous cycle. If all goals are satisfied, then the system has no focus on that cycle, although this may change later, leading the agent to refocus on goals that it achieved previously.

ICARUS stores knowledge about how to accomplish its goals in a long-term *skill memory* that contains skills it can execute in the environment. These take a form similar to conceptual clauses but have a somewhat different meaning because they operate over time and under the agent's intentional control. Each skill clause includes a head that states the skill's objective, along with a body which specifies the perceptual entities that must be present, the concepts that must match to initiate the clause, and the concepts that must match to continue its execution. Primitive skill clauses refer to actions that the agent can execute directly in the environment. These play the same role as operators in AI planning systems but can be durative in nature. In contrast, nonprimitive clauses refer to subgoals, cast as conceptual literals, that the agent should achieve in order.

One of ICARUS' important theoretical commitments is that the head of each skill clause denotes a concept the clause will achieve if executed to completion. This strong connection between skills and concepts figures centrally in the architecture's performance and learning mechanisms. A skill clause may have either a positive head, which specifies a concept that will become satisfied once the skill has been executed to completion, or a negated head, which indicates a concept that will become unsatisfied upon the skill's completion. This distinction makes sense for skill clauses because they are about carrying out action to achieve goals, which may themselves be positive or negative. Multiple skill clauses may have the same heads; these specify different ways to achieve the same goal under distinct conditions.

Once ICARUS has chosen an unsatisfied goal to achieve, the execution module selects skill clauses that it believes will help toward this end. Because the architecture can execute directly only primitive skills, it must find a path downward from this goal to some terminal node in the skill hierarchy. A skill path is a list of instantiated skill clauses, in which each clause head is a subgoal of the one that precedes it in the path. The execution module only considers paths in which each skill clause is *applicable* given its current beliefs. A clause is applicable if, for its current variable bindings, its head is not satisfied, its requirements are satisfied, and, if the system did not execute it on the previous cycle, the start conditions match the contents of belief memory.

Determining whether a given skill clause is applicable relies on the same match process utilized in conceptual inference and goal satisfaction. Matching the percepts, start conditions, and requirements fields of a skill involves comparing the generalized structures to elements in the perceptual buffer and belief memory, as does matching the clause head. The key difference is that variables in the head are typically already bound because a clause invokes its subgoals with arguments that are mentioned elsewhere in its body. This reflects the top-down nature of ICARUS' skill selection process.

Once the architecture has selected a path for execution, it invokes the instantiated actions included in the final (primitive) skill clause. These alter the environment, which in turn produces a different perceptual buffer on the next cycle and a different set of inferred beliefs. Given a choice among applicable paths, ICARUS selects the one that shares the most elements from the start of the path selected on the previous cycle. This bias encourages the agent to keep executing a high-level skill that it has started until it achieves the associated goal or becomes inapplicable. Otherwise, given a choice between ordered subgoals, it selects the first one for which the head is unsatisfied. This bias supports reactive control, since the agent reconsiders achieved subgoals and, if unexpected events have made them untrue, reinvokes them to correct the situation.

In the driving domain, we provided ICARUS with 19 primitive skill clauses and 20 nonprimitive clauses for basic car-handling abilities. When given the single goal (*in-segment self segment2*), the system selects this structure as the focus of attention and attempts to find an applicable skill path that will achieve it. The execution module retrieves a skill clause with the same predicate as the goal and start conditions that, after substituting variable bindings, are satisfied. The first instantiated subgoal of this clause, (*in-intersection-for-right-turn self int3 line1*), matches against the head of another clause that has two subgoals, (*in-rightmost-lane self line1*) and (*in-intersection-for-right-turn self int3 line1*). ICARUS selects the first subgoal, since it has not yet been achieved, and considers its first subgoal, (*driving-in-segment self segment1 line1*), where *segment1* is to the left of *segment2*. This clause has (*in-lane self line1*) as an instantiated subgoal, which terminates the path by matching the head of a primitive skill clause with the action ∗*steer-right*. No others are applicable, so ICARUS selects this one for execution,

The module for skill execution builds directly on ICARUS' mechanism for goal processing, in that the former uses the latter to determine when goals and subgoals have been achieved and which goal stack to focus upon. Both components in turn depend on conceptual inference to produce the beliefs they compare against goals and skill conditions. The modules are highly interdependent, making ICARUS an example of what Newell (1990) has called a *unified* cognitive architecture.

ICARUS' execution module is consistent with additional psychological claims, most of them so obvious that they seldom appear in the literature:

- Humans can deal with multiple goals, some having priority over others, which can lead to interrupted tasks to which attention later returns;
- Skill execution supports complex activities that people report, and our educational system assumes, have hierarchical organization (Rosenbaum et al., 2001);
- Humans can carry out open-loop sequences (e.g., playing a piano piece) but they can also operate in closed-loop reactive mode (e.g., while driving);
- The same generic skill may be applied to distinct objects, provided these objects satisfy conditions for the skill's application.

In summary, the architecture takes into account a number of widespread beliefs about human cognitive skills. However, it does not explain other important behaviors that we should address in future work, such as limited abilities for parallel execution, the generation and abandonment of top-level goals, and the common forgetting of interrupted tasks.

Problem Solving and Skill Learning

ICARUS' ability to execute hierarchical skills in an environment is sufficient when the agent has stored strategies for the goals and situations it encounters, but humans can also respond adaptively to cases in which they lack such procedures. Thus, the architecture also includes a module for achieving its goals through problem solving, which involves dynamically composing known skills into plans and executing these structures.

This capability requires an extension to the goal memory that replaces individual structures with *goal stacks*. Each stack contains a list of elements, the last of which encodes information related to the agent's top-level goal. In familiar situations, the stack never contains more than this element, since the execution module can rely entirely on its long-term skills. However, in less routine settings the goal stack varies in depth as the problem solver adds and removes subgoals. On any cycle, the first element in the list contains information about the current goal, which drives behavior.

ICARUS invokes its problem solver whenever it encounters an impasse, which occurs on any cycle in which it cannot retrieve an applicable skill path that would achieve the current goal. On each cycle, the system checks whether the current goal G on the goal stack has been achieved. If so, then the module pops the stack and addresses G's parent goal or, upon achieving the top-level goal, considers the next one in priority. If the current goal G is unsatisfied, it retrieves skills with heads that match G, selects a candidate at random, and stores it with G as an intention. If the problem solver finds no skill clauses that would achieve the goal G, it uses G's concept definition to decompose it into subgoals. If more than one subgoal is unsatisfied, ICARUS selects one at random and pushes it onto the stack.

The problem-solving strategy we have just described is a version of means-ends analysis (Newell & Simon, 1961; Carbonell et al., 1990). However, its behavior differs somewhat from the standard formulation in that it is tightly integrated with the execution process. ICARUS chains backward off concept or skill definitions when necessary, but it executes the skill associated with the current stack entry when it becomes applicable. Moreover, because the architecture can chain over hierarchical reactive skills, their execution may continue for many cycles before problem solving resumes.

For example, when we give ICARUS only 19 primitive skills for driving and the goal (*in-segment self segment2*), it cannot find any applicable skill path that would achieve the goal. Upon encountering this impasse, the system resorts to problem solving, which retrieves skills that would achieve it if only they were applicable. The module finds a skill clause with the predicate *in-segment* in its head that would serve, but its instantiated start condition, (*in-intersection-for-right-turn self int3 line1*) is unsatisfied, so it pushes this onto the goal stack. The problem solver retrieves another skill clause with a head that matches this subgoal, but its start condition, (*in-rightmost-lane self line1*), also does not hold, so it becomes the next subgoal.

Because ICARUS has no skills that would achieve this literal, it chains off the concept *in-rightmost-lane*'s definition. Only one of the instantiated concepts, (*driving-in-segment self segment1 line1*), is unsatisfied, so the means-ends module pushes it onto the goal stack, then repeats the process with the literal (*in-lane self line1*) through additional concept chaining. In this case, the system finds an applicable skill clause that will achieve the subgoal directly, which it executes in the environment. After several cycles, ICARUS achieves the subgoal, pops the stack, and returns attention to its parent, (*driving-in-segment self segment1 line1*). After satisfying other subgoals in a similar manner, the module pops this goal and considers its parent. This process continues until the agent achieves the top-level goal.

Although ICARUS' problem solver lets it overcome impasses and achieve goals for which it has no stored skills, the process can require considerable search and backtracking. For this reason, the architecture includes a final module that learns from solved problems so that, when it encounters similar situations, no impasse will occur. When the agent achieves a goal G by chaining off a skill clause with head B, which in turn required achieving its start condition A, ICARUS constructs a new skill clause with the head G, the ordered subgoals A and B, and the same start condition as that for the first subskill. When the agent achieves a goal G through concept chaining, the system creates a skill clause with head G, with subgoals based on G's subconcepts in the order it achieved them, and with start conditions based on those subconcepts satisfied at the outset.

Because means-ends analysis decomposes problems into subproblems, these mechanisms lead naturally to the formation of skill hierarchies (Langley & Choi,

2006). Also, because ICARUS utilizes the goal achieved during each impasse as the head of the learned clause, it can produce disjunctive skills and, whenever a goal concept appears as one of its own subgoals, to recursive structures. Although the architecture shares the idea of learning from impasses with Soar and PRODIGY, the process is simpler, in that it draws on local goal-stack information that is needed for problem solving anyway.

From the successful problem-solving episode in the driving domain discussed earlier, ICARUS extracts five new skill clauses. For example, based on skill chaining from the highest-level goal, (*in-segment self segment2*), it learns a clause with two subgoals, *intersection-for-right-turn* and *in-segment*, the second involving a recursive call with the same arguments. The start conditions for this new clause are taken from the start conditions for the skill clause the agent used to achieve the first subgoal. Based on chaining from the concept definition for (*driving-in-segment self segment1 line1*), ICARUS constructs a new skill clause with four subgoals, the first of which refers to the predicate *in-lane*. The start condition includes the concepts *in-segment* and *steering-wheel-straight*, since these held when the system began on this subproblem. The acquired structures let the agent change lanes, align itself, and turn a corner to reach a desired segment by invoking the execution module but without calling the problem solver.

As we have seen, ICARUS interleaves problem solving with execution, which means the former builds upon the latter. The means-ends module also draws on mechanisms for goal processing to determine when subgoals have been achieved. The architecture's learning methods in turn rely on problem solving, both to signal opportunities for skill acquisition and to provide material for constructing new clauses. These interdependencies constitute further evidence that ICARUS moves beyond integration to provide a unified account of cognition.

Moreover, ICARUS' approach to problem solving and learning incorporates other key ideas from psychology:

- Humans often resort to means-ends analysis to solve unfamiliar problems (Newell & Simon, 1961);
- Problem solving often occurs in a physical context, with humans interleaving mental problem solving with execution (Gunzelmann & Anderson, 2003);
- Efforts to overcome impasses during problem solving can lead to the incremental acquisition of new skills (Anzai & Simon, 1979);
- Learning can transform backward-chaining heuristic search into more informed forward-chaining behavior (Larkin et al., 1980).

These ideas have been modeled in other frameworks but ICARUS incorporates them into the architecture, making stronger theoretical claims than its predecessors.

However, the current version is inconsistent with other important phenomena. The problem solver carries out depth-first search, whereas human short-term memory cannot suport such systematic methods, and it also utilizes a strict goal stack, whereas recent evidence suggests more flexible structures. Moreover, ICARUS supports only backward-chaining search, whereas in complex domains like chess, humans also exhibit progressive deepening (de Groot, 1978), which involves limited forward chaining. Finally, the system learns only from success, when it achieves goals, whereas people can also learn from failure. Future versions of the architecture should address each of these shortcomings.

Comparison to Other Architectures

ICARUS has much in common with previous cognitive architectures like Soar (Laird et al., 1987) and ACT-R (Anderson, 1993). Like its predecessors, the framework makes commitments about memories, representations, and cognitive processes that support intelligent behavior. Some shared assumptions include claims that:

- dynamic short-term memories are distinct from long-term memories, which store more stable content;
- both forms of memory contain modular elements that are composed during performance and learning;
- memory elements are cast as symbolic list structures, with those in long-term memory being accessed by matching their patterns against short-term elements;
- cognitive behavior occurs in cycles that retrieve and instantiate long-term structures, then use selected elements to carry out mental or physical actions; and
- learning is incremental and interleaved with performance, with structural learning involving monotonic addition of symbolic elements to long-term memory.

These ideas have their origins in theories of human memory, problem solving, and skill acquisition. They are widespread in research on cognitive architectures but relatively rare in other parts of artificial intelligence.

Despite these similarities, ICARUS also incorporates some theoretical claims that distinguish it from most other architectures. These include assumptions that:

- cognition occurs in a physical context, with mental structures being grounded in perception and action;
- concepts and skills encode different aspects of knowledge that are stored as distinct cognitive structures;
- each element in a short-term memory has a corresponding generalized structure in long-term memory;
- long-term memories have hierarchical organizations that define complex structures based on simpler ones;
- conceptual inference and skill execution are more basic than problem solving, which uses these processes;

These ideas separate ICARUS from most architectures that have been developed within the Newell tradition. We will not argue that they make it superior to earlier frameworks, but we believe they make it an interesting alternative within the space of candidate architectures.

Many of these claims involve matters of emphasis rather than irreconcilable differences. Soar and ACT-R have been extended to interface with external environments, but both focused initially on central cognition, whereas ICARUS began as an architecture for reactive execution and highlights physical interaction. ACT-R

states that elements in short-term memory are active versions of structures in declarative memory, but does not make ICARUS' stronger claim that the former must be instances of general concepts. Soar incorporates an elaboration stage similar to our conceptual inference, but it lacks an explicit conceptual hierarchy. ACT-R programs often include rules that match against goals and set subgoals, whereas ICARUS elevates this idea to an architectural principle. These similarities reflect an underlying concern with similar issues, but they also reveal distinct philosophies about how to approach them.

Our framework also shares some assumptions with BDI architectures, which give central roles to beliefs, desires, and intentions. Research in this paradigm has also focused on hierarchical procedures for physical action, but has typically relied on handcrafted structures and made limited contact with ideas from psychology. Bonasso et al.'s (2003) 3T architecture combines planning with reactive control and Freed's (1998) APEX architecture aims to model human behavior, but neither accounts for skill acquisition. ICARUS shares other features with Carbonell et al.'s (1990) PRODIGY, which uses means-ends analysis to solve problems and learns control knowledge to reduce search. Finally, our framework has links to CLARION (Sun et al., 2001), which also distinguishes between knowledge used for action and inference, and which learns skills from experience.

Concluding Remarks

In summary, ICARUS is a cognitive architecture for physical agents that combines some familiar mechanisms in novel ways. The framework includes modules for conceptual inference, goal selection, skill execution, means-ends problem solving, and skill learning. Higher levels of processing rely heavily on the results of lower levels, making ICARUS an example of what Newell (1990) refers to as a unified theory of cognition. The framework is also consistent with many well-established views about the nature of human cognitive processing.

We illustrated the architecture's mechanisms with examples related to in-city driving, a domain that combines the need for extended activity with reactive control. Elsewhere we have reported ICARUS models for more traditional cognitive tasks, including multi-column subtraction, the Tower of Hanoi, and various planning domains, which together provide evidence of the framework's generality. We have also reported experiments that demonstrate its learning method's ability to transform problem solving into routine execution.

However, we have already noted some important dimensions on which we should extend the framework to better match knowledge of human behavior. Other important omissions include the ability to store and access episodic memory, to acquire and refine concepts, and to use hierarchical skills for interpreting other agents' behaviors. We believe the architecture provides natural ways to address these issues, and their inclusion should let ICARUS offer a more complete account of cognition.

Acknowledgements

This paper reports research sponsored by DARPA under agreement FA8750-05-2-0283. The U. S. Government may reproduce and distribute reprints for Governmental purposes notwithstanding any copyrights. The authors' views and conclusions should not be interpreted as representing official policies or endorsements, expressed or implied, of DARPA or the Government.

References

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

Anzai, Y., & Simon, H. A. (1979). The theory of learning by doing. *Psychological Review*, *86*, 124–140.

Barsalou, L. W. (1999). Perceptual symbol systems. *Behavioral and Brain Sciences*, *22*, 577–609.

Bonasso, R. P., Kortenkamp, D., & Thronesbery, C. (2003). Intelligent control of a water recovery system: Three years in the trenches. *AI Magazine*, *24*, 19–44.

Carbonell, J. G., Knoblock, C. A., & Minton, S. (1990). PRODIGY: An integrated architecture for planning and learning. In K. Van Lehn (Ed.), *Architectures for intelligence*. Hillsdale, NJ: Lawrence Erlbaum.

de Groot, A. D. (1978). *Thought and choice in chess* (2nd Ed.). The Hague: Mouton Publishers.

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

Gunzelmann, G., & Anderson, J. R. (2003). Problem solving: Increased planning with practice. *Cognitive Systems Research*, *4*, 57–76.

Kotovsky, L., & Gentner, D. (1996). Comparison and categorization in the development of relational similarity. *Child Development*, *67*, 2797–2822.

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

Langley, P., & Choi, D. (2006). Learning recursive control programs from problem solving. *Journal of Machine Learning Research*, *7*, 493–518.

Larkin, J. H., McDermott, J., Simon, D. P., & Simon, H. A. (1980). Expert and novice performance in solving physics problems. *Science*, *208*, 1335–1342.

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

Newell, A., & Simon, H. A. (1961). GPS, A program that simulates human thought. In H. Billing (Ed.), *Lernende automaten*. Munich: Oldenbourg KG.

Rosch, E., & Mervis, C. B. (1975). Family resemblance: Studies in the internal structure of categories. *Cognitive Psychology*, *7*, 573–605.

Rosenbaum, D. A., Carlson, R. A., & Gilmore, R. O. (2001). Acquisition of intellectual and perceptual-motor skills. *Annual Review of Psychology*, *52*, 453–470.

Sun, R., Merrill, E., & Peterson, T. (2001). From implicit skills to explicit knowledge: A bottom-up model of skill learning. *Cognitive Science*, *25*, 203–244.