# Reactive Goal Management in a Cognitive Architecture

**Dongkyu Choi**
Department of Aeronautics and Astronautics
Stanford University, Stanford, CA 94305
dongkyuc@stanford.edu

## Abstract

Goals play an important role in human cognition. Different aspects of human mind influence the generation of goals they pursue, and the goals guide their behaviors. In psychology, researchers made significant efforts to study goals and their origin, and cognitive architectures include various facilities to handle the goals of artificial agents. One such architecture, ICARUS, supports goal-driven behaviors while maintaining reactivity, and its top-level goals play an important role of guiding the behavior of the agents. However, the architecture covers neither the origin of its top-level goals nor the dynamic aspects of them, and this imposes various restrictions like limited autonomy on ICARUS. In this paper, we extend the architecture to provide the capability to nominate top-level goals using the notion of long-term, general goals, and manage the nominated goals by prioritizing them. For prioritization of goals, we introduce a novel capability to match concepts in a continuous manner. We describe the details of the extended system using examples from driving a car, and discuss related and future work in this direction.

## Introduction and Motivation

Goals play an important role in human cognition. People have ideas on what they want to do or what they should do, and these give rise to many different goals. The environment influences this process heavily, by affecting both the selection and the prioritization of goals. Selected and prioritized goals then guide people's behaviors by restricting the space of possible actions to take.

Traditionally, this fundamental role of goals caught a lot of attention among psychologists, and we can find numerous accounts in the literature (e.g. Simon, 1967; Sloman, 1987; Gray and Braver, 2002). As computational models of cognition, most cognitive architectures (Newell 1990), too, provide facilities for goals. At the very least, these architectures allow the specification of goals or subgoals that guide the artificial agent's behaviors in either explicit or implicit fashion. But some architectures provide more than others, including nomination and retraction of goals. For instance, CLARION (Sun 2007) has drive and goal mechanisms that correspond to a psychological account of goal nomination. In Soar (Laird, Rosenbloom, and Newell 1986), the top-level operators can act as reactive goals and there are rules that govern their nomination as goals.

Another cognitive architecture, ICARUS (Langley and Choi 2006), operates in a goal-oriented fashion, in which it uses multiple, reactive, top-level goals. But the original ICARUS lacks any mechanism to add, delete, or reorder its top-level goals, limiting its capabilities significantly. In this paper, we present an extension to the architecture with a new goal management mechanism that works reactively to the environment. We extended ICARUS' explicit architectural distinction between long-term knowledge and short-term structures to cover goals as well, by introducing the notion of long-term, general descriptions of goals. The system instantiates these long-term goals based on the current situation of the world to get short-term, specific goals to guide its own behavior. The extended architecture has the ability to nominate, retract, and prioritize its top-level goals, allowing more suitable behaviors in reaction to its surroundings.

In the subsequent sections, we briefly review the ICARUS architecture and describe the extension for reactive goal management in detail, using examples from everyday driving situations. We conclude after discussions on related and future work.

## Review of the ICARUS Architecture

ICARUS (Langley and Choi 2006) shares its basic features with other cognitive architectures like Soar (Laird, Rosenbloom, and Newell 1986) and ACT-R (Anderson 1993). It makes commitments to its representation of knowledge, memory structures, and mechanisms for inference, execution, and learning. The system provides a computational framework for intelligent agents, which stays constant across different domains. In this section, we review the basic capabilities of the architecture before we continue our discussion on the extension for reactive goal management. We start with ICARUS' representation of knowledge and memories that support this, and then cover the architecture's inference and execution processes.

## Representation and Memories

The ICARUS architecture distinguishes conceptual and procedural knowledge. Its *concepts* describe various aspects of the environment, whereas its *skills* define procedures that are known to achieve corresponding concepts when executed to completion. ICARUS also distinguishes long-term knowledge and short-term structures. Long-term knowledge includes general descriptions of the environment and procedures. Short-term structures are instantiations relevant to the current situation, derived from ICARUS' long-term knowledge.

The distinctions along these two directions result in four main memories in ICARUS. Its long-term conceptual memory stores general definitions of concepts that involve variables for object symbols and their attributes to describe various situations. A long-term skill memory houses variablized skills that define general procedures to achieve certain concepts, namely, the *goals*. There also are short-term memories that correspond to these long-term memories. A short-term conceptual memory stores instantiated concepts, which the system believes to be true in the current situation. A short-term skill memory holds instantiated skills, along with their corresponding goals. For these reasons, we often call the short-term memories as the *belief memory* and the *goal memory*, respectively.

Table 1 shows the syntax for concepts in ICARUS. Each concept includes a head, and a body that takes one of the four forms shown. The first two concepts are *primitive*, and they include only perceptual matching conditions that ground on object information from the environment. On the other hand, the other two concepts are *non-primitive*, since they refer to other concepts in the `:relations` field. This hierarchical organization of concepts allows multiple levels of abstraction, and facilitates the description of complex situations in the world. Meanwhile, Table 2 provides the syntax for skills in ICARUS. Each skill has its goal as the head and includes a body in either of the forms shown. In a similar fashion to its conceptual counterparts, there are primitive skills and non-primitive ones. The first skill shown is primitive, and it consists of perceptual matching conditions, preconditions, and direct references to immediate actions in the world. The second skill, however, is non-primitive and it provides a way to decompose the task into subgoals instead of referencing directly to actions. In the next section, we cover ICARUS' processes that work over these knowledge structures.

Table 1: Syntax for ICARUS' concepts.

```
(<head>
 :percepts   (<perceptual matching conditions>))

(<head>
 :percepts   (<perceptual matching conditions>)
 :tests      (<tests against variables>))

(<head>
 :percepts   (<perceptual matching conditions>)
 :relations  (<references to other concepts>))

(<head>
 :percepts   (<perceptual matching conditions>)
 :tests      (<tests against variables>)
 :relations  (<references to other concepts>))
```

Table 2: Syntax for ICARUS' skills.

```
(<head>
 :percepts  (<perceptual matching conditions>)
 :start     (<preconditions>)
 :actions   (<direct actions>))

(<head>
 :percepts  (<perceptual matching conditions>)
 :start     (<preconditions>)
 :subgoals  (<a subgoal decomposition>))
```

## Inference and Execution

The ICARUS architecture operates in distinct cycles. On each cycle, the system invokes a series of processes including the inference of current beliefs and the execution of skill paths relevant to the situation (see Figure 1). At the beginning of each cycle, ICARUS receives the sensory input from the environment in its perceptual buffer. Based on this information, the system infers its beliefs, namely, all the concept instances that are true in the current state. It starts with the lowest-level structures (i.e., primitive concepts), and moves up the hierarchy to non-primitive concepts. ICARUS goes through this process every cycle, and therefore, any naive approach to the inference process is susceptible to the combinatorial effect the system encounters in domains with many objects. There have been several mechanisms proposed to alleviate this problem including the one described in Asgharbeygi et al. (2005).

When the system finishes inferring all its beliefs, it attempts to execute skills based on them. ICARUS retrieves skills that are relevant to its top-level goals, and finds one or more executable paths through its skill hierarchy[1]. A skill path is executable when all the skill instances on the path are executable, from top to bottom. Although a path can include a single primitive skill that achieves an ICARUS agent's top-level goal, most of the time a skill path starts with a non-primitive skill for a top-level goal and continues down several levels until it reaches a primitive skill at the bottom.

The primitive skill provides actions the system should take in the environment. The ICARUS architecture deposits these actions in its motor buffer and executes them to make changes in its surroundings. In turn, the perceptual input on

[1]By default, the ICARUS architecture stops searching through its skill hierarchy upon finding the first skill path that is executable. However, there are cases in which the system should find some or all such paths, and the architecture supports the retrieval of multiple skill paths and the coordinated execution of them. For more details, see Choi (2009).

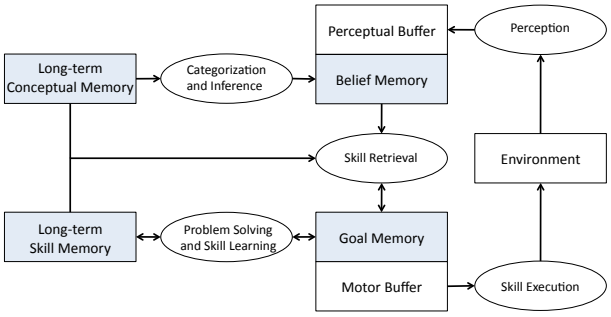


Figure 1: ICARUS' memories and the processes that work over them. The shaded rectangles represent memories, while the oval shapes stand for the processes that use them as inputs and produce outputs. Note that the figure also shows ICARUS' problem solving and skill learning process that we do not use in this paper.

the next cycle changes, and the system repeats all the above processes based on the new sensory data. In the following section, we continue our discussion on ICARUS in the context of the new extension.

## Reactive Goal Management

As seen earlier, the ICARUS architecture has a goal memory that stores information on its top-level goals and subgoals along with their corresponding skill instances. Most contents of the memory are very specific and short-lived, and they change as the agent moves along its path toward achieving its goals. But the top-level goals themselves did not change so far. It was as if an oracle gave the agent a fixed set of goals it should always pursue.

This, however, is not very reasonable. When people are pursuing some goals, more urgent matters come up sometimes and they should deal with them first. Or, they get distracted by something that happens in the environment. Therefore, in the extended architecture, the top-level goals for agents change dynamically, rather than staying constant throughout the course of execution. The system has a new goal nomination process that generates top-level goals for an agent on each cycle. The nominated goals are based on the generalized descriptions of goals stored in a *long-term goal memory*. In this new memory, we can program both general and domain-specific rules for the nomination (and implicitly, the retraction) of goals.

Once some goals are nominated for the cycle, ICARUS sorts them according to their relative priorities, calculated from the default priority value associated with the corresponding long-term goals. Since the default priority values are constants, however, the architecture requires a dynamic measure to modulate these based on the situation. Otherwise, the ordering of goals will stay fixed. For each goal, we use the *degree of relevance* as the measure, computed through continuous matching of the relevance conditions of the goal. In this section, we first introduce a new representation for goals and then describe the processes for nomination, retraction, and prioritization of goals. We also present the notion of continuous concept matching that we use for the prioritization.

### Representation

Probably, the best way to introduce the new representation of goals is by examples. Table 3 shows some sample goals stored in the long-term goal memory, relevant to some everyday driving situations. Each element represents a <*condition*, *goal*> pair that specifies a generalized goal and the conditions under which it is relevant. These resemble the *constraints* in Ohlsson and Rees (1991), but here they apply only in the context of goals. The relevance conditions stored in `:nominate` fields are concepts that the system can match against its beliefs. The goal concepts that appear as heads of the elements use some common variables from these relevance conditions. The relationship between these long-term goals and the elements in the existing short-term goal memory is similar to those between long-term concepts and beliefs, and long-term skills and instantiated ones. This is a feature that has some architectural significance, showing the unified nature of ICARUS.

Table 3: Some sample <*relevance conditions*, *generalized goal*> pairs stored in ICARUS' long-term goal memory.

```
((stopped-and-clear ME ?ped)
    :nominate ((pedestrian-ahead ME ?ped))
    :priority 10)

((clear ME ?car)
    :nominate ((vehicle-ahead ME ?car))
    :priority 5)

((cruising-in-lane ME ?line1 ?line2)
    :nominate nil
    :priority 1)
```

Long-term goals of the ICARUS architecture also have priority values associated with them, which represent the relative importance of the goals compared to others in the memory. Users provide a default prioritization measure by defining the fixed priority values. This corresponds to the general idea people seem to have on what is more important and what is less so. For instance, most people would agree that saving one's life has priority over saving his or her possessions. Many people will also save a child before saving an adult if caught in an accident. There are many examples like these, and we believe that the default priorities among ICARUS' generalized goals represent this behavior. Below, we continue our discussion on the processes that use the new representation.

### Nomination and Retraction Processes

When the ICARUS architecture finds a match for any relevance condition stored in its long-term goal memory, it instantiates the corresponding goal accordingly. The system then stores the instantiated goal in its short-term goal memory. When this nomination process is complete, the system

has a series of top-level goals, which guides its behavior during the cycle as in the original architecture. The nomination process starts after the architecture infers its beliefs based on the perceptual information from the environment. The system then goes through each <relevance conditions, generalized goal> pair stored in the long-term goal memory, and makes attempts to match the relevance conditions against the current beliefs. Whenever this attempt is successful, ICARUS instantiates the corresponding goal with the variable bindings it has found from the match. This also means that the retraction of goals happens without any additional mechanisms. If a currently nominated goal loses its match for relevance on the subsequent cycles, the system no longer nominates the goal and, in effect, retracts it from the short-term goal memory. During this retraction, however, ICARUS retains some information on the recent nomination, and uses it at a later time if the same goal instance is nominated again.

For example, Figure 2 shows a simple situation that involves the nomination and retraction of a goal. Initially, there is nothing ahead of the agent's car (shown as a green box) moving upwards in the figure. Therefore, it has only a single goal to get to its target location. Then a pedestrian, *ped1* (shown as a yellow smily face), suddenly starts to jaywalk the street in front of the agent's car and it causes a concept instance, *(pedestrian-ahead me ped1)* to match in the state. In response, the system nominates the corresponding goal, *stopped*, and now it has two goals as shown in the second column. When the pedestrian moves away, the relevance condition disappears and the goal is retracted. The agent now has a single goal again, as shown in the last column.

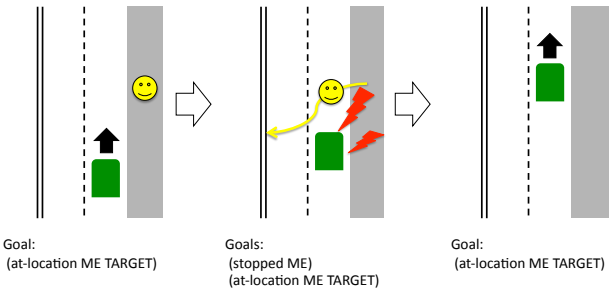


Figure 2: An example of goal nomination process in a driving situation.

## Prioritization Process

So far, we found that it suffices to have the existing boolean descriptions of the world if we are only concerned with the nomination and retraction of goals. This is due to the all-or-none characteristic of the nomination process, which decides whether or not the system should have certain goals at any given time but does not change the ordering of goals from what is initially given to the system. However, we see the necessity for goal prioritization on top of the simpler goal nomination and retraction processes in some situations. For instance, a fire truck driver will observe the traffic rules most of the time as regular drivers would, but when there is an emergency, he might drive on the left side of the road to avoid traffic and run red lights to get to the destination quickly. Situations like this require a change of the priorities among goals (e.g., observing signals vs. getting to the destination), and for this we need a dynamic measure that will affect the priorities.

Since we already have constant priority values, perhaps we need some measure that changes in reaction to the environment, with which we can modulate the priority values and allow reactive prioritization. We focus on the fact that the severity of conditions affect the reactions in human behavior (e.g., whether or not a firetruck driver ignores traffic rules depends on the seriousness of the emergency situation), and propose modulating the constant priority values based on the degree with which the relevance conditions match in the current state. We can compute the *degree of match* for concept instances as scalar numbers between zero, which means that the instance does not hold in the state, and one, which means that the instance is true in the state.

Using the degree of match for the relevance conditions, we modulate the corresponding goal's priority value simply by multiplying them together. Now the priority value for a certain situation would be between zero and the default priority value of the goal. If the relevance condition is close to being true, then the corresponding goal instance gets almost all its default priority value. If the condition is far from being true, its goal gets a lower priority value than the default one. This modulation allows changes in the order of nominated goals based on the situation around an ICARUS agent. Since the continuous match of concepts plays such an important role in this process, we describe the mechanism in a separate section below.

## Continuous Concept Matching

Agents that live in any kind of environment react to what is happening around themselves and maintain a set of beliefs they have about the environment. Therefore, most cognitive architectures support the inference of beliefs based on the descriptive knowledge they have and the information they receive from the environment. As seen earlier in this paper, the ICARUS architecture is no exception, and it provides facilities to infer its beliefs using its concept definitions stored in the long-term conceptual memory and the perceptual information received from the surroundings. In ICARUS, the standard inference process involves matching variables against objects in the world and their attributes. On top of these, there are certain numeric or logical conditions imposed, contributing to the truth value of each concept instance. The inference leads to a boolean value, indicating whether a concept instance is true or not. This *symbolic* matching has served much of our purpose in various domains, but we desire more continuous approach to concept inference for reactive goal prioritization.

We can easily find a source of continuity in concept definitions we have. In dynamic domains, we are most likely to have numeric tests in the definitions of primitive concepts. If we apply a monotonic curve at the boundaries of these tests,

we can get some continuous measures of how close we are to satisfying the tests. We call the variables involved in these tests, *pivots*, to emphasize that they serve as references for the computation of continuous degree of match. It is important to distinguish the degree of match from the probability of match, which describes the possibility of a concept being true, rather than how true a concept is.

As an example, let us use the case of making a right turn while driving a car. Depending on the angle of the turn, there will be an ideal range of the speed and the steering wheel angle for the car an ICARUS agent drives. For a typical 90° turn, we found that the speed between 15 and 20 and the steering angle of 10 degrees show a reasonable behavior. As shown in Table 4, we define two concepts, *at-turning-speed* and *at-steering-angle-for-right-turn* that describe the situation where the agent is ready for a right turn in terms of the speed and steering wheel angle, respectively. The first two concepts are for the standard boolean inference, but the other two include additional information for the continuous concept matching.

Table 4: ICARUS concepts for right turns in the urban driving domain.

```
((at-turning-speed ?self)
 :percepts  ((self ?self speed ?speed))
 :tests     ((>= ?speed 15)
             (<= ?speed 20)))

((at-steering-angle-for-right-turn ?self)
 :percepts  ((self ?self steering ?angle))
 :tests     ((= ?angle 10)))
```

```
((at-turning-speed ?self)
 :percepts  ((self ?self speed ?speed))
 :tests     ((>= ?speed 15)
             (<= ?speed 20))
 :pivot (?speed))

((at-steering-angle-for-right-turn ?self)
 :percepts  ((self ?self steering ?angle))
 :tests     ((= ?angle 10))
 :pivot (?angle))
```

Assuming that we have an agent named 'ME,' the boolean inference process would output *(at-turning-speed ME)* when the agent's speed is between 15 and 20, but nowhere else. For the second concept, it would say *(at-steering-angle-for-right-turn ME)* true, only when the steering wheel angle is 10. When the speed and the angle fall out of the regions even by a miniscule margin, we do not get any of these instances, and there is no way we can tell how bad the situation is for a right turn. If, however, we use the continuous matching capability by designating variables in a new field, `:pivot`, within each of these concepts, the degree of match will tell us how close the current situation is to satisfying these conditions.

When inferring instances for a primitive concept, the extended ICARUS still does the pattern matching of the percepts in the same way as before. But it now applies a monotonic curve around the threshold values for the pivot variable when evaluating numeric tests. For example, in Table 4, the third concept has a pivot variable, `?speed`, and ICARUS constructs a curve like the ones on the first row in Figure 3. For the last concept, the numeric test involves an equality of a pivot variable, `?angle`, to a single number, 10, and the system superimposes two monotonic curves around the number. This results in a bell shape like the ones on the second row in the figure. Although the linear curves on the right column are very simplistic, they sufficiently serve our purpose here to model decreasing degree of match as we move away from the test regions. Ultimately, each test in primitive concepts should have different curves based on their sensitivity to changes in their pivot variables, but for the purpose of this work, we assume they are fixed across all tests.

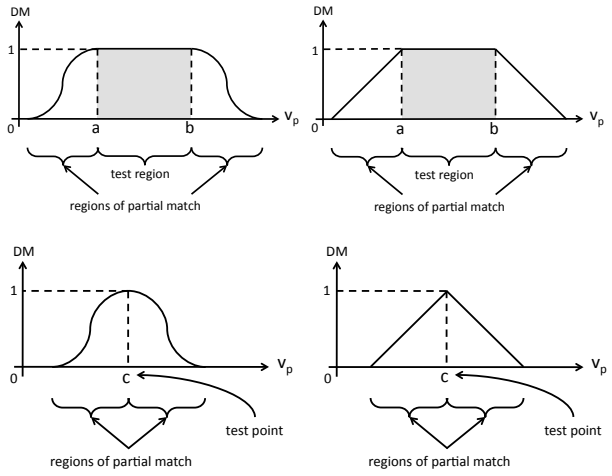


Figure 3: Some monotonic functions applied to numeric tests against pivot variable $v_p$.

Now that we have a way to extract continuous values out of numeric tests, we can propagate them through ICARUS' concept hierarchy. Some primitive concepts have more than one numeric tests, and therefore it is possible to have two or more pivot variables in a primitive concept. This leads to the need for combining multiple degrees of match along these variables. Furthermore, higher-level concepts can consist of multiple concepts with their own degrees of match. Hence, we also need a mechanism to combine two or more degrees of match here. For this, we treat each degree of match as an axis in a multi-dimensional space, and compute the combined degree of match as the vector sum specified by individual degrees of match on different Cartesian axes. After a nomalization, we get a continuous degree of match that propagates upward through the concept hierarchy, enabling us to use higher-level concepts as relevance conditions for goal prioritization.

## Related and Future Work

Our work has been heavily influenced by a long line of work in psychology. One can find a fair amount of work related to motivation and goal selection there. Typically, these work also cover the topic of emotion. (Simon 1967) recognized that the central nervous system, despite being a serial information processor, serves multiple needs in an organism surrounded by unpredictable situations. He suggested that two mechanisms, a goal-terminating mechanism and an interruption mechanism, would satisfy this requirements. Simon further described the relationship among interruption, motivation, and emotion, and outlined an information-processing system that covers these as wells as learning in relation to them.

More recently, (Sloman 1987) suggested that any system with priority in beliefs and actions naturally have emotions. He argued that goals often conflict with each other, and systems must have a mechanism to resolve such conflicts. The author proposed that motivators can serve this purpose. On a different account, (Gray and Braver 2002) explained that the effect of emotion on cognitive control is selective, suggesting a complex functional organization between the two. The authors argued that the integration would be adaptive, and emotion can prioritize conflicting alternatives and tradeoffs. They outlined the emotion-related processing stages, which relates the situation to the behavior through *approach* and *withdrawal* emotions that lead to goals.

There also are some related work in the architectural perspective. CLARION (Sun 2007) and Soar (Laird, Rosenbloom, and Newell 1986) architectures possess their own accounts of goal management. The former is more psychologically positioned, providing interactions between drives and goals. The latter has a rule-based mechanism to nominate its top-level operators as its goals, which resembles the conditionalized goals ICARUS has. Unlike ICARUS, however, Soar proposes a single top-level goal at a time, removing the need for prioritization or the advantage of interactions among multiple goals.

Although the current work is an important first step toward a cognitive architecture with the full capability for goal management, it still ignores a vast amount of psychological accounts on human motivation and goal handling. First of all, we should explain where the long-term knowledge about goals comes from. It is very likely that we will need to deal with even higher-level cognitions like motivations, emotion, and obligations for this purpose. As mentioned earlier, we also need to modify the current extension, so that the system can accept different monotonic curves for concepts and encode various sensitivity levels. We expect evidences from the social psychology literature will help us in the modeling process.

## Conclusions

In this paper, we introduced an extension to the ICARUS architecture for reactive goal management. The extension makes close connections to previous work in psychology and other related fields. The extended framework supports the nomination, retraction, and prioritization of goals based on the current beliefs of an agent. The goal prioritization process uses a novel mechanism for continuous concept matching, which allows partial matching of concepts to provide the continuous measure of relevance for each goal. We described the extension in detail using examples for some everyday driving situations.

## Acknowledgments

The research presented in this paper was performed while supported in part by Grant HR0011-04-1-0008 from DARPA IPTO and Project No. 2E20870 from Korea Institute of Science and Technology. The author would like to thank Pat Langley, Dan Shapiro, and Stellan Ohlsson. Discussions with them contributed to many ideas presented here.

## References

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

Asgharbeygi, N.; Nejati, N.; Langley, P.; and Arai, S. 2005. Guiding inference through relational reinforcement learning. In *Proceedings of the Fifteenth International Conference on Inductive Logic Programming*, 20–37. Bonn, Germany: Springer Verlag.

Choi, D. 2009. Concurrent execution in a cognitive architecture. In *Proceedings of the 31st Annual Meeting of the Cognitive Science Society*. Amsterdam, Netherlands: Cognitive Science Society, Inc.

Gray, J. R., and Braver, T. S. 2002. Integration of emotion and cognitive control: A neurocomputational hypothesis of dynamic goal regulation. In Moore, S. C., and Oaksford, M., eds., *Emotional Cognition: From brain to behaviour*. Philadelphia, PA: John Benjamins Publishing Company. chapter 12, 289–316.

Laird, J. E.; Rosenbloom, P. S.; and Newell, A. 1986. Chunking in soar: The anatomy of a general learning mechanism. *Machine Learning* 1:11–46.

Langley, P., and Choi, D. 2006. A unified cognitive architecture for physical agents. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence*. Boston: AAAI Press.

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

Ohlsson, S., and Rees, E. 1991. Adaptive search through constraint violations. *Journal of Experimental & Theoretical Artificial Intelligence* 3:33–42.

Simon, H. A. 1967. Motivational and emotional controls of cognition. *Psychological Review* 74(1):29–39.

Sloman, A. 1987. Motives, mechanisms, and emotions. *Cognition & Emotion* 1(3):217–233.

Sun, R. 2007. The motivational and metacognitive control in CLARION. In Gray, W., ed., *Modeling Integrated Cognitive Systems*. New York, NY: Oxford University Press.