An Architecture for Adaptive Intelligent Systems[1]

Barbara Hayes-Roth

Knowledge Systems Laboratory
Stanford University
701 Welch Road, Building C
Palo Alto, Ca. 94304

[1] This research was supported by NASA contract NAG 2-581 under ARPA Order 6822, subcontract No. 71715-1 from Teknowledge Federal Systems under ARPA contract No. DAAA21-92-C-0028, and AFOSR grant AFOSR-91-0131. We thank Edward A. Feigenbaum for sponsoring the work in the Knowledge Systems Laboratory. We thank two anonymous reviewers for their many helpful criticisms and suggestions for improving earlier versions of the paper. Many individuals have contributed to different features of the AIS architecture and the Guardian system discussed in this paper: D. Ash, L. J. Barr, L. Boureau, L. Brownston, A. Collinot, V. Dabija, J. Drakopoulos, D. Gaba, G. Gold, M. Hewett, R. Hewett, A.Macalalad, A. Seiver, S. Uckun, A. Vina, R. Washington.

Abstract

Our goal is to understand and build comprehensive agents that function effectively in challenging niches. In particular, we identify a class of niches to be occupied by "adaptive intelligent systems (AISs)." In contrast with niches occupied by typical AI agents, AIS niches present situations that vary dynamically along several key dimensions: different combinations of required tasks, different configurations of available resources, contextual conditions ranging from benign to stressful, and different performance criteria. We present a small class hierarchy of AIS niches that exhibit these dimensions of variability and describe a particular AIS niche, ICU (intensive care unit) patient monitoring, which we use for illustration throughout the paper. To function effectively throughout the range of situations presented by an AIS niche, an agent must be highly adaptive. In contrast with the rather stereotypic behavior of typical AI agents, an AIS must adapt several key aspects of its behavior to its dynamic situation: its perceptual strategy, its control mode, its choices of reasoning tasks to perform, its choices of reasoning methods for performing those tasks, and its meta-control strategy for global corodination of all of its behavior. We have designed and implemented an agent architecture that support all of these different kinds of adaptation by exploiting a isngle underlying theoretical concept: An agent dynamically constructs explicit control plans to guide its choices among situation-triggered behaviors. The architecture has been used to build experimental agents for several AIS niches. We illustrate the architecture and its support for adaptation with examples from Guardian, an experimental agent for ICU monitoring.

1. Toward More Comprehensive AI Agents

"Intelligent agents" continuously perform three functions: perception of dynamic conditions in the environment; action to affect conditions in the environment; and reasoning to interpret perceptions, solve problems, draw inferences, and determine actions. Conceptually, perception informs reasoning and reasoning guides action, although in some cases perception may drive action directly. This abstract definition allows for a great variety of biological and artificial agents whose capabilities range from extremely limited and stereotyped behavior to extremely sophisticated and versatile behavior. Why should different agents exhibit different behavioral capabilities and what underlies these differences?

Differences in their behavioral capabilities allow different classes of agents to function effectively in different niches. A "niche" is a class of operating environments: the tasks an agent must perform, the resources it has for performing tasks, the contextual conditions that may influence its performance, and the evaluation criteria it must satisfy. Human beings are the most sophisticated existing agents. Given their broad range of potential behavior, individual human beings can function effectively in many challenging niches. By contrast, typical AI agents are extremely limited. Given their narrow range of potential behavior, individual agents can function effectively only in a small number (usually one) of severely restricted (usually highly engineered) niches.

We hypothesize that, to a large degree, an agent's architecture determines its potential behavior and, therefore, the niches in which it potentially can function:

Agent Architecture => Potential Behavior => Suitable Niches.

By "architecture" we mean the abstract design of a class of agents: the set of structural components in which perception, reasoning, and action occur, the specific functionality and interface of each component, and the interconnection topology among components. Under this hypothesis, human beings function effectively in many niches that no other animal or existing AI agent could fill--certainly because only human beings have acquired the necessary knowledge and skills, but more fundamentally because only the complex and powerful architecture embodied in the human nervous system [Albus, 1981] supports such a broad range of knowledge and skills.

Conversely, to function effectively in a particular niche, an agent must exhibit the range of behavior required in that niche and, therefore, must have an architecture that supports the required behavior:

Intended Niche => Required Behavior => Sufficient Architectures.

Typical AI agents have simple architectures for good reason: simple architectures are sufficient to support the behavior required in their intended niches. In fact, for restricted niches, architecture often plays a relatively small role in an agent's effectiveness, many alternative architectures may suffice, and architectural design is a relatively insignificant part of the agent-building enterprise. As the intended niche increases in complexity, however, architecture plays a larger role in the agent's effectiveness, fewer alternative architectures will suffice, and architectural design becomes a more critical and expensive part of the agent-building enterprise.

Thus, we argue that present AI agents are "niche-bound" both because they are "knowledge-bound" [Lenat and Feigenbaum, 1991] and because they are "architecture-bound." Increasing only agents' knowledge can expand the very narrow niches in which they currently function. However, it will have diminishing returns as the intended niches increase in complexity and agents' ability to exploit the necessary knowledge and skills runs up against architectural limitations.

Our goal is to provide an architecture for more comprehensive AI agents that function effectively in more challenging niches. Thus, we are working very much in the spirit of Newell's call for "unified theories of cognition" [Newell, 1990]; see also: [Albus, 1991; Laird, et al, 1987]. We focus on a class of "adaptive intelligent systems (AISs)," which operate in a class of niches that is intermediate between the severely restricted niches of typical AI systems and the effectively unrestricted niches of human beings. As discussed below, AIS niches present dynamic variability in their required tasks, available resources, contextual conditions, and performance criteria. As a result, to function effectively in AIS niches, agents must possess a pervasive property of human behavior: *adaptation.* We have designed an agent architecture to support the several dimensions of adaptation required in AIS niches and used it to build experimental agents for several of the domain-specific niches in Figure 1. To ground the discussion, we take examples throughout the paper from a particular niche, patient monitoring in an intensive care unit (ICU), and an

experimental agent called Guardian [Hayes-Roth, et al, 1989; 1992], which was built with our agent architecture.

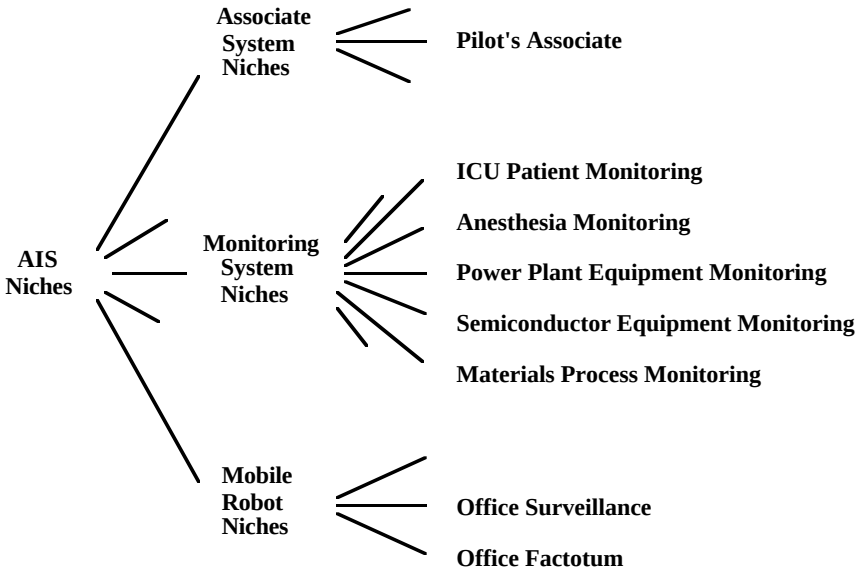

Figure 1. Excerpt from the Class Hierarchy of AIS Niches.

Let us begin by using the ICU monitoring niche to illustrate important shared properties of AIS niches. Intensive care patients are critically ill and depend on life-support devices (e.g., a ventilator) to perform vital functions until their own impaired organs heal and resume normal function, usually a period of several days. The high-level goals of ICU monitoring are to wean the patient from the devices as soon as possible (to minimize cost, discomfort, and undesirable side effects), while detecting and treating any additional problems that arise along the way. Effective patient-management involves: interpretation of many continuously, periodically, or occasionally sensed physiological and device variables; planning and comparative evaluation of many interacting therapy alternatives; detection, diagnosis, and correction of unanticipated problems; control of many patient-management and device-control parameters; and reporting and consulting on patient progress with other members of the ICU team. The complexity of ICU monitoring can overwhelm even skilled clinicians.

Table 1. Shared Properties of AIS Niches versus Typical AI Niches

	AIS Niches	Typical AI Niches
Required Tasks	Diverse, concurrent, interacting	Single isolated task
Available Resources	Variable methods, data, models, facts available	Single correct method, relevant data, and appropriate model
Typical Context	Competing: Percepts, tasks, actions	No competition
Evaluation Criteria	Effective, timely, robust	Correct, efficient, complete

As illustrated by ICU monitoring and summarized in Table 1, AIS niches are considerably more demanding than the niches occupied by typical AI agents. First, *AIS niches require performance of several diverse tasks, sometimes concurrently and often interacting*. For example, ICU monitoring requires tasks such as condition monitoring, fault detection, diagnosis, and planning. Second, *AIS niches provide variable resources for performing tasks*. For example, Guardian has both associative and causal modeling methods for performing diagnosis tasks. It may or may not have the particular class hierarchies or causal relations needed to apply these methods to a given diagnosis problem. Third, *AIS niches entail complex and variable contextual conditions*. For example in ICU monitoring there may be 100 variables sensed automatically several times per second (e.g., blood pressure, pulse), as well as other variables that are sensed irregularly (e.g., laboratory results, x-ray analyses). Data representing these variables differ in criticality and criticality is context-dependent. A patient may manifest several problems simultaneously and therapies for simultaneous problems may interact. Finally, *AIS niches impose more qualitative performance criteria, replacing the usual correctness, efficiency, and completeness criteria with effectiveness, timeliness, and robustness*. For example, if an ICU patient manifests several problems simultaneously, any critical problems must be treated well enough and soon enough to save the patient's life, even if such treatment is sub-optimal and regardless of how many other problems go untreated.

Table 2. Behavioral Adaptations Required of an AIS
versus the Static Behavior of a Typical AI Agent

	Required AIS Adaptations	Typical AI Agent Behaviors
Perception Strategy	Adapt to information requirements and resource limitations	Fixed
Control Mode	Adapt to goal-based constraints and environmental uncertainty	Fixed
Reasoning Tasks	Adapt to perceived and inferred conditions	Single Task
Reasoning Methods	Adapt to available information and current performance criteria	Single Reasoning Method
Meta-Control Strategy	Adapt to dynamic configurations of demands and opportunities	Unnecessary

To function effectively in AIS niches, an agent must be highly *adaptive* (Table 2): it must modify its behavior on each of several dimensions, depending on the situation in which it finds itself. First, *an agent must adapt its perceptual strategy to dynamic information requirements and resource limitations.* For example, when Guardian is monitoring a stable patient, it may divide its perceptual activities among all available patient data in order to maintain a good overview of the patient's condition and remain vigilant to possible problems. However, when it detects a serious problem, Guardian must perceive more selectively, focusing on patient data that help it diagnose the problem and identify an appropriate therapeutic action in a timely manner. Second, *an agent must adapt its control mode to dynamic goal-based constraints on its actions and uncertainty about its environment.* For example, when the patient has a critical, but slowly evolving problem, Guardian can plan and execute an optimal course of therapeutic actions. However, when urgent conditions arise, Guardian must be prepared to react immediately. Third, *an agent must adapt its choices among potential reasoning tasks to dynamic local and global objectives.* For example, when Guardian is monitoring a stable patient, it need only track patient data. When it detects a problem, it must perform a diagnosis task, along with its ongoing monitoring task. After completing its diagnosis, it must perform a therapy planning task, along with its ongoing monitoring task. Fourth, *an agent must adapt its reasoning methods to the currently available information. and performance criteria.* For example, Guardian can use clinical experience to recognize commonly

occurring problems and select standard therapeutic responses. However, when faced with unfamiliar problems, it must fall back on models of the patient's underlying pathophysiology to perform a more systematic diagnosis and design an appropriate therapy. Finally, *an agent must adapt its meta-control strategy to its dynamic configuration of demands, opportunities, and resources for behavior.* For example, Guardian ordinarily interleaves several unrelated or loosely-coupled activities, but may decide to suspend competing activities if a critical problem arises. An effective meta-control strategy may emerge from Guardian's independent decisions regarding co-occurring problems; in other cases it may decide to impose a particular meta-control strategy on a challenging configuration of competing demands and opportunities for behavior.

We have designed and implemented an agent architecture to support the several forms of adaptation required of an AIS. It enables an agent to modify its perceptual strategy, its control mode, its reasoning tasks, its reasoning methods, and its meta-control strategy, depending on relevant features of its dynamic situation. Moreover, our architecture has an important theoretical strength, a kind of architectural parsimony. Its support for all five dimensions of adaptation derives from a fundamental theoretical concept and its architectural embodiment [Hayes-Roth, 1985; 1993a,b]: *An agent dynamically constructs explicit control plans to guide its choices among situation-triggered behaviors.*

The remainder of the paper is organized as follows. Section 2 presents our agent architecture. Sections 3-7 examine the requirements for each of the five dimensions of adaptation in more detail and show how our architecture supports them. Section 8 discusses evaluation of the architecture, summarizes the status of experimental agents built with the architecture, and contrasts the architecture with others in the literature. Section 9 presents conclusions.

2. The Agent Architecture

Our agent architecture (Figure 2) hierarchically organizes component systems for perception, action, and cognition processes. Perception processes acquire, abstract, and filter sensed data before sending it to other components. Action systems control the execution of external actions on effectors. Perception can influence action directly through reflex arcs or through perception-action coordination processes. The cognition system interprets perceptions, solves problems, makes plans, and guides both perceptual

strategies and external action. These processes operate concurrently and asynchronously. They communicate by message passing. Perception-action operations occur at least an order of magnitude faster than cognitive operations.

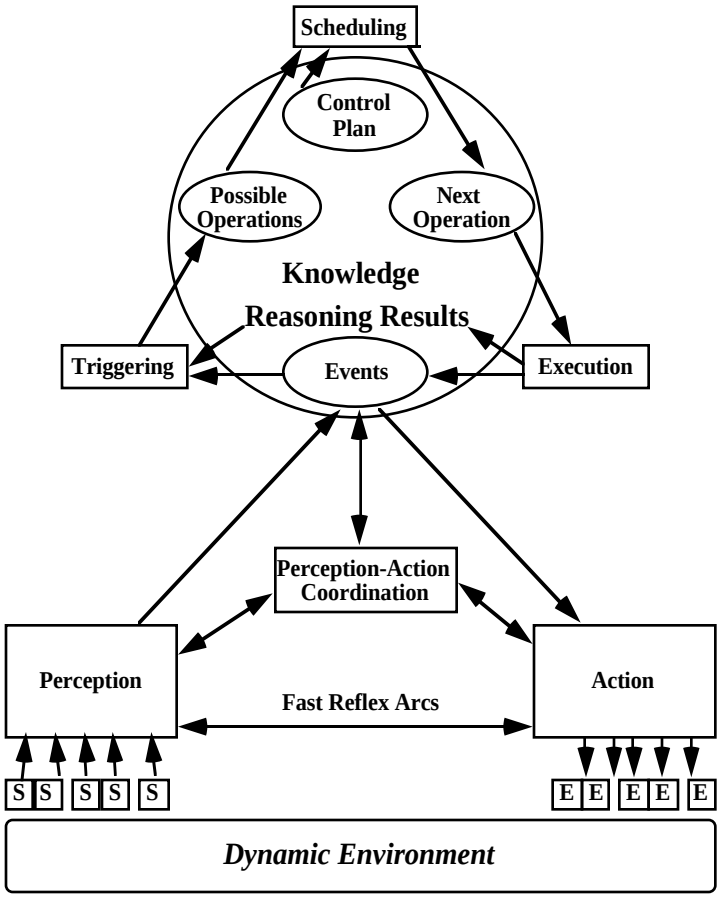

Figure 2. The Agent Architecture.

The cognition system, which is the architecture's most substantial component, is realized as a "blackboard architecture" [Erman, et al, 1980], extended to support dynamic control planning [Hayes-Roth, 1985;1990; 1993b]. For present purposes, we emphasize these features: (a) Perceptual inputs and internal reasoning operations produce changes to a global memory. (b) Each such event triggers some number of possible reasoning operations. (c) Possible operations are scheduled for execution based on active control plans. (d) Control plans are themselves constructed and modified by reasoning operations. (e) Possible actions and control plans are represented in an English-like machine-interpretable language that supports semantic partial matching of actions to plans.

For example, here is one of Guardian's reasoning operations for model-based diagnosis:

Name: Find-Generic-Causes
Trigger: Observe condition, C; where C exemplifies Generic-fault, F
Action: Find generic-faults that can-cause F

Find-Generic-Causes is triggered and its parameters are instantiated whenever a prior reasoning operation indicates that a newly observed patient condition, C, "exemplifies" some generic-fault, F. For example, if C were a decrease in the patient's urine output or inspired air, it would exemplify the generic fault: decrease in the flow of a flow process. When executed, the action of this reasoning operation consults the factual knowledge base and identifies all generic-faults that "can-cause" F (e.g., blockage or leakage of an upstream flow structure can cause a decrease in the flow of a flow process). By recording each such possible cause in the global memory, this operation creates internal events that trigger other reasoning operations. For example, some triggered operations might instantiate possible generic causes with respect to the observed condition, C (e.g., blockage or leakage of various structures in the urinary or respiratory system). Others might continue the backward chaining to identify other generic-faults that "can-cause" those currently hypothesized. Others might attempt to discriminate among alternative hypotheses by examining relevant patient data. To perform a reasoning task such as diagnosis, Guardian triggers and executes a sequence of such reasoning operations, under the control of an appropriate control strategy, incrementally building a solution to the diagnosis problem.

Here is an example of Guardian's control reasoning operations:

Name: Respond-to-Urgent-Problem
Trigger: Observe critical condition, C
Action: Record control decision with
  Prescription: Quickly respond to *C*
  Criticality: Criticality of *C*
  Goal: Diagnosed problems related to *C* are corrected

Respond-to-Urgent-Problem is triggered and its parameter, *C*, is instantiated whenever the perception system delivers an observed condition with a high criticality. When

executed, it creates a control decision to quickly respond to the condition and gives this decision a priority that is proportional to the criticality of *C*. While active, this control decision focuses (some of) Guardian's perception, reasoning (e.g., diagnosis and therapy planning), and action resources on activities related to quickly responding to C. For example, Respond-to-Urgent-Problem could produce a control decision to: Respond quickly to the observed decrease in the patient's inspired air. To identify possible operations that semantically match its control plans, an agent uses explicit knowledge of its own competence as well as its domain: (a) type hierarchies of actions and domain concepts; (b) other relations among actions and concepts; and (c) attached procedures for evaluating modifiers of actions and concepts. Thus, continuing the present example, Guardian would favor execution of possible operations that "quickly" (fast, relative to other operations) "respond to" (monitor, diagnose, correct) the observed decrease in inspired air. A control decision is deactivated when its goal is achieved, in this case, when all diagnosed problems related to *C* have been corrected. Using a small set of general control reasoning operations to generate a variety of specific control decisions, an agent such as Guardian can construct control plans (including plans that have sequential or hierarchical structure) that are appropriate to its situation and it can change those plans as the situation changes [Johnson and Hayes-Roth, 1987].

Figure 3 illustrates the characteristic behavior of agents implemented in this architecture with a simplified episode from Guardian's monitoring of a simulated ICU patient.

At the start of the episode, Guardian has two active control plans: plan A, to update the control plan whenever possible with priority 5; and plan B, to monitor patient data whenever possible with priority 3. Because patient data are always available, the perception system filters continuously sensed patient data and sends selected values to the cognition system at a manageable global data rate. These perceived patient data repeatedly trigger monitoring operations for several variables, including blood pressure and heart rate, all of which match plan B. No events trigger any operations that match plan A. Therefore, for a time, Guardian executes various monitoring operations.

Soon, however, an executed monitoring operation reveals that the patient has abnormally low blood pressure. This observation triggers three new operations, one operation to update the control plan and two alternative operations to begin diagnosing the low blood pressure, all of which compete with recently triggered monitoring

operations. Guardian chooses to update the control plan because that operation matches plan A, its highest priority active control plan. This operation produces control plan C, to respond quickly to the low blood pressure, with priority 3, and lowers the priority of plan B to 1. As a result of the latter change, which is designed to focus resources on the more urgent blood pressure problem, the perception system filters sensed patient data more severely and sends values to the cognition system at a lower global data rate.

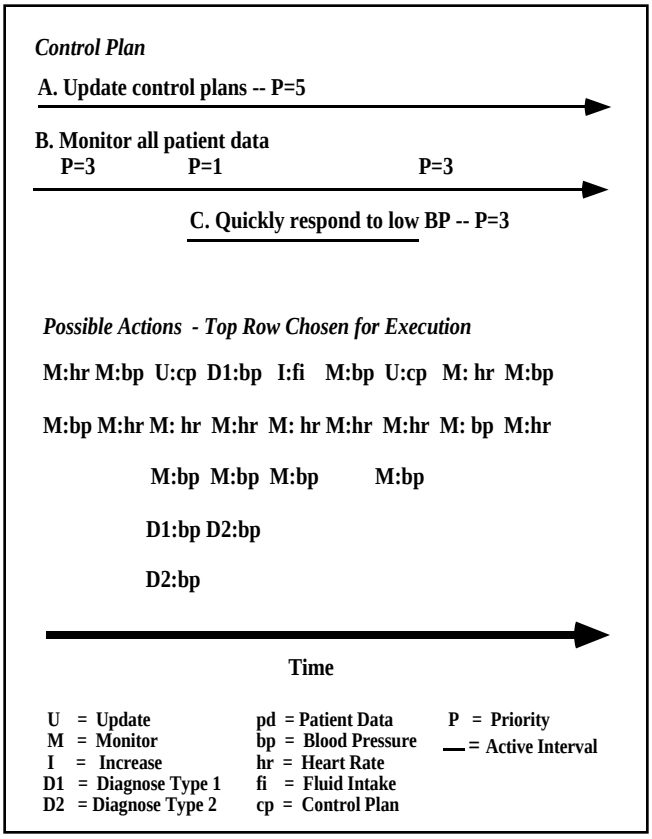

Figure 3. Illustrative Guardian Reasoning Behavior.

Now Guardian executes a series of actions that match plan C, temporarily ignoring repeatedly triggered monitoring actions because of plan B's lower priority. First, Guardian executes one of its pending diagnostic operations--types 1 and 2--for diagnosing the observed low blood pressure. Although both diagnostic operations "respond to low blood pressure" and, therefore, match plan C, diagnosis type 1 matches better because it embodies a "quicker" diagnostic method. Although Figure 3 abstracts Guardian's diagnostic reasoning as a single executed action, in fact diagnosis involves execution of a sequence of reasoning operations. The result of each operation triggers the

next, until the last operation identifies the cause of the high blood pressure, in this case low fluid intake. Identification of this underlying fault triggers an operation to take corrective action (via the action subsystem) by increasing the patient's fluid intake. Guardian executes this operation and, in so doing, triggers an operation to monitor blood pressure, which it expects to rise. This is the last operation Guardian executes under plan C. (Although this simple example involved only a single corrective action, Guardian is capable of performing several corrective actions in parallel--coordinated actions to address a single problem or separate actions to address different problems.)

Confirmation of normal blood pressure indicates that the goal of plan C has been achieved, which triggers a new operation to update the control plan. Guardian executes this operation because it matches plan A, the highest priority active plan. It deactivates Plan C and returns the priority of Plan B to 3. As a result, the perception system filters sensed patient data less severely and sends values to the cognition system at its original higher global data rate. Guardian returns to executing monitoring operations repeatedly triggered by perceived patient data and chosen for execution under plan B.

This example illustrates the architectural mechanism underlying our fundamental theoretical concept: that an agent dynamically constructs explicit control plans to constrain and guide its choices among situation-triggered possible behaviors. Guardian always has some number of active control plans, varying in priority. Some control plans are quite general and favor the execution of a large class of potential operations. Others are more specific and distinguish operations that will help Guardian achieve well-defined objectives. Although the example of Figure 3 shows only simple one-sentence control plans, the architecture allows (and Guardian typically employs) more complex control plans having sequential and hierarchical structure. Similarly, Guardian always has some number of possible behaviors. Some are triggered by inputs from its perception system, while others are triggered by the results of prior reasoning operations. Different operations, if executed, could change Guardian's knowledge of the environment, initiate or extend its performance of particular reasoning tasks, initiate performance of external actions by its action system, or modify its active control plans. At each opportunity, Guardian performs behaviors that best match its highest priority active control plans.

In the following sections, we show how this key architectural mechanism enables an agent to adapt its perceptual strategy, control mode, reasoning tasks, reasoning methods, and meta-control strategy to its dynamic situation.

3. Adaptation of Perceptual Strategy

In order to perform effectively in AIS niches, *an agent must adapt its perceptual strategy to changing cognitive requirements.*

In theory, we might like an agent to perceive all events in its environment and to reason about them in all promising ways, so that it can determine and carry out optimal courses of action. However, AIS niches present high, variable data rates for many environmental conditions; a resource-bounded agent cannot realize unbounded perception. In addition, AIS niches permit many different reasoning tasks and sometimes different methods for performing particular tasks. Each perceived event initiates a potential cascade of reasoning activities; the event itself triggers a number of possible reasoning operations, each of which produces new events, each of which triggers new operations, and so forth. Even if unbounded perception were feasible, the high and cascading demand for reasoning would swamp the cognitive resources of an agent such as Guardian (or a human being, for that matter).

In general, a resource-bounded agent ordinarily cannot--and, equally important, need not--perceive, reason, or act on every condition in its environment. Instead, the agent must be highly selective in its perception of the environment and it must adapt its perceptual strategy to balance two objectives. First, from a purely quantitative perspective, the agent must maximize its vigilance, perceiving as much information as possible about as many environmental conditions as possible, while avoiding perceptual overload. Second, from a qualitative perspective, the agent must maximize goal-directed effectiveness, readily acquiring data that are relevant to its currently important reasoning tasks, while avoiding distraction by irrelevant or insignificant data.

In our architecture, the perception system's basic functions (Figure 4) are to abstract, prioritize, and filter sensed data before sending it to the cognition system. Five parameters determine how the perception system performs these functions. Two static compile-time parameters identify the domain variables to be sensed and ranges of critical values for those variables. Three dynamic run-time parameters (sent asynchronously by the cognition system) specify requested data abstractions, relevance values for different variables, and the desired global data rate. The perception system processes and sends to the cognition system all requested data abstractions at appropriately high rates and sends

unrequested data at appropriately low (but usually non-zero) rates. It dynamically determines the "appropriate" rates at which to send each requested and unrequested observation by distributing the current desired global data rate among them in proportion to their current relevance values. There is one exception to this rule: The perception system sends critical values for all sensed variables, regardless of their current relevance values. Note that, since many variables are not sensed continuously, this provision does not guarantee perception of every critical value that occurs, but only every critical value that is sensed.

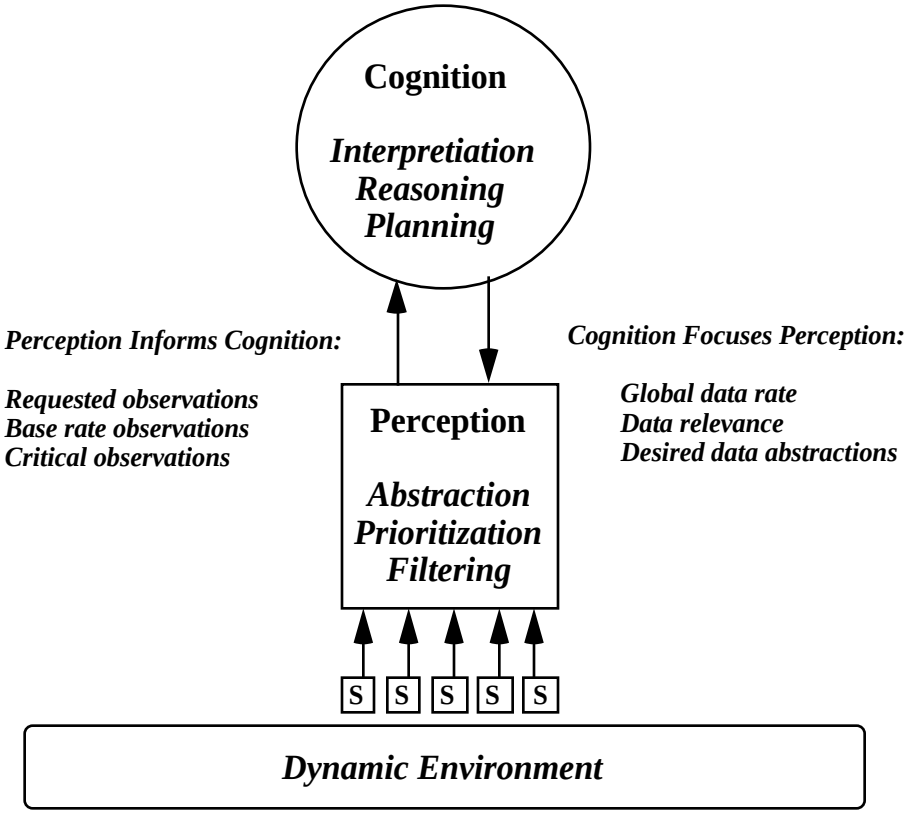

Figure 4. Coordination of Cognition and Perception.

An agent dynamically adapts its perceptual strategy by modifying its three run-time parameters based on both feedback control and predictive control from the cognitive system.

The agent adapts its global data rate based on feedback control from activities in its cognitive system's limited-capacity event buffer. The event buffer is designed to insure that the cognitive system always retrieves the most important, up-to-date perceptions available. Events are ordered in the buffer by priority and recency, with best-first retrieval and worst-first overflow. (In theory, the buffer mechanism also uses a decay factor to remove unretrieved, out-of-date events, but we have not yet implemented a decay factor.)

The specific function used to integrate priority and recency factors to order events in the buffer should be tailored to characteristics of the agent's niche. For example, Guardian's niche presents events that vary widely in priority, with very high priority events occurring infrequently. Important features of its environment change relatively slowly and its deadlines are relatively long compared to the speed of its perception. Therefore, Guardian orders perceptual events by priority and then by recency. It always retrieves the most recent of the most important events and, in case of overflow, loses the least recent of the least important events. In practice, when critical events occur, Guardian retrieves and reasons about them immediately. When multiple critical events co-occur during a brief time interval, Guardian handles them promptly in priority order. Most of the time, however, no critical events occur and Guardian processes all of the incoming events within a few retrieval cycles of their arrival in the buffer--the exact order has no effect on the overall utility of its performance.

Regardless of the specific event-ordering function used, the buffer mechanism is designed for steady-state operation in which: (a) perceptual events enter and leave the buffer at roughly equal rates; and (b) all of the entering events ultimately are retrieved for reasoning. However, steady-state operation assumes that the perception system has been parameterized with a global data rate that is appropriate for the agent's reasoning rate. If there is a decrease in the pace of the agent's reasoning or an increase in the rate of critical sensed events, the event buffer will overflow--this is the architecture's "last line of defense" against perceptual overload. When the event buffer overflows, it means that the agent's reasoning cannot keep pace with perceptual events and, although it is still reasoning about up-to-date events, the agent is losing potentially important information. Conversely, when the buffer underflows (i.e., is empty), it means that the agent is waiting for perceptual events to reason about and, in the meantime, wasting cognitive resources. In either case, the agent corrects the imbalance between perception and reasoning rates by modifying the desired global data rate used by the perception system.

An earlier version of this feedback mechanism implemented a "bang-bang" control response to actual over(under)flow. The current version implements an adaptive control response by monitoring trends in the number of items in the buffer and adjusting the global data rate in anticipation of over(under)flow occurs. As in conventional control applications, the adaptive control gives a smoother correlation between desired and actual global data rates.

The agent also adapts its global data rate predictively. It analyzes newly created or modified control plans to estimate its own future demand for cognitive resources and, complementarily, its future capacity to process perceptual events. Based on this estimate, it may increase or decrease its global data rate. For example, when Guardian adopts plan C in Figure 3, to respond quickly to the patient's low blood pressure, it knows: (a) that the associated reasoning tasks will consume computational resources previously consumed by monitoring a variety of patient data; and (b) that the new tasks are more important than the monitoring task (priority 3 versus 2). It lowers its global data rate. Conversely, it raises its global data rate after achieving the goal of plan C. With a little knowledge about the computational properties of different reasoning methods, an agent can predictively modulate its global data rate more precisely.

The agent also analyzes control decisions to identify useful data abstractions and to determine the context-specific relevance of different variables. For example, in plan B of Figure 3, Guardian's decision to monitor all patient "data" implies that the perception system should send the raw numeric data available from its sensors for all patient variables. Alternatively, given appropriate definitions for various data abstractions, Guardian might decide to monitor "criterial changes in value," "hourly high and low values," "running averages," etc. Plan B's initial priority of 3 translates into a mid-range relevance value for all patient variables. Guardian's subsequent reduction of plan B's priority translates into a reduction in relevance. However, Guardian's simultaneous introduction of plan C, to respond to the patient's low blood pressure with priority 3, preserves the medium relevance value for blood pressure. Although we do not illustrate it in Figure 3, Guardian also could identify other variables that are relevant to its reasoning under plan C, either based on explicit domain knowledge or in the course of the reasoning itself.

In summary, our architecture enables an agent to adapt its perceptual strategy to its cognitive requirements in two ways. First, the agent maximizes its vigilance, while

avoiding perceptual overload, by using feedback control and predictive control (based on control plans) to manage the global data rate underlying in its perceptual strategy. Second, the agent acquires useful data, while avoiding distraction, by using dynamic control plans to adapt the relevance and abstraction parameters underlying its perceptual strategy. In an early experiment [Washington and Hayes-Roth, 1989], Guardian's adaptation of its perceptual strategy reduced its input data rates to less than 10% of the original sensed data rates, with no degradation in the quality of its performance.

4. Adaptation of Control Mode

In order to function effectively in AIS niches, *an agent must adapt its control mode to changing features of its control situation.*

We can characterize control situations on several dimensions, including the uncertainty of events in the task environment, the degree of constraint on which sequences of actions will be effective in achieving goals, and the availability and cost of off-line and on-line computational resources. In simple niches, a characteristic control situation--representing a particular configuration of values on these several dimensions--may predominate. In that case, an agent should adopt the control mode that is most effective in its predominant situation. For example, the most effective control mode for some niches may be to plan and then execute carefully coordinated sequences of actions [Fikes and Nilsson, 1971; Sacerdoti, 1975; Wilkins, 1984], while in other niches the most effective control mode may be to prepare and then execute more localized reactions to a range of possible run-time events [Agre and Chapman, 1987; Nilsson, 1989; Rosenschein and Kaelbling, 1986; Schoppers, 1987].

However, AIS niches do not present characteristic control situations; they present control situations that vary over time on several dimensions. Two salient dimensions of variability, which we analyze here, are environmental uncertainty and constraint on effective actions. Environmental uncertainty determines how much monitoring an agent must do to determine run-time conditions. For example, a cold post-operative ICU patient presents low uncertainty; the patient is probably cold as a natural consequence of the surgery and quite likely to warm up gradually to normal body temperature, with no lingering after-effects. By contrast, a patient whose blood pressure is falling presents high uncertainty; it is unknown how long or how far the blood pressure will fall, what is causing the change, and what related effects may occur. Constraint on effective actions

determines how many alternative courses of action the agent can pursue to achieve its goals. For example, there are many ways to help a cold post-operative patient regain normal body temperature, but there is only one way to enable a patient with a severe pneumothorax (a hole in the lung) to breathe: surgically insert a chest tube to allow accumulated air in the chest cavity to escape and thereby enable the lungs to inflate. As illustrated in Figure 5, these two dimensions define a space of control situations.

To function effectively in AIS niches, therefore, an agent must possess and exploit a corresponding variety of control modes.

Like control situations, we can characterize control modes along several dimensions. Two salient dimensions, which we analyze here, are: the agent's sensitivity to run-time events and its advance commitment to specific actions. Sensitivity to run-time events measures how much the agent monitors its run time environment. Commitment to specific actions measures how much the agent restricts in advance the actions it will execute at run time. (Control modes also vary, for example, on their demands for off-line and on-line computational and real-time resources; however these variables are not included in our analysis.) As illustrated in Figure 5, these two dimensions, sensitivity and commitment, define a space of control modes.

By superimposing the spaces of control modes and control situations in Figure 5, we suggest that particular control modes are appropriate for particular control situations--and, more importantly, that an agent could use a similar dimensional analysis to identify its dynamic control situation and adapt its control mode as appropriate. Let us consider the four corners of the space of control modes.

In a pure planning mode, an agent commits in advance to a sequence of actions, perhaps with limited conditionality, and then executes it at run time with minimal monitoring of run-time events. Planning mode is appropriate for control situations with low environmental uncertainty and high constraints on the selection and sequencing of effective actions. At the cost of preparation time, the agent exploits predictability in its environment to construct and execute an effective, efficient plan. For example, when requested to make patient presentations for physicians on rounds, Guardian should follow a standard protocol for presenting the relevant information in the correct order.

In a pure reactive mode, the agent commits in advance to a set of specific actions and conditions for their execution, but monitors run-time events to control invocation of particular actions from the set. Reactive mode is appropriate for control situations with high uncertainty and high constraints on effective actions. At the cost of preparation time and run-time resources, the agent can exploit its monitoring capabilities to respond flexibly to an uncertain environment. For example, Guardian should operate in reactive mode when responding to critical problems under time pressure, such as reacting to an observed increase in a patient's peak inspiratory pressure (a potentially life-threatening condition) by monitoring relevant data closely and using them to choose among a small set of predetermined diagnoses and associated therapeutic actions .

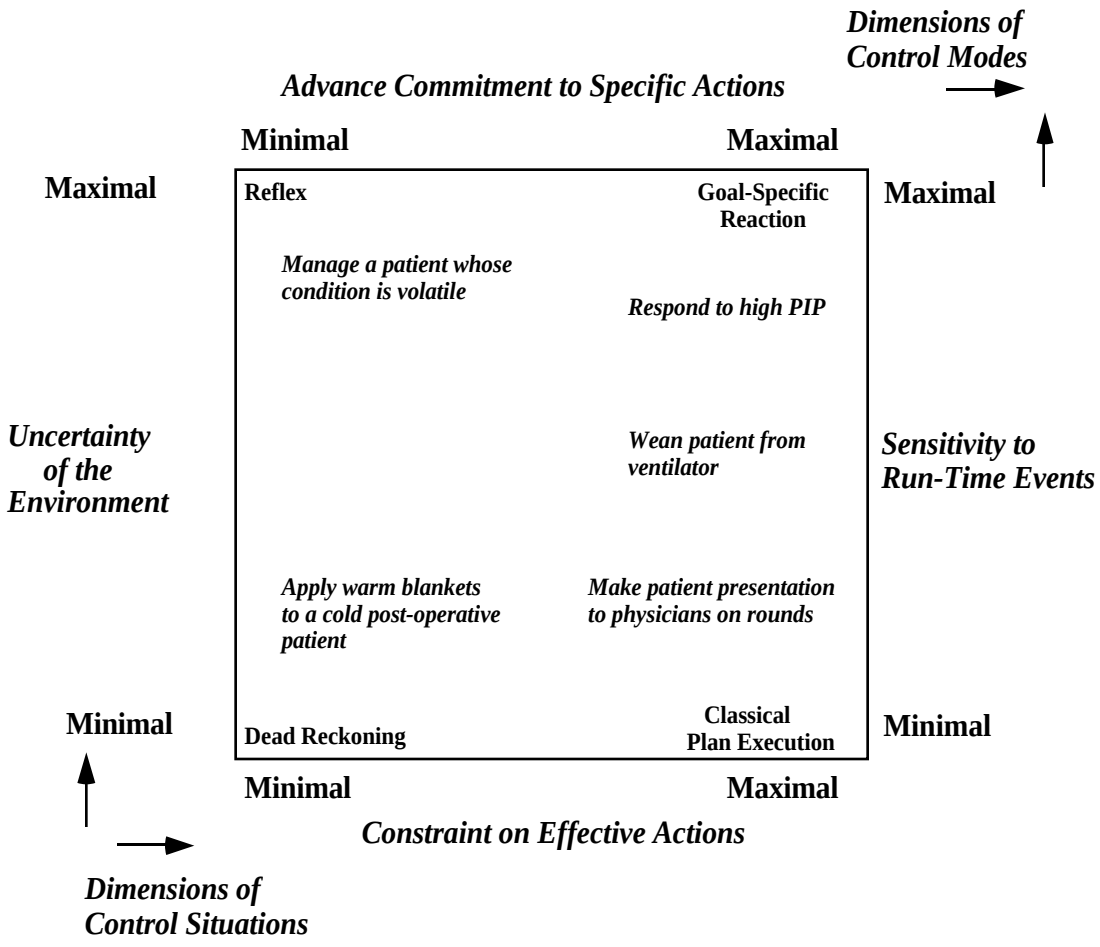

Figure 5. Different control modes for different control situations.

In both planning and reaction modes, an agent commits in advance to specific executable actions in order to meet strong constraints imposed by its goals. Planning mode exploits

predictability of environmental events to minimize monitoring while following a single globally coordinated action sequence; thus it streamlines run-time performance. Reaction mode copes with greater uncertainty of environmental events by preparing a larger number of actions for a larger number of contingencies; run-time performance is less streamlined, but more robust. In intermediate modes between these two extremes, the agent modulates the amount of run-time monitoring and the conditionality of actions. In all regions along this border, however, the agent pays a high cost in advance preparation to choose the specific conditions it will monitor and the specific actions it will perform. The agent is maximally committed and can not respond to a truly unanticipated event or perform a truly unanticipated action.

In what we might call a pure "dead reckoning" mode, an agent commits to a rough sequence of a few classes of actions and executes any sequence of specific actions within each successive class at run time. Dead reckoning mode is appropriate for control situations with minimal uncertainty of the environment and minimal constraints on actions. The agent can produce satisfactory behavior with a low cost of advance preparation. For example, Guardian should operate in dead reckoning mode when it has weak goals for non-critical conditions and plenty of time, such as improving the comfort and condition of cold post-operative patients by taking any of several different actions to help them warm up during their first couple of hours in the ICU.

In what we might call a pure "reflex" mode, the agent commits to a large class of actions, without specifying any of them individually, and monitors a similarly large set of run-time conditions to control its selection of actions for execution. Reflex mode is appropriate for control situations with high uncertainty and low constraint on effective actions. The agent can maximize its flexibility with a low cost of advance preparation. For example, Guardian should operate in reflex mode when a patient is very volatile, monitoring a broad class of patient data and letting observed irregularities elicit corrective actions.

In both dead reckoning and reflex modes, an agent is positioned implicitly to perform a larger number of specific actions and action sequences, compared to planning and reactive modes, respectively. Dead reckoning mode exploits predictability in environmental events to predetermine only the general shape of behavior, which the agent can instantiate as any of many alternative appropriate courses of action at run time. Reflex mode copes with greater environmental uncertainty by relying on run-time monitoring to invoke appropriate actions.

In the intermediate modes between these two extremes, the agent modulates the amount of run-time monitoring and the balance of top-down versus bottom-up control of actions. In all modes on this border, however, the agent pays a minimal cost of advance preparation by identifying arbitrarily large classes of events to monitor and arbitrarily large classes of actions to perform. It does not commit to monitor any specific events or to perform any specific actions at all. Thus, unlike planning and reaction modes, the agent is always responding to “unanticipated” events and performing “unanticipated” actions.

Our analysis assumes that an agent has adequate monitoring and preparation resources for any control mode, but that, other things equal: (a) it prefers to spend resources on preparation rather than on monitoring in order to maximize the efficiency and global structure of run-time performance; and (b) it prefers to spend less resources when that will not compromise its goals. Alternatively, if we assume variations in availability or cost of these resources, the superimposed spaces show how run-time performance may be degraded in order to conserve particular resources. A more comprehensive analysis would introduce availability and demand for monitoring and preparation resources as higher-order dimensions of the superimposed spaces. The purpose of our analysis in the present context is to partially characterize the variability of control situations and differences in the situation-specific efficacy of alternative control modes.

To function effectively in AIS niches, an agent must continually identify its control situation, choose an appropriate control mode, and implement the chosen mode. We use examples from Guardian's niche to illustrate how our architecture supports this kind of adaptation.

First, an agent must identify its control situation. The agent can assess the uncertainty of its environment by recognizing that it is in states with known uncertainty. For example, Guardian might know that certain surgical procedures are more likely than others to be followed by recovery problems (higher uncertainty) in the ICU. The agent also can assess uncertainty empirically at run time, tracking the variance in its observations over time, noticing that planned actions are not having their expected effects, etc. The agent can assess the constraint on effective actions based on domain knowledge or on measurements of the search space associated with a particular goal. For example, Guardian might know that physicians want all patient presentations to follow the standard protocol (high constraint). As mentioned above, control situations also vary

along other dimensions, such as the availability of computational and real-time resources during and prior to run time. As discussed in section 3, an agent can estimate current and future demand for computational resources by analyzing its current and future control plans. It can estimate the availability of real-time resources based on domain knowledge. For example, Guardian might know that some ICU problems evolve slowly, while others quickly become life-threatening.

Next, the agent must choose an appropriate control mode. The superimposed spaces in Figure 5 provide one framework for making this choice. As mentioned above, control modes also vary along other dimensions, such as their demand for computational and real-time resources during and prior to run time. An agent can have qualitative knowledge of the resource requirements associated with generic control modes, such as those in Figure 5. In addition, it might be able to quantify the requirements for a particular control mode in a particular parameterized situation.

Having identified its control situation and chosen an appropriate control mode, the agent must effect the chosen mode. Figure 6 summarizes how our architecture enables an agent to adapt its control mode, modulating its sensitivity to run-time events and its commitment to specific actions by manipulating two properties of its control plan: the specificity of action classes indicated in each component control decision and the degree of sequential organization among control decisions. Again we illustrate this capability with the control modes in the four corners of the space.

The agent goes into a pure planning mode by constructing a control plan that comprises a sequence of decisions, each identifying a specific executable action. It monitors only those events that are necessary to trigger the current next action in the plan. It tries to trigger only each successive next planned action. As a result, the agent triggers and executes the planned sequence of specific actions very quickly and reliably.

The agent goes into reactive mode by constructing a control plan that comprises an unordered set of decisions, each identifying a specific condition-action contingency. It monitors only those events necessary to evaluate the specified conditions. It attempts to trigger only the specified actions and executes whichever ones it triggers. As a result, the agent executes a less predictable sequence of a reliable set of planned actions. It is a little slower than in planning mode because it monitors all conditions all the time.

The agent goes into dead reckoning mode by constructing a control plan that comprises a sequence of a few general action classes. It monitors only those events that might trigger any member of the current planned action class. It attempts to trigger only actions that are members of the current planned action class and executes whichever ones it triggers until the local goal of the current planned action class is met. As a result, the agent executes a roughly predictable sequence of certain kinds of actions, with variability the number and specific identities of executed actions within each successive class.

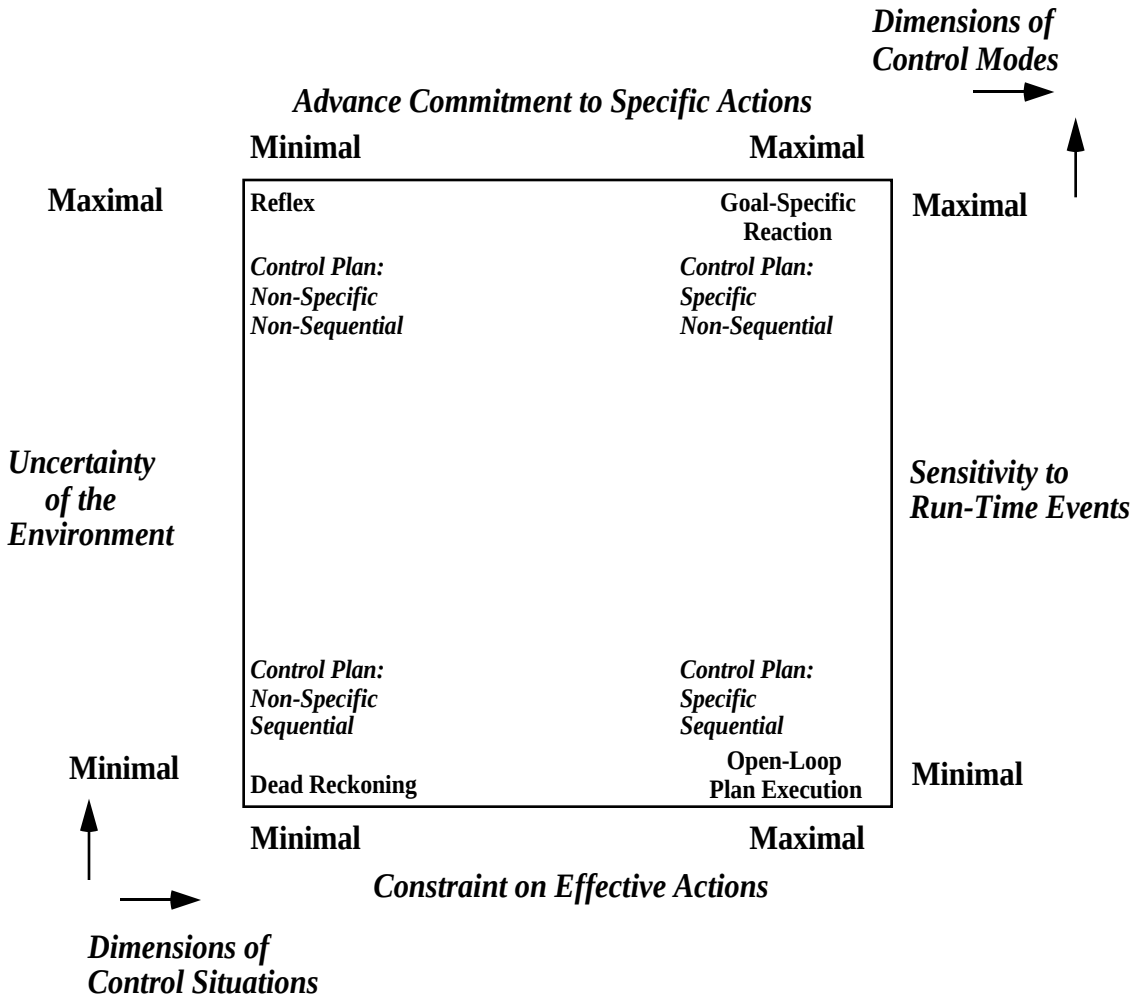

Figure 6. Varying Control Plan Properties to Effect Different Control Modes.

The agent goes into reflex mode by constructing a plan that comprises an unordered set of decisions, each identifying a class of condition-action contingencies. It monitors only those events that might trigger any member of any of the action classes and executes

whichever ones it triggers. As a result, the agent's behavior is quite unpredictable in the number, identities, and sequence of specific actions it executes.

Our analysis can potentially be translated into the language of classical control theory. For example, the border between plans and reactive systems corresponds to the control theoretic distinction between open-loop and closed-loop policies. Techniques for choosing optimal control modes also exploit our concept of uncertainty. Thus, it is known that in a deterministic environment, an optimal open-loop policy exists, while in a stochastic environment there exists a closed-loop policy that performs better than any open-loop policy. To our knowledge, adaptive control theory does not exploit our concept of constraint on actions (which corresponds to the control theoretic concept of solution density) as a basis for prescribing control modes. In addition, although there exist control theoretic approaches to run-time switching of control modes [Rugh, 1990], these approaches typically switch among a much more homogeneous set of alternative controllers in the context of much simpler task environments. Finally, control theoretic approaches do not provide a framework for smooth transitions in a continuous space of controllers.[2]

In summary, our architecture enables an agent to adapt its control mode among a diverse set of control modes, based on its environmental uncertainty and internally determined constraints on its actions by modifying two key parameters of its control plans: specificity and sequential organization of component control decisions.

5. Adaptation of Reasoning Tasks

In order to function effectively in AIS niches, *an agent must adapt its reasoning tasks to dynamic environmental conditions.*

In general, AIS niches demand performance of multiple component reasoning tasks. We define a task in terms of the types of domain entities it takes as inputs and produces as outputs and the relationships that must hold between particular instances of inputs and outputs. For example, in a diagnosis task, inputs are observed symptoms in a monitored system, outputs are conditions within the system, and the relationship is that the conditions caused the symptoms. For example, Guardian might diagnose the

[2] I am grateful to Satinder Pal Singh for calling my attention to the relationship between the present analysis and the classical control-theoretic analysis.

physiological condition that is causing a patient's observed low blood pressure. As a second example, in a prediction task, inputs are initial conditions in a monitored system, outputs are subsequent conditions in the system, and the relationship is that the subsequent conditions have a high conditional probability given the initial conditions. For example, Guardian might predict the consequences of leaving the patient's low blood pressure untreated. As a third example, in a planning task, inputs are initial conditions and desirable subsequent conditions, outputs are specifications for a pattern of actions, and the relationship is that performing the planned actions in the context of the initial conditions is expected to bring about the desirable subsequent conditions. For example, Guardian might plan therapeutic actions to raise the patient's low blood pressure back to normal range. An agent such as Guardian has many opportunities to perform different instances of each of these tasks and to perform sequences of related tasks, for example perceiving a problem, diagnosing it, then planning and executing a corrective response.

Our agent architecture provides a natural platform for realizing and integrating performance of diverse, potentially interacting tasks. As discussed in section 2, each of an agent's reasoning methods is operationalized as a set of event-triggered reasoning operations, including some that construct control plans to organize the reasoning process appropriately in particular situations. Execution of each reasoning operation contributes to an incrementally growing solution in global memory and produces events that may trigger other reasoning operations. This "blackboard model" for reasoning is extremely general; it can support the inferential processes underlying many different reasoning tasks and potential interactions among tasks based on intermediate, as well as final results [Jagannathan, et al, 1989; Engelmore and Morgan, 1988].

Within the architecture, any perceptual or cognitive event potentially can trigger operations involved in any known task. All triggered operations are placed on a global agenda, where they compete to be scheduled and executed. Depending on its control plan, the agent may execute all of the operations in a given reasoning task prior to beginning a new one or it may interleave the operations of several tasks. In either case, the intermediate and final results of different tasks are recorded in the global memory, where they can influence one another. In section 2 above, we illustrated how Guardian initiates, performs, and terminates a single task in response to a perceptual event: it reactively diagnosed and corrected a problem signaled by perceived low blood pressure. More generally, the event-based triggering of task-specific operations allows an agent to adapt its selection and sequencing of reasoning tasks to perceived conditions in the external

environment and to internally generated conditions reflecting the intrinsic relations among reasoning tasks.

Figure 7 illustrates Guardian's performance of a series of interacting perception, reasoning, and action tasks in which each task is triggered by preceding perceptual or cognitive events and produces cognitive events of its own that may trigger subsequent tasks. Figure 7 does not show the triggering, execution, and results of each task's component reasoning operations. And it does not show the many triggered tasks that Guardian does not choose to execute.

In step 1, Guardian is observing a variety of patient data. It intends that the patient it is monitoring should be in a "normal" state (normal for a particular class of post-surgical patients) and, because it is not aware of any problems, expects that all patient data will be normal. As we shall see, much of Guardian's reasoning is driven by discrepancies between phenomena it *observes, expects* and *intends*. Our agent architecture makes these distinctions explicit and automatically detects mismatches to trigger reasoning activities.

In steps 2-3, Guardian detects an oxygen delivery problem and partially diagnoses it. In step 1, Guardian perceives patient data available from its sensors and produces a number of observations. In step 2, one of the new observations, the new value of PIP (peak inspiratory pressure), triggers a task to assess the dynamic state of the patient's PIP. The assessment task produces a new observation, that the patient's PIP is high and has been rising during the time interval t1-t3. This observation violates Guardian's expectation of normal patient data and so, in step 3, triggers a task to diagnose the cause of the discrepancy. The diagnosis task itself produces two results: a hypothesis that the patient is suffering from a compliance problem (inability to inhale sufficient air; as opposed to a sensor error in PIP measurement or a mechanical problem in the ventilator, for example); and an expectation that, as a result, the patient's arterial oxygen will be low.

At this point, the diagnosis is not complete: Guardian does not know what is causing the compliance problem. However, because the expected low arterial oxygen violates Guardian's intention that the patient should be in a normal state and because arterial oxygen is a life-critical physiological parameter, it does not immediately continue the diagnosis task. Instead, in step 4, the expectation of low arterial oxygen triggers a planning task to improve the patient's arterial oxygen now. The planning task produces an intention to raise the FIO2 now (increasing the fraction of inhaled oxygen delivered by

the ventilator), with the conditional expectation that doing so will raise the patient's arterial oxygen gradually. In step 5, the intention to raise FIO2 now triggers the corresponding action and an associated perceptual task to confirm successful execution of the action. Guardian observes that it has indeed raised the patient's FIO2 and, as a result, expects the arterial oxygen to rise. Note that, in the present scenario (without an oximeter in place), Guardian cannot observe the arterial oxygen directly and so must rely on the expected effects of its action of raising the FIO2.

Triggering Events	Tasks Performed	Resulting Events
1. Sense: Patient data	Perceive: Patient data	Observe: Many data values Intend: Normal patient state Expect: Normal patient data
2. Perceive: PIP value	Assess: PIP values	Observe: PIP high, rising t1-t3
3. Observe: PIP high, rising t1-t3	Diagnose: PIP rising	Hypothesize: Compliance Expect: Low arterial O2
4. Expect: Low arterial O2	Plan: Improve arterial O2 now	Intend: Raise FIO2 now Conditionally Expect: Raise arterial O2 gradually
5. Intend: Raise FIO2 now	Do: Raise FIO2 Perceive: FIO2 setting	Observe: Raise FIO2 Expect: Raise arterial O2 grad.
6. Hypothesize: Compliance	Diagnose: Compliance	Hypothesize: Pneumothorax t1 Expect: Falling arterial O2 Expect: Possible death > t8
7. Expect: Falling arterial O2 Expect: Possible death > t8	Plan: Lower PIP now and Normalize arterial O2	(a) Step 1 Intend: Insert chest tube now Conditionally Expect: Lower PIP now Conditionally Expect: Raise arterial O2 promptly
8. Intend: Insert chest tube now	Do: Insert chest tube Perceive: Chest tube insertion Perceive: PIP data	Observe: Chest tube inserted Observe: Lower PIP now Expect: Raise arterial O2

9. Observe: Lower PIP	Plan: Normalize arterial O2	(b) Step 2 Intend: Lower FIO2 now Conditionally Expect: Normal arterial O2
10. Intend: Lower FIO2 now	Do: Lower FIO2 Perceive: FIO2 setting	Observe: Lower FIO2 Expect: Normal arterial O2

Figure 7. Illustrative Chain of Reasoning Tasks Initiated by Cognitive and Perceptual Events.

In step 6, Guardian continues its diagnosis of the oxygen delivery (compliance) problem. The previous hypothesis of a compliance problem (produced in step 3), which violates Guardian's intention of normal patient state, triggers a task to diagnose the underlying cause. This task produces three results: a more specific hypothesis, that the patient suffered a pneumothorax (a hole in the lung that allows inhaled air to rush out into the chest cavity, compressing the lungs and preventing subsequent inhalation) at time t1; an expectation that, as a result of the pneumothorax and despite Guardian's having raised the FIO2, the patient's arterial oxygen will continue to fall; and a second expectation that, as a result of the falling arterial oxygen, the patient may die after time t8.

In step 7, these two expectations, which dramatically violate Guardian's intention of normal patient state, trigger a two-part planning task: (a) to lower the patient's PIP now so that any oxygen at all can be delivered; and (b) to normalize the patient's arterial oxygen. The first part of the planning task produces step 1: an intention to insert a chest tube immediately (to relieve pressure in the chest cavity and enable the lungs to inflate), with the conditional expectation that doing so will lower the patient's PIP immediately and, as a result, raise the arterial oxygen promptly.

At this point, the plan is not complete: Guardian has not determined how to normalize the patient's arterial oxygen. However, because the patient is in a life-threatening condition, it does not immediately continue its planning task. Instead, in step 8, the intention to insert a chest tube now triggers the corresponding action and associated perceptual tasks to confirm the insertion of the chest tube and the expected lowering of the patient's PIP. Again, Guardian cannot observe the expected rise in arterial oxygen directly and must rely on the expected effects of lowering the patient's PIP in the presence of a pneumothorax.

In step 9, Guardian's confirmation of the expected lower PIP triggers resumption of its interrupted planning task, producing step 2: an intention to lower the FIO2 (back to its previous level), with the conditional expectation that arterial oxygen will gradually return to normal. In step 10, this intention triggers the corresponding action and an associated perceptual task to confirm the new FIO2 setting. Again, Guardian expects, but cannot observe directly, that the patient's arterial oxygen gradually will return to normal. At this point in the scenario, with the crisis apparently resolved and the time pressure eased, Guardian may decide to place an oximeter so that it can monitor the patient's arterial oxygen directly or, alternatively, to send a blood sample to the laboratory for a gas analysis after twenty minutes or so.

As this scenario illustrates, our architecture allows an agent to perform a variety of reasoning tasks and, more importantly, to adapt its selection, ordering, and interleaving of reasoning tasks to dynamic perceived and inferred conditions in its environment. Triggering tasks with perceptual events enables the agent to adapt to exogenously produced changes in the world. Triggering tasks with cognitive events enables the agent to follow the intrinsic logical relations among tasks--where the intermediate or final results of one task provide the input to another. Explicit representation of the initial, intermediate, and final results of reasoning tasks allows the agent to interrupt and resume tasks deliberately.

6. Adaptation of Reasoning Methods

In order to function effectively in AIS niches, *an agent must adapt its reasoning methods to the available information.*

Given our input-output definition of tasks, there may be alternative methods for performing particular tasks. For example, an agent might perform a diagnosis task by means of a "model-based" method, in which it instantiates structure/function models of phenomena observed in the monitored system and follows causal links to identify and instantiate hypothesized precursors. Alternatively, an agent might apply a “structured selection” method [Clancey, 1985], in which it abstracts the observed data, performs a heuristic mapping into the hypothesis space, and refines the identified hypothesis back into the problem context. Alternatively, the agent might use a case-based method [Simoudis, 1992], in which it retrieves cases manifesting problems similar to the

observed problem and hypothesizes that the diagnoses associated with those cases may explain the observed problem. Similarly, alternative methods may be applicable to other tasks, such as monitoring, prediction, and planning.

Following our analysis of situation-appropriate control modes (section 4), we offer a similar analysis of situation-appropriate reasoning methods. Again, reasoning situations and methods vary along complementary dimensions: availability versus consumption of resources (e.g., domain knowledge, environmental data, real time, and computation); demand versus provision of performance properties (e.g., interruptability, potentially useful intermediate results); and requirement versus provision of response features (e.g., precision, certainty, quality, and justification).

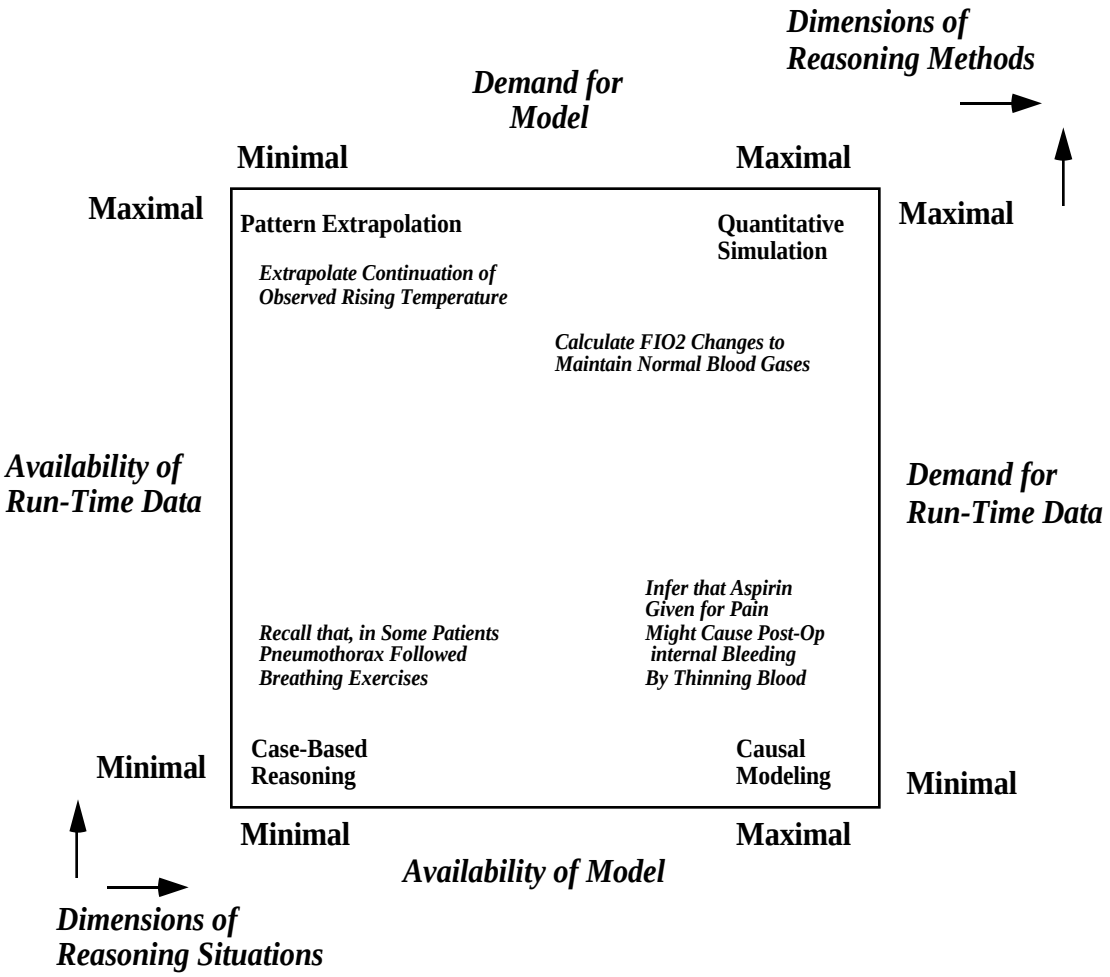

Figure 8. Different Methods for Different Contexts.

Taking a subset of the dimensions defined by these variables for illustration, Figure 8 superimposes two two-dimensional spaces, mapping methods that vary in their consumption of domain knowledge and run-time data onto situations that vary in the availability of these two resources. Methods in particular regions of the superimposed spaces are "appropriate" for situations in corresponding regions, based on two simplifying assumptions: (a) as more knowledge and data are brought to bear, the agent's response improves monotonically on all features; and (b) the agent prefers to expend whatever resources are available in order to produce the highest quality response. A more complete analysis would incorporate information about the actual cost of resources and the utility of particular performance and response features as higher-order dimensions of the superimposed spaces. But even with our simplifying assumptions, the present analysis illustrates the need and potential for agents operating in AIS niches to choose and use appropriate reasoning methods in different reasoning situations.

For illustration, we consider methods representing the four corners of the space in Figure 8, applied to a prediction task.

Applying a quantitative simulation method [Iwasaki and Simon, 1986] to a prediction task, an agent uses observed numeric data to instantiate parameters representing the initial conditions and other important variables in a set of differential equations and calculates the predicted values of the variables of interest after variable time *t*. Quantitative simulation produces precise, reliable, temporally specific quantitative results and explanatory justification in terms of the instantiated equations. Computation time may be high. Other things equal, quantitative simulation is appropriate when the reasoning context includes an appropriate set of differential equations and the run-time data necessary to instantiate the necessary parameters. For example, Guardian should use quantitative simulation to predict whether current values of FIO_2 (amount of oxygen provided by the ventilator on each breath) will maintain normal blood gases for the patient over some time period.

Applying a causal modeling method [Pearl, 1986] to a prediction task, an agent uses qualitative observations to instantiate variables in a causal network with the initial conditions and follows causal links to identify predicted conditions. Causal modeling produces reliable, qualitative results, but no specific temporal information. It provides explanatory justification in terms of the instantiated conditions and causal links in the model. Run-time computation depends on the branching factor and depth of the model.

Other things equal, causal modeling is appropriate when the reasoning context includes an appropriate causal model and when either: (a) the data, knowledge, or resources necessary to instantiate a more precise quantitative model are not available; or (b) the precision of quantitative simulation is not needed. For example, Guardian should use causal modeling to predict that aspirin given to a post-operative patient for pain will also thin the patient's blood (a side-effect) and, therefore, might also cause internal bleeding.

With both quantitative simulation and causal modeling, an agent exploits strong models to make predictions (or perform other tasks) and to explain its conclusions. Quantitative simulation also exploits larger amounts of run-time data to produce more specific, temporally constrained, quantitative predictions. Causal modeling compensates for a lack of relevant run-time data by producing more general, qualitative predictions with less specific temporal properties. Applying intermediate methods between quantitative simulation and causal modeling, the agent uses whatever data are available to quantify its model-based predictions as much as possible. With all methods along this border, however, the agent pays a high cost in run-time computation to reason out its predictions (or other conclusions). In addition, the agent's ability to perform its task with these methods is limited by the availability of appropriate models--which tend to be in short supply in some domains, such as ICU monitoring, but more available in other domains such as device monitoring.

Applying a pattern extrapolation method [Shahar, 1992] to a prediction task, an agent incrementally instantiates time-varying patterns in observed data values and extrapolates their completion to identify predicted conditions. Pattern extrapolation can produce predictions where no models are available, but with high uncertainty and no explanatory justification at all. Run-time computation depends on the number and complexity of known pattern definitions. Pattern extrapolation is appropriate when the reasoning context provides a lot of run-time data for developing and distinguishing among different potential patterns. For example, Guardian could use pattern extrapolation to predict that a monitored patient's rising temperature might continue to rise at its current rate, eventually reaching a dangerous region.

Applying a case-based method [Hammond, 1989; Kolodner, 1984; Riesbeck and Schank, 1989] to a prediction task, an agent retrieves a previous case in which conditions similar to those in the present case occurred and predicts that subsequent conditions in the present case will be similar to those in the previous case. Case-based reasoning can

produce predictions in a broad range of situations, but with high uncertainty and no explanatory justification at all. Computation time depends on the agent's repertoire of cases and indexing mechanism. Case-based reasoning is appropriate when the task context includes a representative sample of cases and the run-time data necessary to index into the "right" prior case. For example, Guardian could use case-based reasoning about previous lung surgery patients to predict that a post-operative patient who is performing his breathing exercises very vigorously might develop a pneumothorax (a hole in the lung) in the area of a lung incision.

With both pattern extrapolation and case-based reasoning, an agent compensates for the absence of good models by using other kinds of knowledge (abstract pattern definitions or previous cases) to make predictions (or perform other reasoning tasks). Pattern extrapolation also exploits the availability of large amounts of run-time data to compensate for the absence of relevant cases. With all methods along this border, the agent pays a minimal cost in run-time computation. Its ability to perform its task is limited primarily by its repertoire of abstract pattern definitions and prior cases, which are readily available in medical domains such as ICU monitoring (often called "clinical experience"), as well as in engineering domains.

Our architecture provides a natural environment for representing, selecting, and applying situation-appropriate reasoning methods. Alternative methods for performing a given task can be represented as different collections of reasoning operations, all of which might be triggered by events signaling a need for that task to be performed. For example, Guardian's decision to give a patient aspirin to relieve pain might trigger a control decision to predict possible side effects, along with the initial reasoning operations underlying quantitative simulation, causal modeling, and case-based reasoning methods for performing prediction tasks. At that point, Guardian is free to apply any, all, or none of the triggered methods. Because reasoning skills are represented explicitly, an agent can determine what run-time data and models are required by a given method in a situation, and which of the required data and models are available in the situation. Continuing the example, Guardian could follow the analysis of Figure 8 (or a similar analysis that incorporates other situational variables) to determine that, for the effects of aspirin, it has a very large number of potentially relevant cases varying on many dimensions, no quantitative models at all, and a simple causal model with a modest demand for run-time data. Under these circumstances, it would be appropriate to use the causal model. By modifying its initial

control decision, so that it now intends to causally predict the side effects of giving aspirin, Guardian insures the selection of causal reasoning operations to perform its task.

In summary, our architecture allows an agent to adapt its reasoning methods to the availability of resources by representing a diverse collection of reasoning methods as sets of event-triggered reasoning operations, explicitly storing method-specific resource requirements, and allowing the agent to construct run-time control plans that reflect its assessment of situation-specific resource availability.

7. Adaptation of Meta-Control Strategy

In order to function effectively in AIS niches, *an agent must adapt its meta-control strategy to dynamic configurations of demands and opportunities for activity.*

An agent's meta-control strategy places global constraints on its allocation of computational and physical (e.g., sensors, effectors) resources among competing activities. As a result, it determines which goals the agent achieves, to what degree, and with what side effects. As illustrated throughout this paper, our architecture permits an agent to adapt its perceptual strategy, control mode, reasoning task, and reasoning method to the requirements of a given activity. However, AIS niches characteristically present demands and opportunities for multiple activities during overlapping time intervals. For example, an ICU patient may manifest several simultaneous problems, varying in criticality. While Guardian is responding to one set of problems, it must continue to monitor other aspects of the patient's condition and, quite possibly, respond to newly occurring problems along the way. In addition, Guardian may perform other tasks not directly concerned with patient monitoring, such as describing a patient's progress during the preceding eight hours to a physician on rounds, explaining its diagnostic reasoning to a medical student, or advising a nurse of anticipated changes in the patient's condition. How should Guardian respond to each new demand or opportunity as it arises? How should its responses to new events impact on its prior commitments to ongoing activities--and vice versa? Thus, in AIS niches, the meta-control problem is: How should an agent allocate its limited computational resources among dynamic configurations of competing and complementary activities so as to achieve a high overall utility of its behavior?

Our architecture provides a natural framework for dynamic adaptation of explicit meta-control strategies for global coordination of its behavior. Working within the basic architectural mechanism, an agent can trigger meta-control operations based on changes to its control plans. It can use some meta-control operations to monitor its activity-specific control decisions, their implications for resource consumption, and its actual progress toward associated objectives--all as they evolve over time. It can use other operations to revise activity-specific control decisions in light of global considerations. For example, Guardian might notice that it has made a series of control decisions to diagnose and treat a series of unanticipated problems. Although each of these decisions may be individually justifiable, together they may exhaust Guardian's computational and perceptual resources and, as a result, compromise its vigilance. Even worse, the division of available resources among the several problems may preclude treating any of them before its deadline. Having made this assessment, Guardian could make a meta-control decision to postpone its diagnosis and treatment of the least important of its pending problems to conserve resources for monitoring and to insure treatment of the most critical problems by deadline.

The architecture also allows an agent to use meta-control decisions prospectively to establish the desired global character of its intended behavior by constraining subsequent meta-level and activity-specific control reasoning. For example, in the episodes illustrated in Figures 9 and 10, Guardian is monitoring a patient who develops two problems, first low blood pressure and then high PIP (peak inspiratory pressure). In both cases, Guardian diagnoses the low blood pressure as resulting from dehydration and treats it by increasing fluid intake. In both cases, it diagnoses the high PIP first as a hypoxia problem, which it treats by increasing the patient's oxygen, and then more specifically as the result of a pneumothorax, which it treats by inserting a chest tube. However, in the two Figures these problems and treatments occur in different meta-level contexts, producing subtle, but significant differences Guardian's behavior and, under some value models, in the overall utility of its behavior.

The two episodes differ in meta-control decision B versus B'. In Figure 9, Guardian has made meta-control decision B, to give its highest priority to urgent problems, its next highest priority to monitoring, and its third highest priority to other problems. As a result, when it observes and decides to respond quickly to the patient's high PIP, Guardian maintains its current monitoring activity, but decides to suspend its activities related to the patient's low blood pressure, a less important problem, until it has resolved the patient's

high PIP. In Figure 10, Guardian has made a different meta-control decision B', to respond to perceived problems immediately. As a result, when it observes and decides to respond quickly to the patient's high PIP, Guardian continues its activities related to the patient's low blood pressure, but reduces its monitoring activities until it has resolved the patient's high PIP. A comparison of corresponding elements of Figures 9 and 10 reveals other consequences of the difference in meta-level strategy. Under strategy B in Figure 9, Guardian completes its diagnosis and treatment of the high PIP problem faster than under strategy B' in Figure 10, but completes its diagnosis and treatment of the low blood pressure problem later. Under strategy B in Figure 9, it remains sensitive to patient data not directly related to its current activities (e.g., heart rate), while under strategy B' in Figure 10, its attention to patient data is depressed by its attention to immediate problems. Depending on Guardian's value model, each of these meta-control strategies could produce a higher overall utility of behavior.

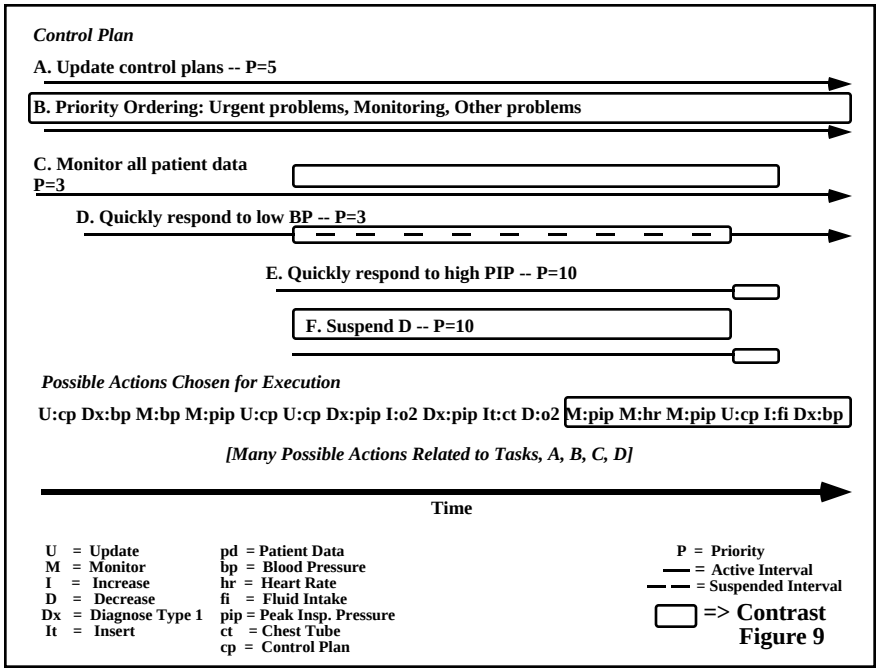

Figure 9. Illustrative Behavioral Effects of Meta-Control Strategy B.

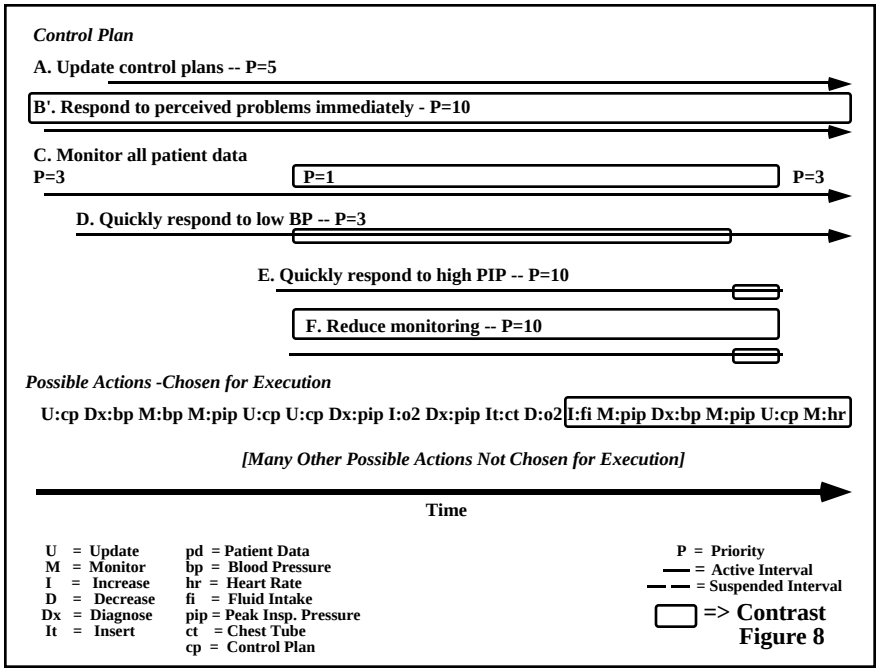

Figure 10. Illustrative Behavioral Effects of Meta-Control Strategy B'.

As these simple examples illustrate, our architecture uses the same underlying mechanism to enable an agent to represent, reason about, and use both activity-specific control plans and meta-control plans. An agent can adapt its meta-control strategy to its dynamic configuration of potential activities by: (a) analyzing control plans representing intended activities to estimate their resource requirements; (b) assessing the availability of required resources in the prospective situation; and (c) making or modifying meta-control decisions that establish appropriate constr aints on the constructions of activity-specific control plans. From the agent's point of view, meta-control plans are no different from other control plans, all of which simply establish local preferences for performing different classes of reasoning operations--which may include different classes of task-level reasoning operations, control operations, and meta-control operations. Similarly, the agent need not treat meta-control planning any differently from its other reasoning activities, all of which occur through the scheduling and execution of event-triggered reasoning operations.

8. Evaluation

8.1 Evaluation Paradigm

How can we evaluate the proposed architecture for adaptive intelligent systems?

Given the complexity of the behavior we aim to support, we emphasize empirical evaluation. Following Simon's observations on computer systems in general, we believe that the problems we are trying to address more closely resemble those of biology or psychology than physics and therefore so should our methods:

> We are never going to have three laws of motion in computer science. ... Now computing systems may or may not be as complicated as living organisms, but that are pretty complicated, and the principal way in which we are going to learn about them is to go into a laboratory and find facts. We do that by building systems and testing them. [Simon, 1991, p. 128]

Moreover, we believe that challenging real-world domains (rather than artificially structured games or toy problems) offer a rich experimental testbed for investigating adaptive intelligent systems, their architectures, and their behavior. In fact, it is difficult to define an artificial task domain that can simulate all of the dimensions of adaptation we observe in real-world AIS niches. Thus, for example, Feigenbaum explains how working on the DENDRAL project [Feigenbaum, et al, 1971] played a critical role in the discovery that production rules could be used for knowledge representation:

> Buchanan succeeded where Waterman failed because Buchanan was immersed in the details of the chemistry, the knowledge representation problem, and the programming of the reasoning process. Waterman was only an onlooker. The immense importance of the experimental method in AI, and more broadly in CS, is that it provides the necessary mental data in sufficient detail to stimulate innovation and discovery. Perhaps it's easier to discover new ideas than to invent them! [Feigenbaum, 1992, p. 197]

Our goal is to develop an architecture that meets a sufficiency criterion, supporting adaptive intelligent systems throughout a large class of AIS niches. Thus, it is less important that any particular aspect of the architecture should embody the optimal approach to achieving any particular form of adaptation than that the architecture should gracefully integrate all of the required forms of adaptation--and that it should demonstratively be able

to produce those behavioral adaptations as required by the operating environment. As Newell remarks on how best to evaluate unified theories of cognition:

> "*Necessary* characteristics are well and good, but they are substantially less than half the story. *Sufficiency* is all-important" [Newell, 1990, p. 158]

Finally, other things being equal--in particular, given that the sufficiency criterion has been met, we prefer architectural parsimony. A compelling architecture should minimize the number of component mechanisms with which it supports the several required forms of adaptation.

8.2 Current Status of Experimental Agents

Our architecture has been implemented in an application-independent form and used to build experimental agents in several of the specific AIS niches in Figure 1.

Guardian is the most substantial of our experimental agents. Guardian demonstration 4 [Hayes-Roth, et al, 1992] monitors on the order of twenty continuously sensed patient data variables and several occasionally sensed variables. Its tasks include monitoring, fault detection, diagnosis, prediction, explanation, and planning. It has relatively fast reasoning methods based on clinical knowledge of commonly occurring problems, their typical symptoms, and their standard treatments. It also has relatively slow, but more comprehensive reasoning methods based on symbolic knowledge of the underlying anatomy, physiology, and pathophysiology. Guardian demonstration 4 has been applied to a small number of simple, but realistic ICU scenarios. As illustrated in examples throughout the paper, this version of Guardian performs rudimentary versions of all of the different kinds of adaptation discussed above. Guardian demonstration 5, which is currently under development, will monitor on the order of 100 variables. It will perform the same set of tasks performed in demonstration 4, but with a more comprehensive set of methods. It will have a much larger medical knowledge base. Most important in the present context, Guardian demonstration 5 will provide a richer environment for evaluating the claimed architectural support for adaptation.

In addition to Guardian, several other experimental monitoring agents have been developed in the architecture. In our laboratory, we have developed experimental agents to monitor power plant equipment [Sipma and Hayes-Roth, 1993] and semi-conductor

manufacturing equipment [Murdock and Hayes-Roth, 1991]. Both of these agents possess symbolic representations of the structure, function, and behavior of the equipment being monitored. They perform model-based process tracking, diagnosis, prediction, and explanation. Each one has been demonstrated on two or three simple, but realistic scenarios. A similar agent has been developed to monitor materials processing [Pardee, et al, 1990]. These applications demonstrate the generality of our agent architecture across diverse domains within the AIS monitoring niche.

More recently, we have begun studying the application of our architecture to a class of niches for adaptive intelligent robots, which we call "AIbots." In a first demonstration [Hayes-Roth, et al, 1993], we developed a simulated robot that plans surveillance destinations and routes, gathers information from the environment, and responds to unanticipated alarms. Despite its simplicity, this agent exhibits several of the kinds of adaptation discussed in this paper. It uses reasoning to select and parameterize perceptual strategies and navigation strategies. It uses dynamic control plans to decide which high-level task to perform (e.g., situation assessment, planning, information gathering) and which method to use for a given task (e.g., a classical planner versus a case-based planner). In a second demonstration, we developed a simulated robot that acts as a general office factotum. It can deliver messages personally or electronically, fetch and deliver objects, and learn unanticipated features of its environment. It accepts asynchronous requests for instances of these message and object goals and generates learning goals for itself. It plans and executes behavior to achieve goals in various sequences and combinations, based on their priorities, deadlines, and interactions with one another. This agent continuously adapts whatever pending goals and plans it has in light of newly perceived information about its environment, new goals, or the unanticipated details of progress on current goals and plans. Finally, in a third demonstration, we have demonstrated the above-described behaviors on an actual Nomad 200 robot [Zhu, 1992] operating in our offices.

The intelligent monitoring niches exemplified by Guardian and the intelligent robotics niches exemplified in our AIbots demand the array of behavioral adaptations characteristic of all adaptive intelligent systems, but they emphasize complementary subsets of these demands. The Guardian niche emphasizes broad and deep domain knowledge and reasoning, important requirements for adaptive selective perception (but no real signal processing), and minimal requirements for action control. The AIbots niche emphasizes signal interpretation as well as selective perception, important requirements

for controlling physical action, and simpler cognitive behaviors. For this reason, we find them to be an interesting combination of niches to which to evaluate our architecture.

8.3 Comparison with Other Architecture

Although our architecture is not the only one that supports adaptation, it is one of a small set of candidate architectures currently in the literature. However, most of these other architectures focus on selected aspects of adaptation, as illustrated by the following examples. Soar [Newell, 1990; Laird, et al, 1987] provides a very general search mechanism that can be applied to a variety of reasoning tasks and a learning mechanism that automatically moves the agent from search to a more reactive control mode based on experience. However, it does not provide a mechanism for perceptual adaptation or a mechanism for deliberately choosing reasoning tasks, reasoning methods, or control modes. The subsumption architecture [Brooks, 1986] embodies a layered control model in which each layer adapts its behavior continuously to relevant perceptual information and imposes constraints on the responsiveness of the layer below itself. However, it has been applied primarily to the perceptual-motor behavior of mobile robots. It has not yet been extended to support reasoning and it does not provide a mechanism that allows an agent to dynamically choose among its own capabilities. CIRCA [Musliner, et al, 1993] and a similar architecture based on the Maruti real-time operating system [Hendler and Agrawala, 1990] offer a two-layer architecture in which unpredictable AI methods are used to set goals and priorities for a real-time scheduler that guarantees to meet hard deadlines (assuming that is feasible) and to use slack resources effectively. This architecture adapts its real-time schedule to available resources and current priorities, but it does not provide other forms of adaptation, particularly within its use of the AI methods.

A caveat: We do not mean to suggest that these architectures are not capable of providing all of the required forms of adaptation, but only that their ability to do so has not yet been demonstrated and is not immediately obvious to us.

8.4 Other Related Work

Several researchers are working on particular forms of adaptation independent of architectural context. Notable examples are: anytime algorithms that trade reasoning time for solution quality [Boddy and Dean, 1989], design-to-time scheduling algorithms for maximizing the use of available resources while meeting deadlines on critical tasks

[Garvey and Lesser, 1993], reactive systems that provide bounded response times for specified events [Rosenschein and Kaelbling, 1986] or flexible adaptation to unanticipated event orderings [Agre and Chapman, 1987; Nilsson, 1989], approximate processing techniques that provide acceptably degraded responses when resources are short [Decker et al, 1990]. We view these approaches as useful capabilities that we would strive to integrate within our architecture.

9. Conclusions

We have characterized a class of AIS niches. They require performance of diverse competing, and complementary tasks. They provide variable, possibly inadequate, resources for performing tasks. They present variably stressful contextual conditions. They impose conflicting performance criteria, which often cannot be satisfied completely. Therefore we have argued that, to function effectively in AIS niches, an agent must be highly adaptive. It must adapt its perceptual strategy to its dynamic cognitive requirements. It must adapt its control mode to uncertainty in its environment and constraints on its actions. It must adapt its reasoning tasks to demands and opportunities presented by its environment. It must adapt its reasoning methods to the available resources. It must adapt its meta-control strategy to its dynamic configuration of potential activities.

We find AIS niches motivating for three reasons. First, AIS niches represent a substantial increment in behavioral requirements compared to niches occupied by typical AI agents. They stress our science. They force us to deal with uncertainty and resource limitations. They force us to balance traditional efforts to design optimal solutions to isolated problems with efforts to design integrated solutions to complex problems. Second, AIS niches appear to represent an achievable objective. They do not overwhelm us with the complexity of all of human behavior, but focus our investigation on a powerful and pervasive property of human behavior: adaptation. Third, AIS niches represent significant real applications (e.g., intelligent monitoring systems, intelligent surveillance systems, intelligent associate systems). Agents that function effectively in these niches would have real practical and social utility.

In this paper, we argue on behalf of a particular agent architecture for AIS niches. However, as noted above, there are other sophisticated agent architectures that could be candidates for AIS niches. The success criteria for an AIS architecture are sufficiency, not necessity, followed by parsimony. It is quite possible that further evaluation of these

candidates will identify several sufficient architectures. In the meantime, we have had a modest success in using our own architecture to build experimental agents in several AIS niches and in demonstrating the required kinds of adaptation. Moreover, we have been able to support the several required dimensions of adaptation parsimoniously, by means of a single architectural concept: *An agent dynamically constructs explicit control plans to guide its choices among situation-triggered possible actions*.

References

P. E. Agre and D. Chapman. Pengi: An implementation of a theory of activity. In Proceedings of the National Conference on Artificial Intelligence, San Mateo, Ca: Morgan Kaufmann, 1987.

J.S. Albus. Brains, Behavior, and Robotics. Peterborough, N.H. : BYTE Books, 1981.

J.S. Albus. Outline for a theory of intelligence. *IEEE Transactions on Systems, Man, and Cybernetics*, 21, 473-509, 1991.

J.S. Albus. System description and design architecture for multiple autonomous undersea vehicles. Gaithersburg, MD: National Institute of Standards and Technology Report 1251, 1988.

D. Ash, G. Gold, A. Seiver, and B. Hayes-Roth. Guaranteeing real-time response with limited resources. *Journal of Artificial Intelligence in Medicine*, 5 (1), 1993, 49-66.

D. Ash and B. Hayes-Roth. A comparison of action-based hierarchies and decision trees for real-time performance. *Proc. of the National Conference on Artificial Intelligence*, Washington, D.C., 1993.

M. Boddy, and T. Dean. Solving time-dependent planning problems. In Proceedings of the International Joint Conference on Artificial Intelligence, 1989.

R.A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*. RA-2 (1), 14-23, 1986.

W. Clancey. Heuristic classification. *Artificial Intelligence*, 27, 289-350, 1985.

R. A. Brooks. Intelligence Without Reason. Proceedings of the International Joint Conference on Artificial Intelligence, San Mateo, Ca: Morgan Kaufmann, 569-595, 1991.

K. Decker, V. Lesser, and R. Whitehair. Extending a blackboard architecture for approximate processing. *Journal of Real-Time Systems*, 2, 47-70, 1990.

R. Engelmore and T. Morgan. (Eds), Blackboard Systems, Menlo Park, CA.: Addison-Wesley Publishing Company, 1988.

L. Erman, F. Hayes-Roth, V. Lesser, and R. Reddy. The Hearsay-II speech-understanding system: Integrating knowledge to resolve uncertainty. *Computing Surveys*, 12, 213-253, 1980.

E.A.Feigenbaum. A personal view of expert systems: Looking back and looking ahead. *Expert Systems with Applications*, 5, 193-201, 1992.

E.A. Feigenbaum, B.G. Buchanan, and J. Lederberg. On generality and problem solving: A case study using the DENDRAL program. In B. Meltzer and D. Michie (Eds.) New York: American Elsevier, Machine Intelligence 6, 165-190, 1971.

R. E. Fikes and N. J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2, 198-208, 1971.

A. Garvey, and V. Lesser. Design-to-time real-time scheduling. To appear in *IEEE Transactions on Systems, Man, and Cybernetics*, 23, 1993.

A. Georgeff and A. Lansky. Reactive reasoning and planning. Proceedings of the National Conference on Artificial Intelligence, San Mateo, Ca: Morgan Kaufmann, 1987.

K. Hammond. Case-Based Planning : Viewing Planning as a Memory Task. Boston : Academic Press, 1989.

B. Hayes-Roth. A Blackboard Architecture for Control. *Artificial Intelligence*, 26, 251-321, 1985.

B. Hayes-Roth. Architectural foundations for real-time performance in intelligent agents. *Real-Time Systems: The International Journal of Time-Critical Computing Systems,* 2, 99-125, 1990.

B. Hayes-Roth, On building integrated cognitive agents: A review of Newell's Unified Theories of Cognition. *Artificial Intelligence*, 59, 213-220, 1993a.

B. Hayes-Roth. Opportunistic control of action. *IEEE Transactions on Systems, Man, and Cybernetics*, in press, 1993b.

B. Hayes-Roth, P. Lalanda, P. Morignot, M. Balabanovic, and K. Pfleger. Plans and behavior in intelligent agents, submitted to AI Journal, 1993.

B. Hayes-Roth, R. Washington, D. Ash, A. Collinot, A. Vina, and A. Seiver. Guardian: A prototype intensive care monitoring agent. *Journal of Artificial Intelligence and Medicine*, 1992.

B. Hayes-Roth, R. Washington, R. Hewett, M. Hewett, and A. Seiver. Intelligent Real-Time Monitoring and Control. Proceedings of the International Joint Conference on Artificial Intelligence, San Mateo, Ca: Morgan Kaufmann, 1989.

J. Hendler, and A. Agrawala. Mission critical planning: AI on the MARUTI real-time operating system. In Proceedings of the Workshop on Innovative Approaches to Planning, Scheduling, and Control, 77-84, November, 1990.

T. Iwasaki and H.A. Simon. Causality in device behavior. *Artificial Intelligence*, 29, 3-32, 1986.

V. Jagannathan, R. Dodhiawala, and L. Baum. (Eds.). Blackboard Architectures and Applications. Boston, MA.: Academic Press, Inc., 1989.

M.V. Johnson and B. Hayes-Roth. Integrating diverse reasoning methods in the BB1 blackboard control architecture. Proceedings of the International Joint Conference on Artificial Intelligence, San Mateo, Ca: Morgan Kaufmann, 1987.

J.L. Kolodner. Retrieval and Organizational Strategies in Conceptual Memory : A Computer Model. Hillsdale, N.J. : L. Erlbaum Associates, 1984.

J. Laird, A. Newell, and P.S. Rosenbloom. Soar: An architecture for general intelligence. *Artificial Intelligence*, 33, 1-64, 1987.

D. Lenat, and E. A. Feigenbaum. On the thresholds of knowledge. *Artificial Intelligence*, 47, 1991, 185-250.

T. Mitchell, J. Allen, P. Chalasani, J. Cheng, O. Etzioni, M. Ringuette, and J. Schlimmer. Theo: A framework for self-improving systems. In K. VanLehn (ed.), Architectures for Intelligence. Hillsdale, NJ: Lawrence Erlbaum Associates, Inc., 1991.

J. L. Murdock and B. Hayes-Roth. Intelligent Monitoring and Control of Semiconductor Manufacturing. *IEEE Expert*, 6, 19-31, 1991.

D.J. Musliner, E.H. Durfee, and K.G. Shin. CIRCA: A cooperative intelligent real-time control architecture. To appear in *IEEE Transactions on Systems, Man, and Cybernetics*, 23, 1993.

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

N. Nilsson. Action networks. Stanford, CA: Technical Report, 1989.

W. Pardee, M. Shaff, and B. Hayes-Roth. Intelligent control of complex materials processes. *Journal of Artificial Intelligence in Engineering, Design, Automation, and Manufacturing*, 4, 55-65, 1990.

J. Pearl. Fusion, propagation, and structuring in belief networks. *Artificial Intelligence*, 29, 241-288, 1986.

C.K. Riesbeck and R.C. Schank. Inside Case-Based Reasoning. Hillsdale, N.J. : L. Erlbaum, 1989.

S. J. Rosenschein and L. P. Kaelbling. The synthesis of digital machines with provable epistemic properties. Proceedings of the Conference on Theoretical Aspects of Reasoning about Knowledge, Morgan Kaufmann, pp. 83-98, 1986.

W.J. Rugh. Analytical framework for gain scheduling. Proceedings of the American Control Conference, 2, 1990.

E. D. Sacerdoti. The non-linear nature of plans. Proceedings of the International Joint Conference on Artificial Intelligence, San Mateo, Ca: Morgan Kaufmann, 1975.

M. Schoppers. Universal plans for reactive robots in unpredictable environments. Proceedings of the International Joint Conference on Artificial Intelligence, San Mateo, Ca: Morgan Kaufmann, 1987.

Y. Shahar. A temporal abstraction mechanism for patient monitoring. Proceedings of SCAMC. 121-127, 1991.

H.A. Simon. Artificial Intelligence: Where has it been and where is it going? IEEE Transactions on Knowledge and Data Engineering, Vol. 3, No. 2, 1991, 128-136.

H. Sipma and B. Hayes-Roth. Model-based monitoring of dynamical systems. Stanford, CA: Technical Report, in preparation, 1992.

R. Washington and B. Hayes-Roth. Managing input data in real-time AI systems. Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, San Mateo, Ca: Morgan Kaufmann, 1989.

D. E. Wilkins. Domain-independent planning: Representation and plan generation. *Artificial Intelligence*, 22, 1984.

D. Zhu. Nomadic host software development environment (Unix Version 1.1). Mountain View, Ca.: Nomadic Technologies, Inc., 1992.