Architectural Foundations for Real-Time Performance in Intelligent Agents

Barbara Hayes-Roth

Knowledge Systems Laboratory
Computer Science Department
Stanford University

Real-Time Systems, May, 1990

This research was supported by DARPA contract N00039-83-C-0136, NIH contract 5P41-RR-00785, EPRI contract RP2614-48, and AFOSR contract F49620-89-C-0103DEF, and by gifts from Rockwell International, Inc. and FMC Corporation, Inc. The Guardian system is being developed in collaboration with Adam Seiver, Rich Washington, David Ash, Rattikorn Hewett, Anne Collinot, Luc Boureau, Angel Vina, Ida Sim, and Michael Falk. The paper's treatment of real-time requirements reflects discussions with colleagues involved in the AFOSR Program on Intelligent Real Time Problem Solving Systems--especially Stan Rosenschein, Lee Erman, and Yoav Shoham. The paper also benefited from constructive criticism by several anonymous reviewers. Thanks to Ed Feigenbaum for sponsoring the work at the Knowledge Systems Laboratory.

Abstract

Intelligent agents perform multiple concurrent tasks requiring both knowledge-based reasoning and interaction with dynamic entities in the environment, under real-time constraints. Because an agent's opportunities to perceive, reason about, and act upon the environment typically exceed its computational resources, it must determine which operations to perform and when to perform them so as to achieve its most important objectives in a timely manner. Accordingly, we view the problem of real-time performance as a problem in intelligent real-time control. We propose and define several important control requirements and present an agent architecture that is designed to address those requirements. The proposed architecture is a blackboard architecture, whose key features include: distribution of perception, action, and cognition among parallel processes, limited-capacity I/O buffers with best-first retrieval and worst-first overflow, dynamic control planning, dynamic focus of attention, and a satisficing execution cycle. Together, these features allow an intelligent agent to trade quality for speed of response under dynamic goals, resource limitations, and performance constraints. We illustrate application of the proposed architecture in the Guardian system for surgical intensive care monitoring and contrast it with alternative agent architectures.

1. Real-Time Performance in Intelligent Agents

Imagine an "errand robot" driving an automobile on its way to some destination. Noticing a yellow traffic light at the next intersection in its path, the robot infers from its current speed, distance to the light, and conservative traffic-light policy that it should stop. The robot immediately releases the accelerator and, after a few seconds, applies the brake to bring its vehicle to a gradual stop just before entering the intersection. The robot's behavior is satisfactory not simply because it produces the correct result, but because it does so at the right time. If the robot stopped very much before or after reaching the intersection, its behavior would be unsatisfactory and potentially catastrophic.

The errand robot illustrates a class of computer systems, which we call "intelligent agents," whose tasks require both knowledge-based reasoning and interaction with dynamic entities in the environment--such as human beings, physical processes, other computer systems, or complex configurations of such entities. Tasks requiring an intelligent agent occur in diverse domains, such as power plant monitoring [53], process control [10, 41], experiment monitoring [40], student tutoring [38], aircraft pilot advising [47], and intensive care patient monitoring [15, 27].

To perform such tasks, an agent must possess capabilities for: *perception*--acquiring and interpreting sensed data to obtain knowledge of external entities; *cognition*--knowledge-based reasoning to assess situations, solve problems, and determine actions; and *action*--actuating effectors to execute intended actions and influence external entities. For example, the errand robot perceives signals from which it infers that the traffic light is yellow. It reasons with this perception, its traffic light policies, and other perceptions and knowledge to determine that gradually coming to a stop at the intersection is the desired result and that releasing the accelerator and applying the brake are the appropriate actions. It performs those actions in the appropriate temporal organization, thereby achieving the intended result.

Because external entities have their own temporal dynamics, interacting with them imposes aperiodic hard and soft real-time constraints on the agent's behavior. Following [3] we use the term "aperiodic" to describe tasks having irregular arrival times. Following [16, 50] we use the terms "hard" and "soft" to distinguish between constraints whose violation precludes a successful result versus those whose violation merely degrades the utility of the result. For example, a vehicle

that happens to stop in front of the errand robot is an aperiodic event with a hard deadline. The robot must stop in time to avoid colliding with the other vehicle. When that is not possible, the robot should consider alternative actions, such as maneuvering around the stopped vehicle.

In a complex environment, an agent's opportunities for perception, action, and cognition typically exceed its computational resources. For example, in the scenario above, the errand robot has opportunities to perceive the physical features and occupants of other automobiles on the road and the buildings and landscape along the sides of the road. It might reason about any of these perceptions or other facts in its knowledge base. It might perform a variety of actions more or less related to driving its automobile. Fortunately, the robot largely ignores most of these opportunities to focus on matters related to the traffic light. Otherwise, it might fail to perform the necessary perception, reasoning, and actions in time to stop its automobile at the right time. On the other hand, the errand robot cannot totally ignore incidental information without risking the consequences of rare catastrophic events. For example, the robot should notice a child running into its path. In some cases, the robot might benefit from noticing information that is not immediately useful. For example, it might notice a sign posting business hours on a shop window and use that information when planning a subsequent day's errands.

Because an intelligent agent is almost always in a state of perceptual, cognitive, and action overload, it generally cannot perform all potential operations in a timely fashion. While faster hardware or software optimization may solve this problem for selected application systems, they will not solve the general problem of limited resources or obviate its concomitant resource-allocation task [49]. For an agent of any speed, we can define tasks whose computational requirements exceed its resources. Moreover, we seek more from an intelligent agent than satisfactory performance of a predetermined task for which it has been optimized. Rather, we seek satisfactory performance of a range of tasks varying in required functionality and available knowledge as well as real-time constraints. And we seek adaptation to unanticipated conditions and requirements. For example, the errand robot should be able to respond appropriately to traffic signals and other usual and unusual events in a broad range of driving situations. It should drive competently on freeways as well as on surface streets. If it unexpectedly finds itself on surface streets where others are driving at freeway speeds (or, more likely, vice versa), it should adapt its own behavior accordingly. The agent might have other sorts of skills, such as planning its own errands under high-level goals and constraints or learning new routes from experience taking

necessary detours. Other things being equal, the broader the range of tasks an agent can handle and the wider the range of circumstances to which it can adapt, the more intelligent it is.

For these reasons, we view real-time performance as a problem in intelligent control. An agent must use knowledge of its goals, constraints, resources, and environment to determine which of its many potential operations to perform at each opportunity. For example, the errand robot might decide to give high priority to perceiving and reasoning about traffic lights so that it can always stop in time for yellow or red lights. When the operations required to achieve an agent's current goals under its specified constraints exceed its computational resources, it may have to modify them as well. For example, if the errand robot finds itself unexpectedly late to an important destination, it might decide to relax its conservative traffic-light policy and drive through selected yellow lights. Because it is situated in a dynamic environment and faces a continuing stream of events, an agent must make a continuing series of control decisions so as to meet demands and exploit opportunities for action as they occur. For example, if the errand robot is making a planned gradual stop at a traffic light and a child runs into its path, the robot should perceive the child and stop immediately. In general, an agent should use intelligent control to produce the best results it can under real-time constraints and other resource (e.g., information, knowledge) constraints.

Our conception of real-time performance in intelligent agents is qualitatively different from conceptions embodied in other sorts of computer systems [3, 7, 16]. In particular, we do not view real-time performance as a provable, guaranteed, universal property of the agent. Nor do we seek real-time performance through effective engineering of the agent for narrowly specified task environments. We feel that these constructs are surely premature and probably unrealistic for the versatile and highly adaptive agents we envision. Rather, we view real-time performance as one of an agent's several objectives, which it will achieve to a greater or lesser degree as the result of interactions between the environment it encounters, the resources available to it, and the decisions it makes. In many cases, the agent will produce timely results for a task only at the expense of quality of result or by compromising the quality or timeliness of its performance of other tasks. Ironically, as the agent's competence expands, so will its need to make such compromises.

From this perspective, real-time performance in intelligent agents depends on an underlying architecture that enables agents to make and apply effective control decisions. Sections 2, 3, and 4 define requirements

for real-time control and the architecture we have designed to address the requirements. Section 5 illustrates application of the architecture in the Guardian system for intensive care monitoring. Section 6 discusses alternative approaches to real-time performance in intelligent agents. Section 7 discusses the architecture's emphasis on satisficing methods--dynamically balancing quality and speed of performance.

2. Requirements for Real-Time Control in Intelligent Agents

In section 2.1, we introduce a neutral framework in which to discuss agents and their environments. In sections 2.2 and 2.3, we operationalize environmental characteristics and agent requirements in the terms of the framework and show how the former motivate the latter (see also [12, 30, 31, 34, 42, 46, 49]).

2.1 A Framework

Following [42], we model an intelligent agent as a dynamic embedded system. The overall system is modeled as a time series of states in which instants of time are mapped to a state space of values representing the variables of interest. A change in the value of a state variable is an *event*, *e*. The system's behavior is described with *measurements* defined as functions on state values. Because the system is dynamic, we describe properties of both individual states and time series of states. *Descriptive measurements* represent objective properties, for example the *importance* of an event *e1* or the *latency* of event *e2* following the occurrence of *e1*. *Utility measurements* represent valuational properties, for example the satisfaction of particular constraints on the latency of *e2*.

We partition the overall system into components representing the intelligent agent, *I*, and the environment, *E*. Each component has its own dynamic state, which varies as a function of information passed among its internal components, as well as information received from the other component. We further partition the agent, *I*, into components for perception, *P*, cognition, *C*, and action, *A*, which similarly manifest events generated internally or by other components. To describe interactions between components, we refer to pairs of *trigger* and *response* events, where both events occur in one component but presumably are mediated by interaction with another component. For example, a trigger-response pair in *E* may be mediated by events in *I*. In some cases, we refer simply to a mediated event, for example an *I*-mediated event in *E.*

2.2 The Environment of an Intelligent Agent

In the terms of our framework, intrinsic characteristics of an agent's environment may be defined as measurements on events in *E*, while characteristics of the relationship between an agent and its environment may be defined as measurements on events in *E* and *I*. Where definitions of environmental characteristics require domain-specific assumptions, we simply indicate the forms such definitions would take.

Data Glut. It is not feasible for the agent to process all potentially interesting events in the environment. That is, the average rate of events in *E* very much exceeds the maximum rate of *E*-mediated events in *I.*

Data Distribution. Important environmental conditions may correspond to configurations of events on different state variables and over variable time intervals. This can be described as particular kinds of many-to-one mappings of events in *E* to events in *I.*

Diversity of Events. Environmental conditions vary in importance. This can be expressed as the variability of values on an "importance" attribute of events in *E*.

Real-Time Constraints. The values of events vary, in part, as a function of when they occur. This can be expressed in terms of utility measurements that incorporate the absolute or relative times of occurrence of events in *E*.

Multiplicity of Conditions. It is not feasible to enumerate all interesting conditions the agent will encounter, that is, the set of *E*-mediated events in *I* that produce criterial values on some measurement.

Predictability. The environment is orderly enough to permit probabilistic prediction of some future events. This can be expressed as descriptive measurements on particular patterns of events in *E*.

Potential Interactions. Globally coordinated courses of action are sometimes superior to sequences of locally determined actions. This can be expressed as utility measurements on particular patterns of *I*-mediated events in *E*.

Underlying Model. Some knowledge of the environment is available. This can be expressed as descriptive measurements on the correspondence between patterns of state values or events in *E* and *I*.

Diverse Demands. Multiple interacting demands for interaction with the environment include: interpretation, diagnosis, prediction, reaction, planning, and explanation. These can be expressed as utility measurements on particular types of *I*-mediated events in *E*.

Variable Stress. The environment varies in its stressfulness over time. This can be operationalized as descriptive measures involving particular environmental variables, for example, the rate of important events or the number and types of different demands for interaction.

2.3 Agent Requirements

We define the primary objective of an intelligent agent very generally:

> *To maintain the value of its own behavior within an acceptable range over time*.

For a given agent in a given environment, we could formalize this requirement in terms of some utility measurement on *I*-mediated events in *E* and also on events in *I* if we wish to constrain the agent's management of its own resources. Although we could use this utility measurement to evaluate the agent's behavior in the given context, it would provide little guidance toward the design of effective agents.

We need a more specific set of requirements to constrain the space of possible agent architectures. Below, we define several requirements that we hypothesize will allow an agent to meet its primary objective in the kinds of environments characterized above. (This is a sufficiency hypothesis, not a neccessity hypothesis. There may well be other requirements whose satisfaction would enable an agent to meet its primary objective.) In the terms of our framework, these requirements refer primarily to events in *E* and to interactions between *I* and *E*. In some cases, we extrapolate requirements to interactions among *I*'s components, *P*, *C*, and *A*, in an effort to support satisfaction of the higher-level requirement. Again, where requirements involve domain-specific assumptions, we simply indicate the forms their definitions would take.

Communications. Given the need for *I* to interact with *E*, there must be appropriate communications involving *I*''s components, with information passing at least: from *E* to *P*, from *P* to *C*, from *C* to *A*, and from *A* to *E*.

Asynchrony. Given data glut and real-time constraints, the agent must function asynchronously with respect to the environment. That is, the

rates of events in *I* and *E* must be independent and the rates of events in *P*, *C*, and *A* must be independent.

Selectivity. Given data glut and the diversity of events in the environment, the agent must determine whether and how to perceive, reason about, and act upon different environmental events. Other things being equal, the conditional probability of an *I*-mediated response event in *E*, given its trigger event, should be an increasing function of the trigger event's importance. The same holds for events in *P, C*, and *A*.

Recency. An agent's sensory information is perishable, the utility of its reasoning degrades with time, and the efficacy of its actions depends upon synchronization with fleeting external events. Therefore, recency is one important selectivity criterion. This can be expressed as a sharply decreasing conditional probability of an *I*-mediated response event in *E*, given its trigger event, over time. The same holds for events in *P*, *C*, and *A*.

Coherence. The agent should produce a globally coordinated course of action when that is preferable to a sequence of locally determined actions. That is, we impose utility measurements on certain patterns of *I*-mediated response events in *E*, as well as on mediated response events in *P*, *C*, and *A*. Other things being equal, we require a low conditional probability of mediated response events, given associated trigger events, when those response events would not fit an ongoing pattern.

Flexibility. Conversely, the agent must react to important unexpected events in a dynamic environment. Other things being equal, we require a high conditional probability for an *I*-mediated response event in *E*, even if it does not fit an ongoing pattern, given a very important trigger event. The same holds for anomalous response events in *P*, *C*, and *A*.

Responsiveness. Other things being equal, the more urgent a situation is, the more quickly the agent should perceive relevant information, perform necessary reasoning, and execute appropriate actions. That is, the latency of an *I*-mediated response event in *E*, following its trigger event, should decrease as the urgency of the trigger event increases. Similar constraints apply to response events in *P*, *C*, and *A*.

Timeliness. Given its dynamic environment, the agent must meet various hard and soft real-time constraints on the utility of its behavior. These may be expressed as utility measurements involving latencies within *I*-mediated pairs of trigger and response events in *E*. Similar measurements could be applied to events in *P, C*, and *A.*

Robustness. An agent must adapt to resource-stressing situations by gracefully degrading the utility of its behavior. As environmental stress increases (for example, as event rates increase or required latencies (deadlines) for trigger-response pairs decrease), the global utility of the agent's behavior (for example the rate of *I*-mediated response events in *E*, weighted by importance) should decrease gradually, rather than precipitously. The same holds for interactions among *P*, *C*, and *A.*

Scalability. In the terms of our framework, the agent's satisfaction of the requirements above (but perhaps not its absolute level of performance on any one task) should be invariant over increases in problem size.

Development. An agent must exploit new knowledge to improve the utility of its behavior. As the amount of relevant knowledge in *I* increases, we should observe improvement in the agent's satisfaction of some of the above requirements and, therefore, in the global utility of its behavior.

3. Proposed Agent Architecture

The proposed agent architecture is designed to address the above requirements. Except where noted, the architecture is implemented as described.

3.1 Top-Level Organization

Following the terminology of section 2, we propose an architecture for the agent, I, comprising subsystems for perception, cognition, and action--*P*, *C*, and *A.* The architecture (see Figure 1) partitions each subsystem into smaller components and permits multiple subsystems for different application-specific perception/action modalities. A *communications interface (CI)* routes data among the I/O buffers of different subsystems. Subsystems function and interact as follows. Signals from the environment enter sensory buffers in *perception subsystems*, which selectively interpret and filter the signals under attentional parameters determined by the cognitive subsystem and place the resulting perceptions in their output buffers. The CI relays these perceptions to input buffers in action subsystems, where they directly drive action execution, or to input buffers in the cognitive subsystem, where they compete with other perceptions and internally generated events for cognitive processing. The *cognitive subsystem* retrieves perceptions from its input buffers for incorporation in its knowledge base, performs all knowledge-based reasoning, and places decisions regarding attentional parameters or intended actions in its output buffers.

The CI relays these decisions to the input buffers of appropriate perception/action subsystems. Each *action subsystem* retrieves action descriptions from its input buffers and controls their execution on particular effectors under performance parameters determined by the cognitive subsystem. Executed actions affect entities in the environment. Subsystems operate in parallel. They do not communicate directly or otherwise interfere with one another. They influence one another only indirectly, by placing information in their own output buffers, from which it is transferred to the input buffers of appropriate other subsystems by the CI. Thus, the architecture limits potential interference to simultaneous efforts to access a subsystem's I/O buffers by the CI and the subsystem itself. Our experiments [32] suggest that, in practice, the architecture provides constant communication latencies among perception, cognition, and action subsystems (the absolute latency being determined by processor speed, network speed, and program optimization) over a wide range of activity levels within each subsystem. Conversely, it provides constant operation latencies within subsystems over a range of levels of communication activities.

The architecture is designed to support graduated reactions. Very fast *peripheral reactions* occur within a perception or action subsystem, producing input-driven attentional shifts or feedback control of action execution. Fast *reflex reactions* occur across perception-action arcs, with information from perception subsystems directly driving the behavior of action subsystems. Slower *cognitive reactions* involve all three kinds of subsystems, with cognition mediating the performance of actions in response to perceived information. Absolute response latencies at each level depend on the architecture's implementation and its instantiation in a particular agent. In our current work, cognitive reactions fall along a latency spectrum, ranging from "immediate" reactions, with latencies under one minute, to "delayed" reactions, with latencies on the order of several minutes or longer. As discussed below, the agent can control the latencies of its cognitive operations in several ways. Although we have not implemented peripheral or reflex reactions, our current implementation could provide latencies on the order of a few seconds.

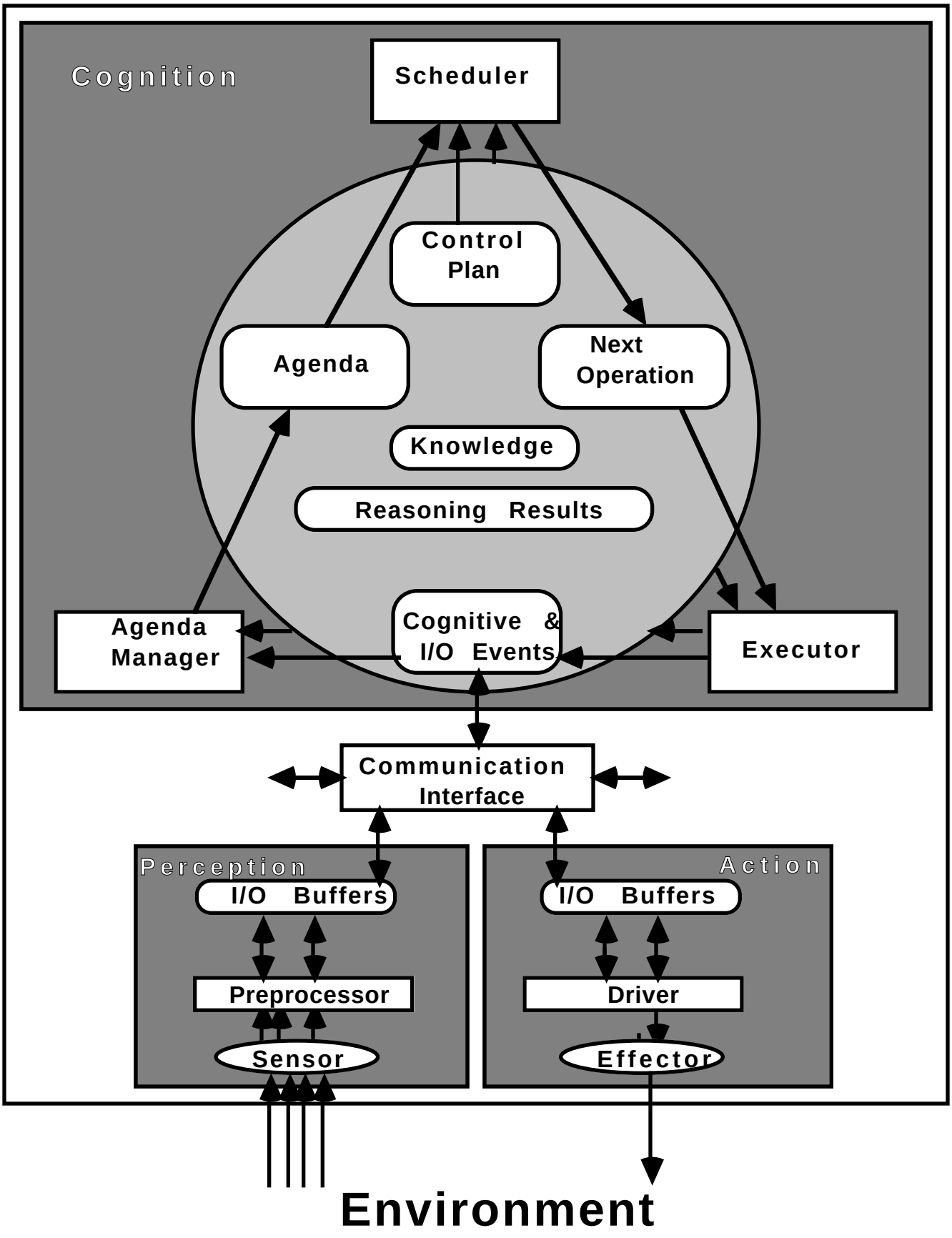

Cognition
Scheduler
Control Plan
Agenda
Next Operation
Knowledge
Reasoning Results
Agenda Manager
Cognitive & I/O Events
Executor
Communication Interface
Perception
I/O Buffers
Preprocessor
Sensor
Action
I/O Buffers
Driver
Effector
Environment

Figure 1. Overview of the Proposed Agent Architecture. Curved boxes represent data structures. Rectangular boxes represent parallel processes. Arrows show information flow among data structures and processes. The diagram is hierarchical. Thus, cognition is a process comprising several component processes, data structures, and information flows among them. Arrows that involve compound data structures (e.g., the circle representing the cognitive system's global memory) signify information flow involving all component data structures (e.g., knowledge, reasoning results, control plan).

As mentioned above, *P*, *C*, and *A* subsystems communicate via *I/O buffers*, with the CI, running on a separate process, routing information among them. All I/O buffers have *limited capacity*, with *best-first retrieval* and *worst-first overflow.* Capacity is an architectural parameter that can be defined differently for different agents or for different buffers within an agent. In our current implementation, it is defined in terms of number of items. Best-first and worst-first criteria are defined in terms of four orthogonal attributes of each buffer item: the item's *relevance* to the agent's current reasoning activities; the item's *importance* with respect to the agent's objectives; the *recency* of the item's appearance in the buffer; and the *urgency* of processing the item in order to have the intended effect. Other things equal, buffer items that score higher against these criteria are retrieved earlier, while those that score lower overflow earlier. These attributes are determined and dynamically modified by the agent's reasoning, as discussed below.

3.2 Perception Subsystems

Perception subsystems [6, 54], comprising *sensors* and *preprocessors*, acquire information about the dynamic environment as a basis for cognition and action. Each *sensor* acquires signals of a characteristic type, transduces them into an appropriate internal representation, and holds the results in a limited-capacity buffer for retrieval by its associated preprocessor. Each *preprocessor* abstracts, annotates, and filters sensed information and places the results in its output buffer for relay by the CI to the input buffer of the cognitive subsystem or an action subsystem. Abstraction involves interpreting and often compressing sensed data according to current *abstraction forms*. These forms might specify transformations on individual data values (for example, assignment to a value category), on sequences of values (e.g., running

averages, trends, modal values), or on patterns of values across multiple variables (for example, co-occurence or temporal succession of related values on different variables). Filtering involves restricting the communication of abstraction results to values that meet current *filtering criteria*. These criteria may be specified for example, as criterial values on particular variables, criterial value changes on particular variables, or deadlines. In our current work, we use a combination of criterial value changes (send a new value when it differs from the last value sent by at least *p*%) and deadlines (send a new value when at least *m* seconds have passed since the last value was sent). This allows the agent to bound the variability on data sensed between sent values in the context of some minimum rate. Annotation involves marking and prioritizing abstraction results according to current *standards of relevance, importance, and urgency* (defined above).

All preprocessing parameters--abstraction forms, filtering criteria, and annotation standards--are dynamic. They can change in two ways. The preprocessor can have peripheral reactions, redirecting its own focus of attention in response to sensed data values. For example, a preprocessor might react to a sudden increase in the variability of any sensed data variable by changing its abstraction forms to a finer granularity and weakening its filtering criteria for that variable. The preprocessor also can change its parameters in response to focus of attention instructions from the cognitive subsystem. For example, the cognitive subsystem might instruct the preprocessor to use parameters appropriate for the current reasoning task. Focus of attention is discussed in more detail below.

3.3 Action Subsystems

Action subsystems, comprising *drivers* and *effectors*, retrieve action descriptions from their input buffers, control action execution on effectors, and return feedback to the cognitive subsystem. Each *driver* monitors its input buffer, retrieves intended actions, translates them into executable programs of effector commands, and monitors the execution of those programs by sending successive commands to the appropriate effector at the appropriate times. Each driver also should take into account importance, urgency, and other constraints on performance, but we have not yet implemented these capabilities. For example, a driver might give priority to important and urgent actions over competitors, translate intended actions into different executable forms given their urgency and resource constraints, and if necessary accelerate execution of urgent actions. The driver also should send feedback to the cognitive

system regarding the success or failure of action execution. Each *effector* immediately executes commands in its input buffer.

3.4 Cognition Subsystem

The cognition subsystem holds all of an agent's knowledge and performs all of its reasoning. It asynchronously incorporates perceived information, retrieved from its input buffers, into its knowledge base. It performs a variety of knowledge-based reasoning tasks, which vary across different task environments, but typically would include: interpretation of perceived information; detection and diagnosis of exceptional events; reaction to important events; prediction of future events; modeling dynamic external systems; planning longer-term courses of action; explaining its observations, inferences, and plans; explaining its reasoning; learning to improve its behavior based on experience and to adapt its behavior to changing environmental conditions. In addition, it reasons about global control of multiple tasks both to coordinate their interactions and to insure timely achievement of the most important objectives given the available resources. The cognitive subsystem initiates actions by placing descriptions of them in its output buffers.

As shown in Figure 1, the cognition subsystem extends the "dynamic control architecture" [25], previously implemented as the BB1 system. All reasoning operations occur in the context of a *global memory*, which represents all information--*knowledge* and *reasoning results*--known to the agent, in a conceptual graph formalism [48].

One important kind of knowledge is a repertoire of *reasoning operations* and associated *strategies*, which can be instantiated to perform particular *tasks* (for example, diagnosis, prediction, explanation, or planning) by particular *methods*. For example, an agent might have knowledge of the operations involved in *associative diagnosis*, along with strategies for selecting and applying those operations. It might have similar knowledge of *model-based diagnosis*. It might also have the "meta-knowledge" that model-based diagnosis requires less data, but more knowledge and computation time, and produces more comprehensive and more explanatory results than associative diagnosis.

As discussed above, the global memory contains *input buffers* for perceptions sent by perceptual subsystems and *output buffers* for intended actions to be sent to action drivers and control parameters to be sent to perceptual preprocessors or action drivers. I/O buffers have limited capacity, with best-first retrieval and worst-first overflow.

The global memory also contains information regarding the agent's cognitive behavior (discussed below). A *cognitive buffer* holds cognitive events produced by reasoning operations. An *agenda* holds executable reasoning operations suggested by perceptual or cognitive events. A *control plan* represents the agent's intended course of behavior as determined by reasoning operations. The *next operation* is the reasoning operation that the agent will execute next. Like I/O buffers, the cognitive buffer and the agenda have limited capacity, with best-first retrieval and worst-first overflow. Although our current implementation does not limit the size of the control plan, we intend to impose some sort of limitation.

Finally, the global memory contains the results of reasoning operations: observations, inferences, predictions, and plans. These results are organized in an interval-based time-line representation, with conceptual links to one another and to other knowledge. For example, an agent might record that a diagnosis believed during interval *i2 explains* an observation that persisted during interval *i1* and that the explanatory relationship between the observation and its diagnosis *instantiates* a known causal relationship within systems of the type under observation.

The cognitive subsystem performs reasoning operations that are suggested by and produce changes to information in the global memory. Its *satisficing cycle* comprises three component processes:

1. The *agenda manager* uses recent perceptual or cognitive events to identify and rate executable reasoning operations, which it records on the agenda. Identification of an executable reasoning operation involves determining that a perceptual or cognitive event satisfies the trigger requirements of a particular type of operation and that other contextual information satisfies its preconditions. On a given cycle, the agenda manager may identify several executable reasoning operations relevant to each of several tasks. Rating an executable operation involves evaluating its importance and urgency against the current control plan, which may include strategic decisions related to different tasks.

2. The *scheduler* determines which of the identified executable operations to execute and when to execute them, based on their ratings, and records each successive one as the next operation.

3. The *executor* executes each next operation as it is recorded. It instantiates the program defined for the chosen operation type, binding program variables to triggering events other contextual information. It then executes the instantiated program, producing associated changes in the global memory. These changes might represent a new inference or

conclusion for a new or ongoing reasoning task. They might record new perceptual filters or intended actions in output buffers. They might change the control plan itself by initiating or terminating new tasks or by extending or modifying control decisions for an ongoing task. As discussed below, changes to the control plan change the criteria used to trigger, rate, and schedule operations for execution, from that time forward.

Because control in the cognitive system determines the utility--quality and timeliness--of the agent's perception, reasoning, and action, it is fundamental to the proposed agent architecture, especially to its support for real-time performance. The following sections examine three aspects of control more closely, *dynamic control planning,* by which the agent determines and guides its own reasoning behavior; *focus of attention*, by which the agent parameterizes the behavior of its perception/action subsystems; and the *satisficing cycle* by which the agent controls the time spent on each reasoning cycle.

3.5 Dynamic Control Planning

A *control plan* is a temporally organized pattern of *control decisions*, each of which describes a class of operations the agent intends to perform during some period of time. Control decisions may vary widely in content and specificity, ranging from specific primitive operations intended to be executed at particular moments in time to broad classes of operations intended to be executed during extended time intervals. Control decisions may "stand alone," specifying an independently desirable class of actions. Alternatively, sets or sequences of control decisions may be coordinated to achieve a common objective. Multiple competing, complementary, or independent control decisions regarding a particular time interval may co-exist in the control plan. Multiple constituent plans for performing concurrent tasks may co-exist in the control plan.

For example, Figure 2 illustrates an abstract control plan comprising three constituent plans. Plan A is a single, independent, long-term control decision governing behavior prior to, during, and beyond the time period shown. Plan B is another single, independent, long-term control decision governing behavior during a period that begins during the time period shown and continues into the future. Plan C is a local plan, governing behavior during a sub-interval of the time period shown. In addition, Plan C is elaborated in terms of more detailed subordinate decisions at two lower levels of abstraction, which govern behavior during a hierarchically organized sequence of component time intervals.

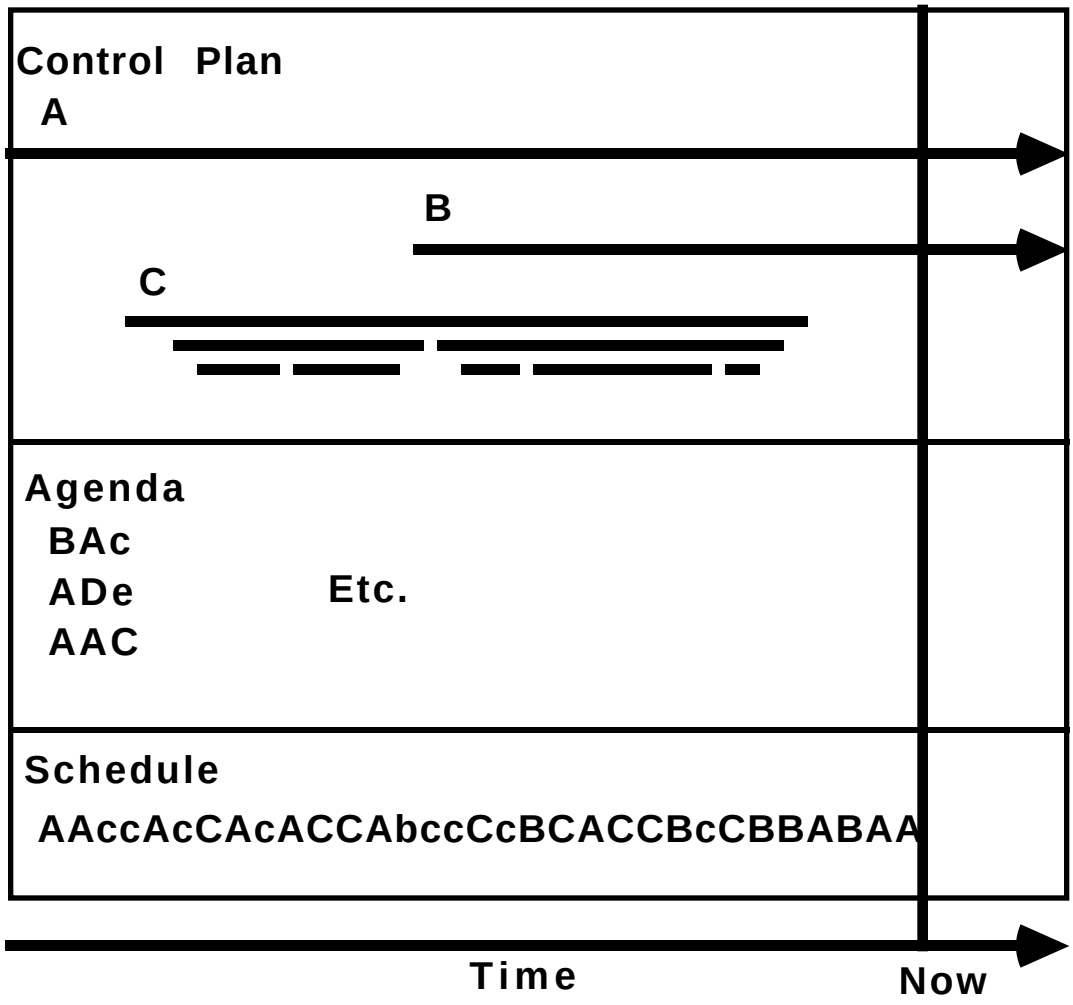


Figure 2. An Abstract View of Dynamic Control Planning. The top panel shows a control plan with constituent plans for tasks A, B, and C, governing the agent's behavior during overlapping time intervals. The middle panel shows the dynamic agenda of executable control (lower case) and task (upper case) operations relevant to tasks A, B, C, D, and E. The bottom panel shows the schedule of control and task operations chosen for execution in accordance with active control plans and agendas during corresponding time intervals.

Control plans can represent not only what task to perform, but also how to perform it given particular policies or resource constraints. In particular, control plans can indicate preferences for reasoning operations that meet time constraints. For example, in Figure 2, Plan C may have been constructed as shown because it will produce a satisfactory outcome within the designated period of time. Alternative control plans may have produced "better" outcomes, but taken longer to do so.

Control plans can represent not only what task to perform, but also how to perform it given particular policies or resource constraints. In particular, control plans can indicate preferences for reasoning operations that meet time constraints. For example, in Figure 2, Plan C may have been constructed as shown because it will produce a satisfactory outcome within the designated period of time. Alternative control plans may have produced "better" outcomes, but taken longer to do so.

The cognitive system constructs control plans incrementally by means of control operations that generate or modify constituent control decisions. As illustrated in Figure 2, the cognitive system treats control operations (lower case) like other reasoning operations. Control operations are suggested by perceptual or cognitive events, rated and placed on the agenda, and scheduled for execution. Thus, they compete for execution with one another and with all other executable operations.

Different control operations embody different reasoning methods [33]. Some operations generate control decisions bottom up, for example when a perceptual event triggers a decision to respond to the perceived situation. In Figure 2, Plans A and B and the top-most decision of Plan C presumably were generated bottom-up in response to perceived demands or opportunities. Other operations generate decisions top-down, for example when an abstract control decision triggers a sequence of more specific control decisions. In Figure 2, the subordinate decisions in Plan C presumably were generated top-down to elaborate the more general decisions. Other operations generate decisions in a goal-directed fashion, for example when a lack of operations satisfying a prior control decision triggers a decision to perform operations that would trigger such operations. In Figure 2, if no operations on the agenda satisfy Plan A, goal-directed reasoning would generate a decision to perform operations whose results would trigger operations compatible with Plan A.

Control decisions may be generated at any time prior to the time at which they are intended to influence the agent's behavior. Some control decisions are generated and take effect immediately, while others are generated in advance and do not take effect until much later. For example, in Figure 2, Plan C might have been generated in response to a perceived event immediately prior to its initiation point. Alternatively, the agent might have decided much earlier that at that point in time it would follow Plan C. Similarly, the agent has decided that both Plans A and B will persist into the future, governing behavior well beyond the "Now" point.

Regardless of the content or specificity of control decisions, the reasoning methods used to generate them, and the times at which they are

generated, all control decisions appear in a single control plan. The agenda manager rates executable operations against all active control decisions whose time intervals include the current time. For example, early in the time interval shown in Figure 2, the agent uses Plan A to rate and schedule reasoning operations. Later, it uses Plans A and C--actually, the current lowest-level decisions of Plan C. Still later, it uses all three Plans, A, B, and C. Following the completion of Plan C, the agent uses Plans A and B to rate and schedule operations for the remainder of the time interval shown and into the future. As mentioned above, we intend to limit the size or complexity of an agent's active control plan during a given time period, but we have not yet implemented such limitations.

In general, the agent can perform operations that change its control plan on any cycle, thereby changing the rating criteria subsequently used by its agenda manager and, as a consequence, the operations subsequently chosen by its scheduler for execution. Dynamic control planning allows the agent to construct strategic plans that are appropriate to an evolving task environment and to follow strategic plans to which it has committed, but also to change those plans as appropriate.

3.6 Focus of Attention

The cognitive subsystem determines the agent's global focus of attention by sending perception/action subsystems control parameters determined by its dynamic control plan and other state information. As discussed above, *perceptual control parameters* are of three types. *Abstraction forms* specify desired transformations on data values. *Filtering criteria* specify conditions under which abstracted data should be sent to the cognitive system. *Annotation standards* specify criteria for determining the *relevance*, *importance*, and *urgency* of perceived data.

The architecture provides three kinds of *perceptual focus operations*, all of which, when executed, place control parameters in output buffers for relay to perceptual preprocessors [6, 54]. *Information-focusing* operations, which are triggered by changes in the agent's control plan, send focus instructions to discriminate among different kinds of input data. For example, if a control decision initiates a new reasoning task, an information-focusing operation will send a perceptual control parameter to increase the relevance, importance, or urgency of the associated data types. Thus, the agent will focus its interpretation of sensed information on those data that are useful to its reasoning. *Resource-focusing* operations, which also are triggered by changes in the control plan, modulate the overall input data rate in anticipation of changing resource

demands. For example, if a new task is computationally intensive, a resource-focusing operation will send a parameter that tightens the filtering criteria on all data types in proportion to their relevance and importance. Thus, the agent will focus its perceptual resources on types of data anticipated to be most useful. *Load-balancing* operations, which are triggered by overflow or underflow conditions in the cognitive input buffers, also modulate the overall input data rate, but they do so in response to unanticipated changes in resource demands. For example, if input data arrive faster than the cognitive system can process them, producing repeated input buffer overflows, a load-balancing operation will send a perceptual parameter that tightens filtering criteria. Conversely, if the cognitive system has the capacity to process more frequent input data, a load-balancing oepration will send a parameter to loosen the filtering criteria. Thus, the agent will coordinate its input data rates with its dynamic cognitive capacity to incorporate new input data.

With these operations, the agent focuses its perception of a complex, dynamic environment "top-down," in accordance with its current control plans and available resources. Thus, it protects its cognitive system from being swamped by non-critical inputs. However, the agent remains sensitive to exceptional events outside of its current focus of attention. One way is by instructing perceptual subsystems to relay all data values that fall in critical ranges. In our current work, we "hard-wire" very general forms of these criteria so that the agent is guaranteed to notice extreme events. In addition, preprocessors can potentially redirect their own attention in response to particular patterns of sensed data. Although we have not yet implemented such "peripheral responses," we anticipate that they will play an important role in maintaining an agent's sensitivity to important unanticipated events in a dynamic environment.

As mentioned above, we are studying corresponding sorts of focus operations to set action control parameters related to performance criteria, resource consumption, and side effects.

3.7 The Satisficing Cycle

Let us examine the cognitive system's *satisficing cycle* [28]. Because this cycle is the unit-process underlying all reasoning, bounding and, in fact, controlling its computation time is a prerequisite to controlling computation times for reasoning tasks under real-time constraints. Recall that the cycle comprises three processes: the agenda manager, the scheduler, and the executor. The scheduler's computation time is easily bounded and insignificant. The executor's computation time depends upon

the operation it is executing. We currently rely upon programming guidelines to bound operation execution time within acceptable ranges (but see section 5.) Therefore, our efforts to bound and control cycle time have focused on the agenda manager.

As discussed above, the agenda manager identifies and rates executable operations based on cognitive and perceptual events. The time consumed by agenda management is an increasing function of the number of known operations, the number of perceptual and cognitive events, and the number of rating criteria in the control plan. Given the continuous flow of events in the environment and the many tasks and operations an intelligent agent can perform, identification of all executable operations can take a very long time. Given real-time constraints on the agent's behavior, the agenda manager ordinarily cannot identify all currently executable reasoning operations before the agent must execute one of them. Conversely, there is no need to identify the many possible operations that the agent will never execute.

Therefore, the agenda manager is designed to operate in an incremental, non-exhaustive fashion, identifying and rating a subset of the executable operations one at a time and terminating according to current *cycle parameters*. These parameters are of three types. *Criterial operations* describe executable operations that, when identified by the agenda manager, would be "good enough" to execute. *Criterial events* and *deadlines* describe perceptual or cognitive events or specific times whose occurrence requires immediate execution of the "best available" operation. (Criterial events may be viewed as uncertain deadlines.)

Cycle parameters are determined and modified dynamically by the agent's own reasoning in the context of its dynamic control plan. For example, if the current control plan simply specifies operations of a particular type, any executable operation of that type would be "good enough." Other things being equal, a task deadline in the control plan would be translated heuristically into component deadlines for individual reasoning operations.

The occurrence of any condition specified in the current cycle parameters causes the agenda manager to terminate. The scheduler then chooses the highest priority operation on the current, usually incomplete agenda and the executor executes it. In the case of a criterial operation, the highest priority operation will be, by definition, one that is "good enough." In the case of a criterial event or deadline, the highest priority operation will be the "best available" one at that time. We have

experimented with cycle parameters that specify criterial operations and deadlines, but not yet with criterial events.

Because the agenda manager is non-exhaustive, the order in which it identifies executable operations is critical. To maximize the speed with which it identifies "good enough" operations and to maximize the priorities of the "best available" operations at those times when it is interrupted by criterial events or deadlines, the agenda manager applies a *heuristic best-first algorithm*. Using whatever criteria appear in the the current control plan, it attempts to instantiate the highest priority operation type for the highest priority event on each iteration. Viewing agenda management as a generate-and-test problem, this algorithm effectively moves some of the test criteria into the generator.

For a given control plan and set of events, the heuristic best-first algorithm identifies executable operations in roughly descending order of priority. How closely it approximates the actual descending order depends on the details of the control plan and the order in which rating criteria are applied. However, because perception and cognition are asynchronous, the agenda manager works with a dynamic set of perceptual events and control decisions, incorporating new ones into its computations as they occur. Thus, it often happens that newly identified executable operations have significantly higher priorities than those already on the agenda.

The agenda manager places each newly identified executable operation on the agenda, ordered by priority. As mentioned above, the agenda has limited capacity, with best-first (highest priority) retrieval by the scheduler and worst-first overflow. Thus, at any point in time, the agenda constitutes a short, ordered list of high priority reasoning operations suggested by recent high-priority events and control decisions.

The satisficing cycle can produce a spectrum of agent behavior, depending on the agent's dynamic control plan and cycle parameters. Control plans are *discriminating* to the degree that they restrict the assignment of high priorities to a smaller set of events and operations. Other things being equal, more discriminating control plans facilitate rapid identification of high-priority executable operations. Cycle parameters are *stressful* to the degree that they reduce the time available for agenda management (lower thresholds for criterial events and operations, short deadlines). Other things being equal, more stressful cycle parameters lead to rapid execution of a large number of operations.

These two factors interact to determine the agent's style of behavior. For example, given a very discriminating control plan and non-stressful

cycle parameters, an agent would appear to behave "methodically," executing a small number of very high priority operations per unit time. With more stressful cycle parameters, the agent would appear to behave "purposefully," performing more operations per unit time and perhaps compromising quality by performing some lower-priority operations. With very stressful cycle parameters, the agent could still behave "purposefully" if, for example, its control plan restricted its triggering of executable operations to a very small set of very important operations, categorically excluding less important operations. At the other extreme, given an undiscriminating control plan and very stressful cycle parameters, an agent would appear to "thrash," executing a large number of arbitrary operations per unit time. In fact, given an undiscriminating control plan, the agent's behavior would appear arbitrary regardless of cycle parameters, varying primarily in rate of executed operations.

Ideally, it seems that intelligent agents should perform near the "methodical" end of the spectrum when time and other resources permit and move cautiously along the spectrum when required to do so by time and other resource constraints. In practice, we anticipate that many agents will not often have the luxury of behaving "methodically." However, we are more optimistic about agents' ability to behave "purposefully" by constructing effective control plans. We are exploring these issues.

4. Satisfaction of Real-Time Control Requirements

Let us briefly summarize how the proposed agent architecture is hypothesized to address the requirements introduced in section 2.

Communications. Information passes from the environment to perception subsystems, from perception subsystems to cognition and action subsystems, and from the cognition subsystem to perception and action subsystems.

Asynchrony. Parallel subsystems, with buffered communications, provide asynchronous perception, cognition, action.

Selectivity. Limited-capacity event buffers selectively favor "high priority" inputs--those that are recent, relevant, important, and urgent. Perception/action subsystems selectively process high priority sensed data and intended actions. The agenda manager selectively triggers and schedules high priority operations. Dynamic control plans selectively favor high priority reasoning tasks and establish associated focus of attention parameters.

Recency. Limited-capacity buffers with best-first retrieval and worst-first overflow favor recent items, as does the heuristic best-first agenda manager.

Coherence. Dynamic control plans provide a global focus of attention to coordinate perception, cognition, and action over time. They also strategically organize reasoning operations within a task and among concurrent reasoning tasks.

Flexibility. Exceptional events can override global focus of attention in perceptual preprocessors or the cognitive system.

Responsivity. Graduated reactive responses--peripheral, reflex, and cognitive responses--span a range of latencies. Within cognitive responses, additional gradations are supported. The agenda manager can control cycle time. Dynamic control planning can establish deadlines and discriminate among alternative reasoning methods strategies.

Timeliness. Satisfying each of the requirements discussed above contributes to an agent's timely response to the most important events. In addition, dynamic control planning allows an agent to reason explicitly about the time requirements of alternative operations and the time constraints on its behavior.

Robustness. Satisfying many of the requirements discussed above entails trading amount of computation, and therefore, expected quality of response, against latency of response, in a gradual manner.

Scalability. Several aspects of the architecture are designed to accommodate changes in scale. For example, perceptual preprocessing and focus of attention will protect the agent against increasing perceptual overload. Given a discriminating control plan, the satisficing cycle will produce stable cycle times regardless of increases in problem size.

Development. Increases or improvements in knowledge should improve the agent's ability to meet several of these requirements. For example, improvements in its control knowledge should enable it to focus perceptual attention more effectively, improve the strategic control of its reasoning, and execute higher-priority operations more rapidly.

5. The Guardian Application

Because our long-term research goal is to develop a general architecture for intelligent agents, experimental development of agents that operate in diverse domains is a major part of our research. Each new domain tests the sufficiency and generality of the current architecture and presents new requirements for subsequent versions of the architecture. To illustrate how agents are implemented within the proposed architecture, we briefly discuss the Guardian system for intensive care monitoring [27].

5.1 Guardian's Task Environment and Requirements

The sickest surgical patients in the hospital are cared for in the surgical intensive care unit (SICU). Most of these patients have temporary failure of one or more organ systems--usually the lung or the heart--which is treated with life-support devices that assume the fundamental functions of the ailing system until it heals. For example, the ventilator is an artificial breathing machine that augments the patient's own breathing. Life-support devices are adjusted based upon frequent patient observations. Some observations are made continually and automatically, for example, measurements of air pressures and air flows in the patient-ventilator system. Other observations are made intermittently. Blood gases, for example, are measured once every hour or so, while chest xrays are usually taken once or twice a day. Based on patient observations, device settings are adjusted to vary the amount of assistance the device provides. For example, ventilator settings determine the number of breaths delivered to the patient per minute, the volume of air blown into the patient's lungs on each breath, and the amount of oxygen in the air. Other therapeutic actions might include adjusting a ventilator tube, clearing the patient's air passages, administering drugs, etc. The short-term goal of SICU monitoring is to keep the patient as comfortable and healthy as possible, while progressing toward therapeutic objectives. The long-term goal is to withdraw life-support devices gradually so that the patient eventually can function autonomously.

Although we do not anticipate using Guardian in closed-loop mode in a hospital setting, our objectives for it include all of the perception, reasoning, and action necessary for closed-loop control. Thus, Guardian's task instantiates all of the requirements for real-time control discussed earlier in this paper. Because Guardian has access to over one hundred automatically acquired patient data variables, each of them sensed

several times per second, and because it can reason about and act upon these observations in many different ways, Guardian must selectively perceive important patient data and perform key reasoning operations that contribute to its performance of the most important actions. Because the patient embodies a dynamic physical process with its own temporal dynamics, Guardian must asynchronously perceive patient data, reason about the patient's condition, and perform therapeutic actions. To insure that its behavior is current, Guardian must "forget" unrealized past opportunities for perception, reasoning, and action in favor of present opportunities. To achieve longer-term therapeutic goals, Guardian must enact a coherent pattern of perception, reasoning, and action over a period of time. On the other hand, uncertain changes in the patient's physiological condition require flexibility and adaptation. Guardian must be responsive to patient conditions of varying urgency; other things being equal, the more urgent the patient's condition is, the more quickly Guardian must perceive the relevant information, perform the necessary reasoning, and execute the appropriate actions. Guardian must satisfy a variety of hard and soft real-time constraints on the utility of its behavior. Because Guardian inevitabably will encounter situations that strain or exceed its capacity--too many important new signs and symptoms, too many important interpretation, diagnosis, prediction, and planning tasks, too many important therapeutic actions--its performance must degrade gracefully and not precipitously. Guardian must maintain the quality of its behavior as we scale up to more realistic problems. Ideally, it should improve the utility of its behavior as it acquires more knowledge.

5.2 Guardian's Current Implementation and Performance

Figure 3 illustrates how Guardian instantiates the proposed agent architecture and how it interacts with a simulation of the patient-ventilator system and hospital laboratories.

Figure 3. Guardian's Current Design and Implementation.

A single perceptual preprocessor currently manages Guardian's perception of twenty automatically sensed patient data variables, with an average overall sensed data rate of one data value per second. In addition, Guardian perceives irregularly reported lab results and messages from human users. Each sensed data value, if passed to the cognitive system, would trigger a number of cognitive operations, whose execution would produce a number of cognitive events and trigger new operations. Thus, although this is not a high data rate in absolute terms, it is considerably beyond Guardian's current cognitive capacity, which is one cognitive operation every two to fifteen seconds with an exhaustive agenda manager and controllable to within a couple of seconds with the heuristic agenda manager. Moreover, we anticipate that, during the next twelve months, Guardian's sensory activity will increase from twenty to one hundred automatically sensed variables, with each of them sensed at least once per second. There will be about twenty irregularly sensed data variables. Finally, as SICU technology advances, Guardian will have access to new data. Thus, Guardian faces significant and growing perceptual overload.

To avoid falling behind real time, Guardian's perceptual preprocessor applies dynamic abstraction, filtering, and annotation parameters sent by the cognitive system. It abstracts numerical data values into value classes and trends. It assigns data values to three levels of importance: life-threatening, abnormal, and other. It distinguishes data that are relevant to ongoing reasoning activities from those that are not relevant. It distinguishes three levels of urgency: events that permit an effective response within four minutes, one hour, or longer. It filters data based on criterial value changes within deadlines. Thus, the cognitive system can bound the variability of unsent intervening values. Using these mechanisms, the preprocessor typicaly reduces sensed data rates by over 90%, maintaining an average overall perception rate of approximately one perceptual input every twenty-two seconds, without reducing solution quality [54]. Additional selectivity is provided by the cognitive system itself. Our preliminary experiments suggest that the proposed approach to perceptual preprocessing will scale up to protect Guardian from overload under the anticipated increase in sensed data rates [6].)

Guardian has a wide range of medical knowledge including: knowledge of meaningful classifications and trends of the twenty-five currently sensed patient data variables; knowledge of a twenty-node hierarchy of respiratory disease conditions, patient data that probabilistically

implicate those diseases, and therapeutic actions that correct them; knowledge of the normal structure and function of the respiratory, circulatory, pulmonary exchange, tissue exchange, and tissue metabolism systems; knowledge of the normal structure and function of the ventilator; knowledge of the normal and abnormal structure and function of abstract flow, diffusion, and metabolic systems; knowledge of prototypical therpautic protocols for managing a small number of evolving disease conditions; knowledge of the importance and urgency of particular observations and diagnoses; knowledge of the precondition, results, and time required to perform a number of therapeutic actions.

Guardian also has knowledge about performing several reasoning tasks, including: interpretation of time-varying data, diagnosis of observed signs and symptoms, determination of corrective actions for diagnosed conditions, prediction of future physiological conditions, explanation of observations, diagnoses, and predictions, and dynamic therapy planning. Moreover, for most of these tasks, it has both *associative* and *model-based reasoning methods*. Associative methods capture clinical knowledge and permit quick responses to familiar situations. Model-based methods capture more fundamental biological and physical knowledge and permit more thorough (and time-consuming) responses to both familiar and unfamiliar cases. Each reasoning method is implemented as a set of abstract reasoning operations that are triggered by particular kinds of perceptual or cognitive events, along with control operations that construct resource-bounded control plans in particular contexts. The results of all reasoning activities are recorded in temporally organized episodes in the global memory.

Depending upon the circumstances, Guardian may be logically capable of pursuing many different reasoning tasks with both associative and model-based methods. Given the real-time constraints on its behavior, however, Guardian typically must be quite selective about which tasks it pursues and how it allocates reasoning resources among them. Accordingly, it uses strategic knowledge to construct a dynamic global control plan that differentially favors the triggering and scheduling of executable operations involved in competing reasoning tasks.

For example, in one scenario, Guardian observes that a post-operative patient has low body temperature. It makes a global control decision to perform a sequence of reasoning tasks: diagnosing the low temperature; predicting a spontaneous rise in temperature to normal over a period of hours; predicting the undesirable physiological consequences of low temperature; and planning a course of action to be executed over a period

of hours to avoid those consequences. Within each of these tasks, Guardian makes local control decisions about whether to apply associative or model-based reasoning methods and how to organize its reasoning within the chosen method. At the same time, Guardian's global control plan also allows it to incorporate new perceptions, but not to reason about most of those perceptions since they are less important than ongoing activities.

As the scenario continues Guardian deviates from this purposeful behavior only when a new perception, very high peak inspiratory pressure, indicates a life-threatening patient condition with a four-minute deadline. Guardian makes a new global control decision to direct all of its resources to correcting this critical condition as quickly as possible. This decision impacts three aspects of Guardian's behavior. Its perceptual preprocessor refocuses to favor patient data relevant to the high peak pressure and to minimize distraction by less important data. Its agenda manager adopts a shorter deadline to insure a quick sequence of responses under a short deadline. And, given the content of the new control decision, its agenda manager and scheduler favor associative reasoning operations (because they have shorter latencies) that diagnose and act to correct the high pressure problem. Given these adaptations, Guardian very quickly (within a minute) performs a sequence of operations to deal with the high peak pressure: diagnoses the immediate problem, inadequate ventilation; increases the breathing rate so the patient will get enough oxygen; diagnoses the underlying problem, a pneumothorax (hole in the lung); performs (on the simulated patient) the appropriate action, inserting a chest tube to relieve the pressure of accumulated air in the chest cavity; reduces the breathing rate now that the pressure is relieved; confirms that the pressure is normal; and confirms that the blood gases are normal. Once the problem is solved, Guardian makes a new global control decision to resume its previous interrupted activities.

Several display drivers manage Guardian's communications with human users. These communications include dynamic graphical displays of: the patient's SICU history; ongoing reasoning and results related to diagnosis, prediction, and therapy planning; structure/function explanations of the patient's condition, diagnosis, prognosis, and therapy; and Guardian's current global control plan. Each of these displays is interactive, permitting the user to pose particular kinds of questions, as well as reviewing previous observations and conclusions.

Guardian can run either closed-loop, executing recommended actions directly on the simulation, or open-loop, simply recommending actions, which human users decide whether or not to execute.

We have developed Guardian's architecture and component capabilities for a small number of characteristic SICU scenarios. Although this knowledge base is far from complete, it allows Guardian to handle a wider set of SICU scenarios than we have actually tested it so far. In addition to extending and refining Guardian's component capabilities, our current work involves collecting a library of new SICU scenarios to identify the limits of Guardian's current knowledge base and to drive extension of the knowledge base. Although our patient simulator provides realistic SICU data, we are interested in evaluating Guardian on real patient data. We have begun collecting patient histories for "re-enactment" studies. We are investigating establishing a direct link between Guardian and computers in the SICU at the Palo Alto Veterans Administration Medical Center.

6. Other Approaches to Intelligent Agents

6.1 Variations on the Proposed Architecture

Designing an agent architecture involves making design decisions in a large space of design features. To put our proposed architecture in perspective, we mention a few of the features we have considered and rejected and a few that we are planning to explore further.

We designed the satisficing cycle and heuristic agenda manager as a replacement for the optimizing cycle and exhaustive agenda manager that we and others have used in the past [9, 13, 25]. This appears essential for real-time performance and probably for efficient performance in large non-real-time systems that have a lot of knowledge and run for many cycles. The present satisficing cycle preserves the sequential nature of the optimizing cycle. However, we are exploring the possibility of allowing agenda management to run continuously, with parallel scheduling and execution of criterial or best available operations. So far, we have finessed the problem of unbounded operation execution times by imposing programming constraints. However, we are studying more flexible approaches that would allow variable computation times for executed operations, with the possibility of interruption by identification of newer, higher-priority next-operations. Depending upon the specification of the operation currently being executed, the executor would either abort execution of the current operation or suspend it and place a rated resumable form of the operation in an appropriate position on the agenda.

Regarding limited-capacity buffers, we have given some thought to introducing spontaneous temporal decay of items in buffers. In a dynamic environment, even very important events are perishable and may not

warrant processing after a period of time. Although it has been suggested to us that an agent's buffer capacities might be variable in different contexts, we continue to assume that they are static. However, we are investigating the concept of limited-capacity "back-up buffers" which catch and preserve very important overflow items.

We considered modeling perception and action processes as operations in the cognitive system, but that approach did not provide the desired asynchrony and interfered with timeliness [32].

Finally, we have distributed perception, cognition, and action among parallel processes because they respresent minimally interacting, coarse-grained chunks of knowledge and computation. Therefore, we hypothesize that they can be distributed among parallel processes without incurring excessive communications demands or knowledge redundancy. So far, that hypothesis seems to be correct. Although we have considered distributing cognitive tasks among parallel processes [29], our experience with Guardian suggests that cognitive tasks have many important interactions, including sequential constraints, and associated needs for communication. Operating on a single processor in the context of a single global data structure supports these interactions, so we would favor distribution of cognitive tasks only in a shared-memory architecture.

6.2 Alternative Architectures

A considerable body of research has focused on "classical" planners [17, 44]. Under this model, an agent perceives information from the environment and then constructs a goal-oriented sequence of actions, a plan, which it subsequently executes. Classical planning architectures are not intended to provide comprehensive capabilities for intelligent agents, so it is not surprising that they do not satisfy all of the requirements for real-time control put forth in this paper. Global coherence is the most prominent advantage of classical planners. However, the computational cost of formulating a complete plan by reasoning backward from goals can be excessive [8]. Classical planners do not meet the other requirements.

Relaxing this perceive-plan-act sequence, some researchers allow the agent to interleave planning and execution, either to build the plan incrementally or to modify the plan in response to unanticipated conditions [9, 20, 25, 31, 35]. Other researchers introduced more knowledge intensive and computationally tractable methods for generating partial plans, including: instantiating goal-oriented action schemas [19, 26]; integrating top-down and bottom-up planning methods [24, 33],

transferring successful plans to new situations [11, 23]; or successively applying constraints among potential actions [51]). Interleaving planning and execution permits an agent to several real-time requirements. However, their success is limited by unbounded computation times for component processes, especially the match processes that trigger reasoning operations. Although researchers have made progress in developing efficient match algorithms [18, 22], these approaches only speed up the match process. They do not reduce the computational complexity of the process and, more importantly, they do not permit an agent to directly control the amount of time spent on the match process.

By contrast, in an effort to avoid the computational cost of control reasoning and thereby create real-time responsivity, some researchers have turned their attention to the theory, design, and implementation of "reactive agents" [1, 2, 39, 43, 45]. Basically, reactive agents store large numbers of perception-action rules in a computationally efficient form and execute actions invoked by environmental conditions on each iteration of a perceive-act cycle. Thus, they are similar to control theoretic methods [5], where traversal of symbolic networks replaces computation of numerical models. Reactive models often assume synchronization of reactive cycles with the occurrence of events in the environment. Selectivity is achieved to the degree that the system builder has encoded it in the network and flexibility is a natural consequence of the perceive-act cycle. On the other hand, coherence occurs only fortuitously, presumably emerging from the agent's characteristic reactions to events in an orderly task environment. Reactive agents provide responsiveness and robustness only when perception-action networks include context-specific alternative subnetworks. In general, we view the reactive agent model as a good framework for engineering solutions to particular, narrowly defined feedback control tasks for which control-theoretic models are inapplicable--those for which numerical models are either non-existent or intractable. It also might be an appropriate mechanism for low-level perception-action programs that by-pass the cognitive subsystem within the proposed architecture. For example, the proposed architecture might incorporate reactive peripheral programs for focusing perceptual attention or feedback control of actions. However, we suspect that the reactive model is not an appropriate general model for tasks that present challenging requirements for selectivity, global coherence, responsivity, or robustness. And it is not appropriate for complex tasks or for the multiple task behavior expected of generally intelligent agents where enumerating all possible perception-action contingencies and encoding them in a computationally tractable form may be infeasible.

Finally, robotics researchers aim to build "task-level" robot systems (]14, 36]. Unlike robots programmed to perform specific mechanical tasks, task-level robots are intended to accept high-level goals and then determine and perform whatever behaviors are necessary to achieve the goals. They are intended to operate under a variety of incidental contextual conditions, including low-frequency exceptional conditions related to hardware, software, or environmental state. Significant applications of this work include efforts to build autonomous vehicles [21, 37]. Robotics work is similar in spirit to the present research, integrating perception, action, and cognition to achieve goals in a real-time task environment. However, robotics research traditionally has focused on challenging perceptual-motor tasks, only recently beginning to incorporate more cognitive activities, such as goal determination, planning, exception handling, and learning [4]. Conversely, our work grows out of earlier work emphasizing reasoning and problem solving, with new emphases on perceiving and acting in a real-time environment.

7. Limitations of the Proposed Architecture

Despite our interest in the proposed architecture, we must acknowledge that it makes agents vulnerable to errors that do not occur under conventional software architectures. By definition, the architecture's real-time control mechanisms--its perceptual filtering, limited capacity I/O buffers, dynamic control planning, focus of attention, and satisficing cycle--allow an agent to ignore many opportunities to perceive, reason, and act and to perform sub-optimal operations. In general, the agent allocates limited computational resources among competing activities in proportion to their urgency and importance. In many cases, this will not affect the global utility of the agent's performance. In others, it will produce acceptable degration in particular aspects of performance. In extreme cases, however, an agent might decide prematurely to perform costly, ineffective, or counterproductive operations; or it could fail to perform highly desirable operations that are well within its capabilities. Nonetheless, it is our hypothesis that, if we wish to build agents that function well in complex real-time environments, of which the natural environment is a prime example, we must forego optimality in favor of effective management of complexity [52]. Allowing the possibility of occasional, more or less consequential error is a necessary concession toward that end. Formulating control knowledge that allows an agent to meet the most important real-time performance requirements while minimizing the impact of incompleteness and suboptimality is a primary objective of our research.

References

1. Agre, P.E., and Chapman, D. Pengi: An implementation of a theory of activity. Proceedings of the National Conference on Artificial Intelligence, 1987.

2. Andersson, R.L. A Robot Ping-Pong Player: Experiment in Real-Time Control. MIT Press, 1988.

3. Baker, T.P., and Shaw, A. The cyclic executive model and Ada. Real-Time Systems, 1:1, 17-26, 1989.

4. Bares, J., Hebert, M., Kanade, T., Krotkov, E., Mitchell, T., Simmons, R., and Whittaker, W. Ambler: An autonomous rover for planetary exploration. Computer, 22:6, 18-28, 1989.

5. Bollinger, J., and Duffie, N. Computer Control of Machines and Processes. 1988.

6. Boureau, L., and Hayes-Roth, B. Deriving priorities and deadlines in real-time knowledge-based systems. Proceedings of the IJCAI89 Workshop on Real-Time Systems, 1989.

7. Brinkley, S., Sha, L., and Lehoczky, J. Aperiodic task scheduling for hard-real-time systems. Real-Time Systems, 1:1, 27-60, 1989.

8. Chapman, D. Planning for conjunctive goals. Artificial Intelligence, 32:3, 333-378, 1987.

9. Corkill, D.D., Lesser, V.R., and Hudlicka, E. Unifying data-directed and goal-directed control: An example and experiments. Proceedings of the National Conference on Artificial Intelligence, 143-147, 1982.

10. d'Ambrosio, B., Fehling, M.R., Forrest, S., Raulefs, P., and Wilbur, M. Real-time process management for materials composition in chemical manufacturing. IEEE Expert, 1987.

11. Daube, F., and Hayes-Roth, B. A case-based mechanical redesign system. Proceedings of the International Conference on Artificial Intelligence, 1989.

12. Dodhiawala, R., Sridharan, N.S., Raulefs, P., and Pickering, C. Real-time

AI Systems: A definition and an architecture. Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, 1989.

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

14. Ernst, H.A. A computer-controlled mechanical hand. PhD Thesis, MIT, Cambridget, MA., 1961.

15. Fagan, L.M. VM: Representing time-dependent relations in a medical setting. PhD Dissertation, Stanford University, 1980.

16. Faulk, S.R., and Parnas, D.L. On synchronization in hard-real-time systems. Communications of the ACM, 31:3, 274-287, 1988.

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

18. Forgy, C.L. RETE: A fast algorithm for the many pattern/many object pattern matching problem. Artificial Intelligence, 19, 17-32, 1982.

19. Friedland, P.E. Knowledge-based experiment design in molecular genetics. Technical Report CS-79-71, Stanford University Computer Science Department, 1979.

20. Georgeff, M.P., and Lansky, A.L. Reactive reasoning and planning. Proceedings of the National Conference on Artificial Intelligence, 1987.

21. Goto, Y., and Stentz, A. Mobile robot navigation: The CMU system. IEEE Expert, Volume 2, 4, 44-54, 1989.

22. Gupta, A., Forgy, C., and Newell, A. High-speed implementations of rule-based systems. Technical Report, Carnegie-Mellon University, 1987.

23. Hammond, K. CHEF: A model of case-based reasoning. Proceedings of the National Conference on Artificial Intelligence, 1986.

24. Hayes-Roth, B., Hayes-Roth,, F., Rosenschein, S., and Cammarata, S. Modelling planning as an incremental, opportunistic process. Proceedings of the Sixth International Joint Conference on Artificial Intelligence, 6:375-383, 1979.

25. Hayes-Roth, B. A blackboard architecture for control. Artificial Intelligence, 26:251-321, 1985.

26. Hayes-Roth, B., Buchanan, B.G., Lichtarge, O., Hewett, M., Altman, R., Brinkley, J., Cornelius, C., Duncan, B., and Jardetzky, O. Protean: Deriving protein structure from constraints. Proceedings of the National Conference on Artificial Intelligence, 1986.

27. Hayes-Roth, B., Washington, R., Hewett, R., Hewett, M., and Seiver, A., Intelligent real-time monitoring and control. Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, 1989.

28. Hayes-Roth, B. A multi-processor interrupt-driven architecture for adaptive intelligent systems. Proceedings of the IJCAI89 Workshop on Real-Time Systems, 1989.

29. Hayes-Roth, B., Hewett, M., Washington, R., Hewett, R., and Seiver, A. Distributing intelligence within a single individual. In L. Gasser and M.N. Huhns (Eds.) Distributed Artificial Intelligence, Volume 2. Morgan Kaufmann, 1989.

30. Hayes-Roth, B. Making intelligent systems adaptive. In K. VanLehn (Ed.), Architectures for Intelligence. Lawrence Erlbaum, 1989.

31. Hayes-Roth, B. Dynamic control planning in adaptive intelligent systems. *Proceedings of the DARPA Knowledge-Based Planning Workshop*, Austin, Texas, 1987

32. Hewett, M., and Hayes-Roth, B. Real-Time I/O in Knowledge-Based Systems. In V. Jagannathan, R.T. Dodhiawala, and L. Baum (Eds.), Current Trends in Blackboard Systems, Morgan Kaufmann, 1989.

33. Johnson, M.V., and Hayes-Roth, B. Integrating diverse reasoning methods in the BB1 blackboard control architecture. Proceedings of the National Conference on Artificial Intelligence, 1987.

34. Laffey, T., Cox, P.A., Schmidt, J.L., Kao, S.M., and Read, J.Y. Real-time knowledge-based systems. AI Magazine, 9:1, 1988.

35. Lesser, V.R., Pavlin, J., and Durfee, E. Approximate processing in real-time problem solving. AI Magazine, 9:1, 49-62, 1988.

36. Lozano-Perez, T., Jones, J.L., Mazer, E., and O'Donnell, P.A. Task-level planning of pick-and-place robot motions. Computer, 22:3, 21-31, 1989.

37. McTamaney, L.S. Mobile robots: Real-time intelligent control. IEEE Expert, 2:4, 55-70, 1989.

38. Murray, W. Dynamic instructional planning in the BB1 blackboard control architecture. In V. Jagannathan, R. Dodhiawala, and L. Baum (eds.), Current Trends in Blackboard Systems, Morgan Kaufman,1989.

39. Nilsson, N. Action Networks. Working Paper, Stanford University Department of Computer Science, 1989.

40. O'Neill, D.M., and Mullarkey, P.W. A knowledge-based approach to real time signal monitoring. Proceedings of the Sixth National Conference on Artificial Intelligence Applications, 1989.

41. Pardee, W.J., Shaff, M.A., and Hayes-Roth, B. Intelligent control of complex materials processes. Proceedings of the Workshop on Blackboard Systems, 1989.

42. Rosenschein, S.J., Hayes-Roth, B., and Erman, L. Notes on methodologies for evaluating IRTPS systems. Proceedings of the AFOSR Workshop on Intelligent Real Time Problem Solving Systems. Santa Cruz, 1989.

43. Rosenschein, S.J., and Kaelbling, L.P. The synthesis of digital machines with provable epistemic properties. In J. Halpern (ed.), Proceedings of the Conference on Theoretical Aspects of Reasoning about Knowledge, Morgan Kauffman, 1986.

44. Sacerdoti, E.D. The non-linear nature of plans. Proceedings of the International Joint Conference on Artificial Intelligence, 1975.

45. Schoppers, M. Universal plans for reactive robots in unpredictable environments. Proceedings of the International Joint Conference on Artificial Intelligence, 1987.

46. Shoham, Y., and Hayes-Roth, B. Report on issues, testbed, and methodology for the IRTPS research program. Proceedings of the AFOSR Workshop on Intelligent Real Time Problem Solving Systems. Santa Cruz, 1989.

47. Smith, D.M., and Broadwell, M.M. The pilot's associate - An overview. Proceedings of the Eighth International Workshop on Expert Systems and their Applications.

48. Sowa, J. Conceptual Structures: Information Processing in Mind and Machine. Addition-Wesley, 1984.

49. Stankovic, J.A., Misconceptions about real-time computing: A serious problem for next-generation systems. Computer, 21:10, 10-19, 1988.

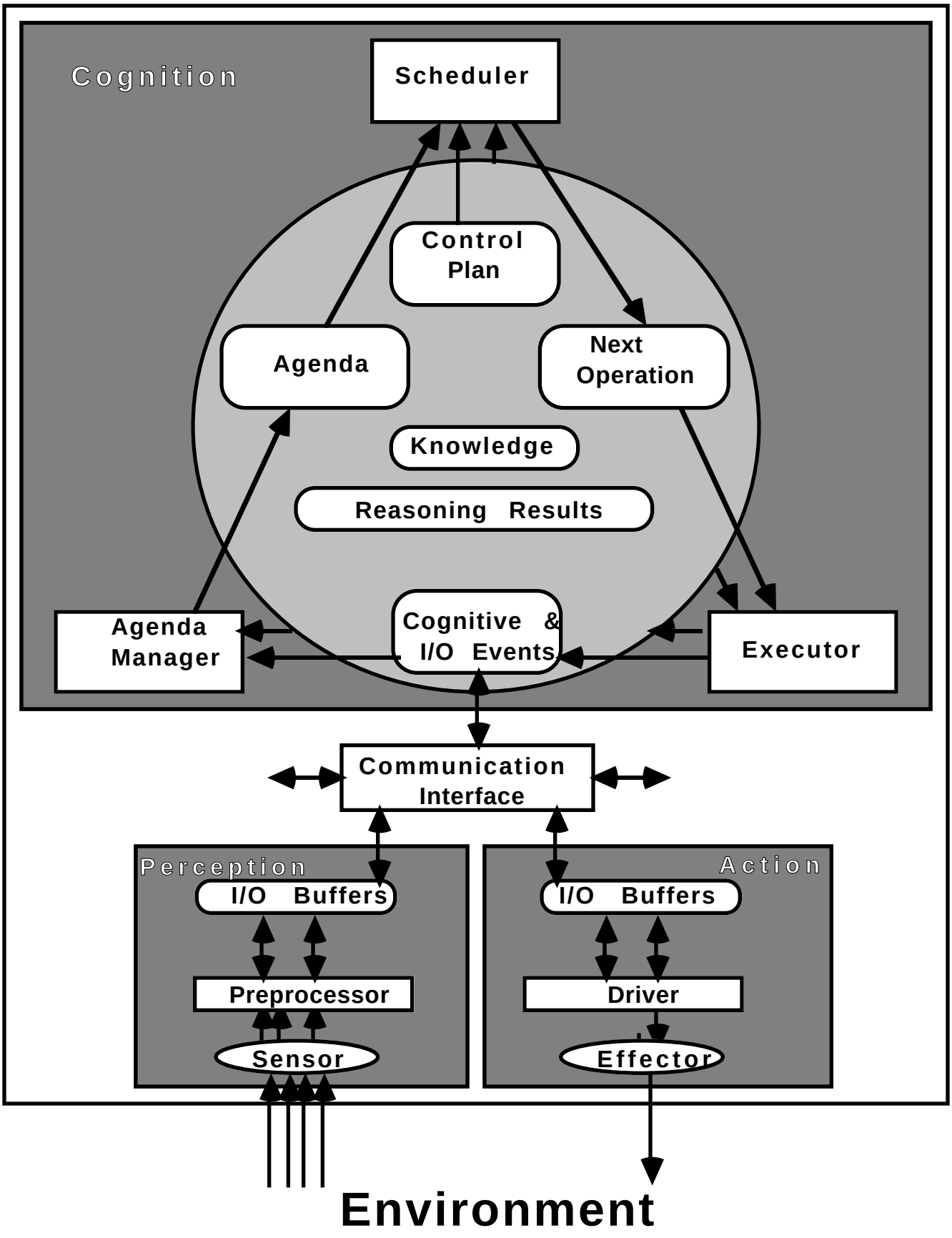
50. Stankovic, J.A., and Zhao, W. On real-time transactions. SIGMOD Record, 17, 4-18, 1988.

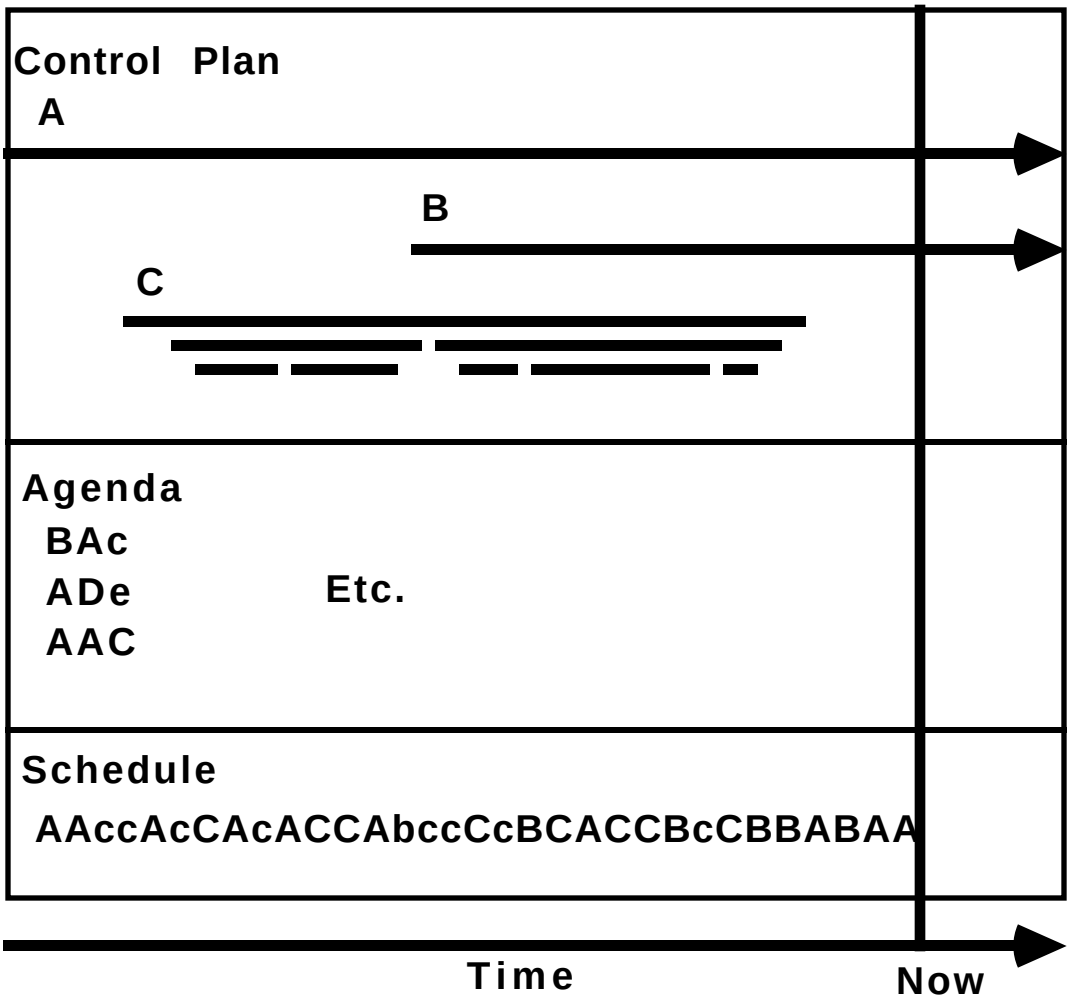
51. Stefik, M. Planning with constraints. Artificial Intelligence, 16:2, 111-140, 1971.

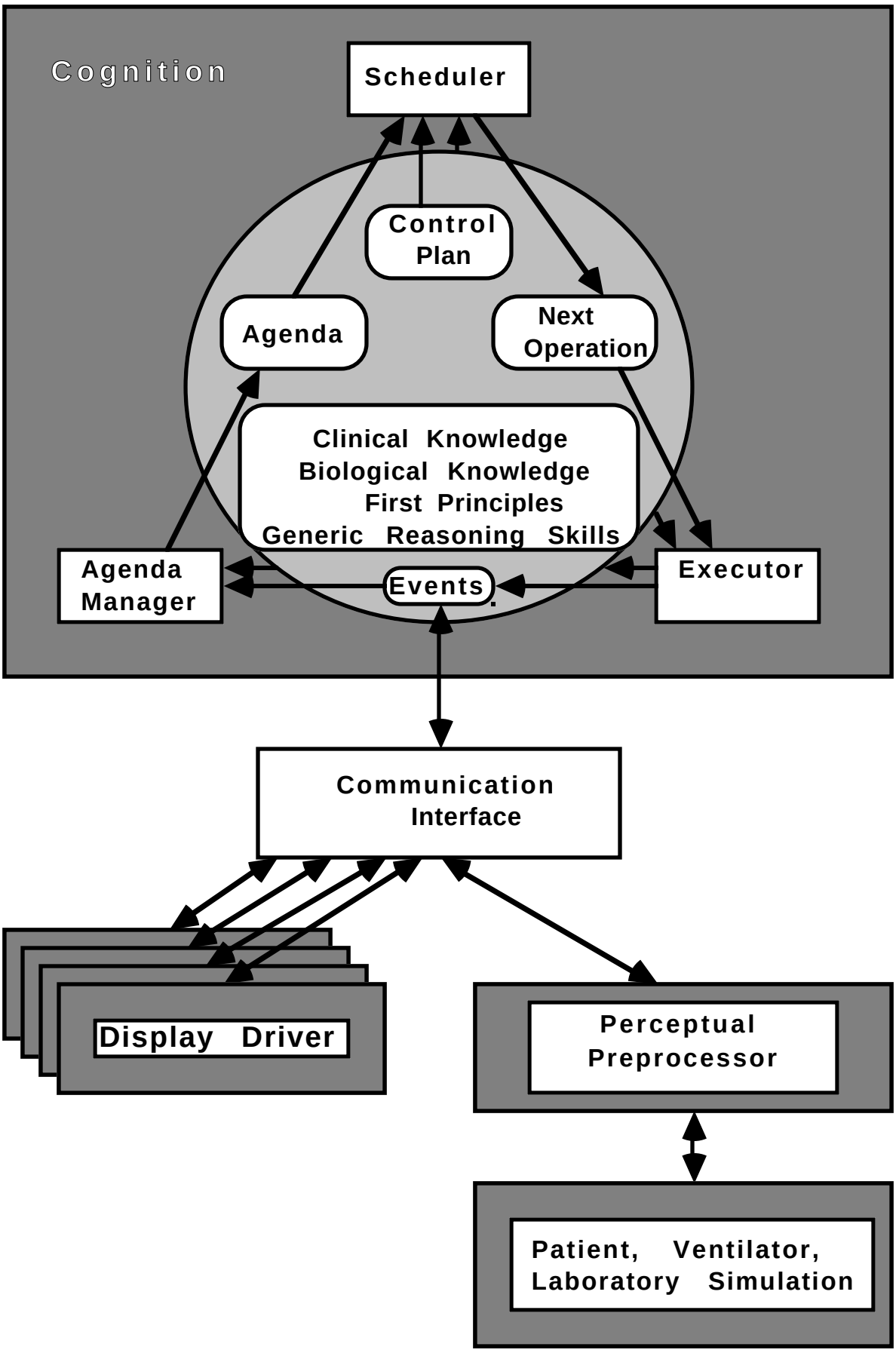
52. Simon, H.A. The Sciences of the Artificial. MIT Press, 1969.

53. Touchton, R.A. Reactor emergency action level monitor. Technical Report NP-5719, Electric Power Research Institute, 1988.

54. Washington, R., and Hayes-Roth, B. Managing input data in real-time AI systems. Proceedings of the Eleventh International Joint Conference in Artificial Intelligence, 1989.


Cognition
Scheduler
Control Plan
Agenda
Next Operation
Knowledge
Reasoning Results
Agenda Manager
Cognitive & I/O Events
Executor
Communication Interface
Perception
I/O Buffers
Preprocessor
Sensor
Action
I/O Buffers
Driver
Effector
Environment


Control Plan
A
B
C
Agenda
BAc
ADe
AAC
Etc.
Schedule
AAccAcCAcACCAbccCcBCACCBcCBBABAA
Time
Now


Cognition
Scheduler
Control Plan
Agenda
Next Operation
Clinical Knowledge
Biological Knowledge
First Principles
Generic Reasoning Skills
Agenda Manager
Events
Executor
Communication Interface
Display Driver
Perceptual Preprocessor
Patient, Ventilator, Laboratory Simulation