The Challenges of Real-Time AI

David J. Musliner and **James A. Hendler** and **Ashok K. Agrawala**
Institute for Advanced Computer Studies
The University of Maryland
College Park, MD 20742
{musliner,hendler,agrawala}@cs.umd.edu

Edmund H. Durfee
Dept. of EE & Computer Science
The University of Michigan
Ann Arbor, MI 48109
durfee@eecs.umich.edu

Jay K. Strosnider and **C. J. Paul**
Dept. of Elec. & Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213
{jks,cjpaul}@ece.cmu.edu

U. Maryland Technical Report CS-TR-3290, UMIACS-TR-94-69
June, 1994
An abridged version of this paper was submitted to IEEE Computer.

1 Introduction

The research agendas of two major areas of computer science are converging: Artificial Intelligence (AI) methods are moving towards more realistic domains requiring real-time responses, and real-time systems are moving towards more complex applications requiring intelligent behavior. Together, they meet at the crossroads of interest in "real-time intelligent control," or "real-time AI"[1]. This subfield is still being defined by the common interests of researchers from both real-time and AI systems. As a result, the precise goals for various real-time AI systems are still in flux [28, 31, 43]. This paper describes an organizing conceptual structure for current real-time AI research, clarifying the different meanings this term has acquired for various researchers. Having identified the various goals of real-time AI research, we then specify some of the necessary steps towards reaching those goals. This in turn enables us to identify promising areas for future research in both AI and real-time systems techniques.

1.1 Background: Real-Time Systems

In many applications, a computer control system must sense the environment and directly influence it through action. Such control systems are subject to the real-time constraints of the environments in which they operate. For example, an autonomous vehicle operating in the real world needs a control system that responds quickly enough to avoid collisions with obstacles or

David Musliner and James Hendler are also affiliated with the UM Institute for Systems Research (NSF Grant NSFD CDR-88003012).

[1]These two terms are used interchangeably by many researchers. We will use the somewhat more common "real-time AI" for consistency and brevity.

other vehicles. This requirement for timely behavior is the defining characteristic of a class of environments known as *hard real-time* domains. Hard real-time domains have deadlines by which control responses must be produced, or catastrophic failure may occur. Other common examples of hard real-time domains include nuclear power plant control, medical monitoring, and aircraft control.

Because catastrophic failure may occur if deadlines are missed, control systems for agents operating in real-time environments must not only choose appropriate actions in varied situations, they must also make those action choices at appropriate times. Research in real-time systems addresses precisely this issue, by developing methods for *guaranteeing* that the reaction rate of a control system matches the rate of change in the environment. Real-time computing is *not* about building "fast" systems; it *is* about building systems that are predictably "fast enough" to act on their environments in well-specified ways [31, 43, 66, 69].

This understanding of what it means to be "real-time" is dramatically different from the casual, non-technical use of the term which has become common in many fields. For example, if a database querying system responds quickly according to human time-scales (i.e., in a few seconds or less), it is called "real-time." But what if we use that same database system in a critical application requiring responses in milliseconds? Clearly, the system is no longer "fast enough." The fact that the inadequacy of the system in this new domain (and its "adequacy" in the slower domain) could not be recognized or predicted in any rigorous fashion indicates that this system was never "real-time" in the technical sense; it was never known to meet the required deadlines.

Early real-time systems operated in relatively simple, well-characterized environments. Such "traditional" real-time systems are composed of a set of repeated tasks with known execution times and arrival patterns. The primary challenge in building such systems is to schedule these tasks and ensure they will meet their deadlines. Real-time systems researchers have developed a powerful set of tools for both specifying a task's resource requirements and deadlines and for predictably scheduling and executing the described behaviors to guarantee that they will meet their deadlines.

With the success of these techniques, researchers have been extending real-time systems to more complex applications. Faced with resource limitations that make it impossible to schedule and guarantee all of the possible behaviors that might ever be needed in a particular domain, it has become necessary for systems to *adapt* to the operational modes of the physical processes they monitor and control. For example, a real-time system controlling the avionics of an aircraft has to deal with different types of activities during the take-off, cruising, and landing stages. The control system's tasks and their execution characteristics will vary, depending on the particular operational mode of the aircraft. Lack of adaptability would require that the superset of all tasks required in all modes be active at all times. This is clearly an inefficient and infeasible approach— hence, real-time systems have been developed with limited abilities to change their task characteristics for different operational modes.

However, this type of simple mode-switching adaptability is insufficient for the emerging generation of complex, dynamic, and uncertain application domains. For example, consider the proposed

NASA Mars Rover, which must operate at a distance of about 15 light-minutes from Earth, and thus cannot be tele-operated. This system must operate continuously and autonomously in an uncertain and incompletely-specified environment. The system must detect and react in real-time to unpredictable but dangerous situations such as navigation route blockages and non-geometric terrain hazards (e.g., sand-pits or other potential traps that cannot be detected by sensors directly [29]). These situations require a broad range of adaptability and reasoning processes (e.g., route planning) that are beyond the abilities of conventional real-time systems. Fortunately, these topics (reasoning, adaptability, general and flexible intelligent behavior) have long been the purview of a different research area: Artificial Intelligence.

1.2 Background: AI

In general, AI research attempts to computationally model the various "intelligent" capabilities of humans, including their abilities to solve complex reasoning problems while also operating in and adapting to dynamic, uncertain, and incompletely-specified domains. Many of the problems addressed by traditional AI research can be viewed as search. In its most general sense, search is the process of finding an appropriate set of operators (actions) to lead an agent from some initial state to some goal state. Early AI research centered on basic search methods and representations for tasks involving the sorts of symbolic knowledge humans use, as opposed to the numeric data processed by simple algorithms. Much traditional and current AI research revolves around building powerful search-based planning mechanisms that can find useful plans of action in complex domains that may include goal interactions, uncertainty, and temporal information.

Traditionally, AI systems have been developed without much attention to the resource limitations that motivate real-time systems researchers. However, as these AI systems move out of the research laboratories into real-world applications, they also become subject to the time constraints of the environments in which they operate. For example, AI applications are now being used for monitoring space shuttle telemetry during launches to detect problems quickly when they develop. They are also used to assist operators of electrical power distribution centers. As the complexity of the controlled processes increases, the human operators of these systems are increasingly overloaded with data, particularly in time-critical fault recovery operations. Sorting through large amounts of data to identify problems and evaluate solutions can result in *cognitive overload*. The new generation of AI systems are being called upon to provide decision-support capabilities that reduce the cognitive burden in these situations.

Early efforts to build real-time AI systems focused largely on *ad hoc* speedups for existing AI methods, yielding systems that are only *coincidently real-time* [43]. That is, these systems have been tested and shown to operate quickly enough to meet domain deadlines for the test scenarios, and are thus considered guaranteed real-time. However, complex behavior interactions and domain variations beyond the set of tested scenarios may still lead to failure, because there is no rigorous proof that these systems will meet deadlines or provide logically adequate responses in time. As Stankovic [69] points out, this type of performance is not suited to *mission-critical* real-time systems[2]. Instead, the rigorous design techniques developed by real-time systems researchers must be used to *guarantee* that a system will meet domain deadlines, even in worst-case scenarios.

Thus, the technologies of real-time system design and AI are being brought together by the needs of real applications. We would like to combine the guaranteed performance methods of real-time systems with AI planning, problem-solving, and adaptation mechanisms to build a flexible, intelligent control system that can dynamically plan its own behaviors and guarantee that those behaviors will meet hard deadlines. Unfortunately, building such a system is not as easy as describing its behavior.

1.3 Real-Time AI: The Problem

In the most general sense, the difficulty of building real-time AI systems stems from constraints imposed by the agent, its tasks, and its environment. The agent itself is subject to "bounded rationality" [67], because its data processors have limited speed and memory. The agent is also subject to "bounded reactivity" [56], because its sensors and actuators are limited in their range, field of view, torque, accuracy, etc. Thus the agent has only a restricted capacity to sense its environment, process the sensed data, and use that information to affect its environment. Nevertheless, this limited agent is required to perform a wide variety of tasks ranging from mission-critical emergency reactions to complex knowledge-based problem-solving that may involve searching for solutions in exponentially-large search spaces. The domain itself may impose both severe real-time deadline constraints and complexities that result from incompletely-modeled dynamics, unpredictable agents, and failures. Putting all of these constraints together, we see that real-time AI systems must reliably accomplish complex and mission-critical tasks under realistic resource limitations in dynamic domains.

More specifically, real-time AI systems are required to work continuously over extended periods of time, interface to the external environment via sensors and actuators, deal with uncertain or missing data, focus resources on the most critical events, handle both synchronous and asynchronous events in a predictable fashion with guaranteed response times, and degrade gracefully. Graceful degradation may be required to cope with both resource overloads (in which case a system must alter its various performance goals or methods) and faults (in which case a system may have to execute recovery actions to resume its normal operations). Realizing such system behavior not only requires that significant research advances are made in both the real-time and AI methodologies, but also requires that techniques from the two disciplines be combined or interfaced. This paper outlines many of the specific research challenges that must be solved to develop future generations of real-time AI systems.

[2] In mission-critical systems, failure to take appropriate and timely action may lead to loss of life or unacceptably high economic costs.

1.4 An Example Domain

Throughout this paper, we will draw examples from the medical Intensive Care Unit (ICU) domain[3]. In an ICU, a patient is hooked to a number of monitoring and life support devices under electronic control. The goals of the system are obvious: keeping the patient healthy and stable, dealing with unexpected changes in the patient's condition, and alerting medical personnel if the situation becomes critical. With current technology, the individual devices connected to the patient are each separately controlled. If the patient goes into some type of trauma (e.g., shock), a number of alarms sound to summon medical personnel. During the time that it takes for the personnel to arrive, the patient's condition deteriorates. Even worse, upon entering the ICU room, medical personnel must waste precious time trying to figure out what has happened, why, and what to do about it.

An intelligent real-time control system could vastly improve this situation. A computer control system that could see the whole picture (integrating the results of the many sensor readings coming at various rates from a patient) could then initiate small actions (such as adjusting the feed rate on a respirator) which might prevent the patient from experiencing the trauma. In addition, if the patient does go into shock, the system can diagnose the cause and have diagnostic (and treatment) suggestions ready by the time medical personnel arrive. In cases where immediate steps must be taken, the system can initiate precursor actions such as reducing a particular gas in a ventilator, which should be done if emergency surgery may be required.

From this example domain, the need for combined real-time and AI capabilities becomes clear. Traditional real-time control systems could not handle the decision-making inherent in performing on-line diagnosis or treatment planning (which may also require access to symbolic information such as patient history, information about the surgery from which the patient is recovering, or the qualitative assessments made by medical personnel). On the other hand, a traditional AI-based medical diagnostic system could not handle the real-time needs of monitoring sensor data or of producing action plans during critical events (such as a drop in blood pressure). Thus the ICU domain captures the need for both intelligent, symbolic, deliberative AI techniques, and the need for guaranteed real-time reactions.

2 Categorizing Approaches to Real-Time AI

There are three principal ways that real-time systems and AI techniques can be combined into a single system [22, 56]: by embedding AI into a real-time system, by embedding real-time reactions into an AI system, and by coupling AI and real-time subsystems as parallel, cooperating components. Figure 1 illustrates these three system organizations. As we will see, these different approaches do not aim at the same result: the desired performance of a real-time AI system has not been clearly and uniquely defined. In the followings subsections, we provide additional details on the three main approaches to real-time AI, describing their overall performance goals and the

[3]These examples are based on the Guardian real-time AI project [34] under the direction of Dr. Barbara Hayes-Roth at Stanford University.

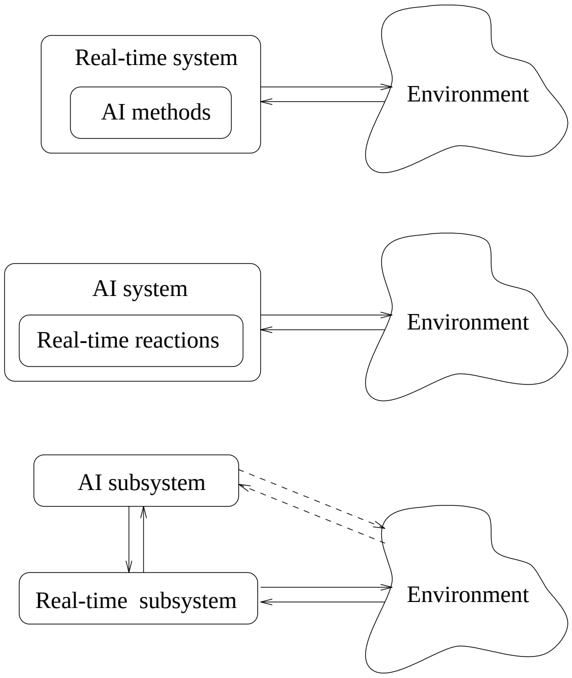

Figure 1: The three main approaches to real-time AI.

difficulties to be addressed.

Note that these categories are somewhat artificial, in that many implemented systems combine aspects from more than one approach. However, by classifying the general approach a system takes to implementing real-time AI, we can understand more clearly the precise goals and intent of the system, and its suitability for different types of tasks.

2.1 Embedding AI in Real-Time

The simplest approach to combining real-time and AI technologies is to embed AI methods within a conventional real-time system, forcing the AI computations to meet deadlines just like other real-time tasks. The goal, then, is to be "intelligent *in* real-time," so that all of the system's reasoning capacity is applied to each decision before its deadline. In many ways this performance goal is the most intuitive interpretation of "real-time AI," since the term itself seems to imply that the intelligent methods will run in real-time. Some researchers refer to this particular approach as "real-time problem solving," clearly indicating that complex, traditional AI tasks such as search-based problem-solving will be constrained to meet real-time deadlines.

The fundamental problem with this approach is that AI tasks are generally ill-suited to real-time scheduling mechanisms, which rely on allocating the worst-case execution time for all tasks. AI tasks such as planning and search-based problem-solving often have unknown or extremely large worst-case execution times [58]. While conventional real-time tasks often have small execution time variations due to data dependency, many AI tasks have additional variations due to search and backtracking. As a result, simply applying conventional worst-case scheduling methods to AI tasks results in systems which are either not schedulable, or have very low utilization.

There are two general methods for addressing this conflict between high-variance AI tasks and real-time schedulers: the variance of the AI tasks may be reduced, or the AI tasks may be cast as incremental, interruptible algorithms. Systems that address the conflict and are able to use the embedded-AI approach include:

- Deliberative AI architectures such as PRS [30, 37], in which the AI search mechanisms may be bounded so that the overall response-time of the system can be predicted. Approximation techniques and multiple problem-solving methods [14, 46] may be used to provide additional flexibility within time bounds.
- Purely reactive AI architectures like the subsumption architecture [5] and REX/Gapps [39], in which all of the reactive elements are assumed to run in real-time. Reactivity can be seen as the ultimate simplification or removal of AI search from planning tasks, and the variance of the tasks is removed with the search, making real-time guarantees feasible.
- Any-time algorithms and deliberation-scheduling systems [8], in which all AI tasks are cast as incremental methods that can be interrupted before any deadline, yielding a result that may have reduced precision, confidence, accuracy, etc.

2.2 Embedding Real-Time in AI

Embedding real-time capabilities into an AI system is an alternative approach, which essentially assumes that the overall system will employ typical AI search-based deliberation techniques, but that under some circumstances these techniques might be short-circuited in favor of a real-time reflexive action. This type of system is suited to domains in which deliberative action is the norm, and mission-critical real-time reactions will not be common. By retaining a unified system with a single locus of control, this embedded-real-time approach prioritizes real-time reactions without restricting the complexity of the AI methods that may be used. These systems are designed to operate in domains where they can rely mainly on deliberative processing for the intelligent selection of actions, occasionally overridden or preempted by real-time reflexes.

Implemented embedded-real-time systems have used several methods to achieve this behavior, including:

- Modified production systems such as Soar [45]. In Hero-Soar [44], distinguished real-time productions bypass the normal deliberative operator selection phase; instead, the action of a real-time production is executed as soon as the production is matched.
- Interruptible blackboard-style systems like PRS [30, 37] and RT-1 [17]. In response to a high-priority input that requires a response before a deadline, PRS may interrupt ongoing non-real-time AI tasks and switch to a predictable real-time task to generate the response[4].

[4] Note that we have now seen how PRS can operate both as an embedded-AI and embedded-real-time system; the architecture is flexible enough for either approach, and the choice is made when encoding procedural knowledge for the system.

There are several difficulties with the embedded-real-time approach, including the problem of effectively coordinating and mediating reflexive behaviors with the overall deliberative behavior of the system; if the reflexive actions can bypass the normal deliberation mechanisms, it may be difficult or impossible for the deliberation processing to reason about and affect the real-time reactions. Furthermore, some systems using this approach can add to their repertoire of reflexive actions based on inference or on experience (e.g., Soar's chunking [45]). The trouble with this approach is that the pattern-matching activity performed by the AI system against all of the possible cognitive and reflexive actions might be hard to bound[5]. As a result, even if the reflexive actions themselves have well-characterized real-time properties, they are invoked by high-variance, unpredictable AI techniques.

2.3 Cooperating Real-Time and AI

The third approach to real-time AI attempts to keep the strengths of real-time and AI systems undiluted by not mixing them directly, but instead allowing separate real-time and AI subsystems to cooperate in achieving overall desirable behaviors. The AI and real-time subsystems must be isolated, so that the AI mechanisms cannot interfere with the guaranteed operations of the real-time subsystem, but the subsystems must also be able to communicate and judiciously influence each other.

The performance goals of this type of cooperative system are considerably different from the embedded-AI approach, in which all of the AI processing was required to meet real-time deadlines. In the cooperative approach, the AI subsystem can be isolated from the real-time environment by the concurrent control behaviors of the real-time subsystem. Thus the goal of cooperative systems can be more broadly stated as being "intelligent *about* real-time," rather than necessarily being "intelligent *in* real-time."

The tasks that are executed on the real-time subsystem of a cooperative system may contain virtually any processing, as long as the tasks will definitely execute within their allocated time in the task schedule. Thus cooperative systems can take advantage of any advances in embedded-AI technology: when AI methods are built that can be embedded within real-time environments, those methods may also be used within the real-time subsystem of a cooperative real-time AI system. In other words, the real-time subsystem may execute complex AI methods, as long as they are predictable and scheduled.

Cooperative real-time AI systems span a wide range of designs, varying greatly in the complexity of the processing available on the respective AI and real-time subsystems, and in the precise relationship between these subsystems. At one extreme is a modified subsumption-based model such as DR/MARUTI [35] where a number of real-time reflexive behaviors execute concurrently, while higher-level AI processes adjust parameters that affect how the reflexive behaviors combine into outward action. This model essentially assumes that the system has sufficient resources to guarantee that every real-time reaction will meet its deadline, and that intelligence is primarily

[5] But see [18] for recent results indicating that, in some cases, such pattern matching may be scalable.

- Scheduler Feedback.
- Scheduling Iterative Tasks.
- Communication.
- Scenario Swapping (Mode Switching).
- Real-time Task Languages.
- Non-real-time Tasks.
- Enhanced Task Specifications.

Figure 2: Summary of real-time systems challenges.

useful for mediating amongst these sometimes-contradictory reactions (e.g., when one ICU reaction suggests increasing respirator flow, while a different reaction suggests the opposite).

However, as noted in Section 1.1, trying to scheduling the superset of all tasks required in all modes of behavior can lead to very inefficient or infeasible system designs. Thus, at the other extreme of coroutining, real-time AI systems such as CIRCA [56, 57] have an AI component that reasons about which real-time behaviors need to be carried out at any given time, and designs a real-time control plan appropriately. The real-time subsystem is only expected to schedule and execute a subset of all of its possible behaviors; as circumstances change, the AI system constructs alternative real-time reactive plans. The challenges in this approach include minimizing the real-time subsystem's reliance on receiving new schedules within hard deadlines, and developing rich world models that the AI system can use to reason about the real-time characteristics of a situation.

3 Challenges for Real-Time Systems

To date, real-time systems research has focused largely on developing low-level operating system mechanisms to support predictable execution of traditional periodic control tasks with only minor data dependencies. As more varied types of intelligent control tasks and real-time AI architectures are implemented, the role of system-level support for predictable hard real-time operations is expanding and changing. Figure 2 summarizes several focus areas we have identified as being of prime concern for supporting future mission-critical real-time AI systems. The following paragraphs provide additional details.

Scheduler Feedback

As always in real-time systems, scheduling is a paramount concern. However, the unusual nature of the tasks generated by real-time AI systems imposes special requirements and suggests several areas for future development. Most fundamentally, the inclusion of intelligent deliberation techniques in a real-time AI system raises the hope that AI methods based on decision theory might be used to control performance tradeoffs in the face of resource limitations. Such tradeoff techniques could make effective use of feedback information that an enhanced "cooperative" scheduler could provide. In the ICU domain, for example, the scheduler of a cooperative real-time AI system like CIRCA [56, 57] might indicate that a particular heart-monitoring task is the most severe constraint making a particular schedule infeasible. Using that feedback information, the system's planner might decide to decrease the required frequency of the monitoring task (and thus ease the scheduling constraints), if the patient is currently stable. However, if the planner deems the high-frequency monitoring task essential, it might instead choose to omit a less-important task from the schedule, such as a long-term data evaluation and diagnosis task.

Research on this type of advanced scheduling system could involve extending the view of scheduling as a search process that can generate explicit feedback, and/or integrating scheduling mechanisms with more traditional AI planning technologies (e.g., Miller's work on planning by search through simulations [54]). Search is a particularly apt metaphor for complex multi-processor scheduling tasks, so applying the AI community's experience with search methods may yield significant advances in this area.

Scheduling Iterative Tasks

Scheduling techniques that can utilize task performance profiles and integrate real-time and non-real-time tasks in a fair way are also of interest. The real-time systems counterpart to the any-time algorithm method, known as imprecise computation [47], has led to several advances in such scheduling technologies. Additional work will be required, however, to improve our ability to build performance profiles automatically and to enhance schedulers to manage tasks that are characterized by more complex representations such as conditional performance profiles [73] or quality functions.

In the ICU domain, for example, the utility of the results from an incremental treatment-planning task may vary greatly depending on the patient's condition: if the patient is having severe respiratory difficulty, even an abstract, incomplete treatment plan may have very high utility because it would begin by increasing respirator pressure and summoning medical personnel. On the other hand, if a patient has no emergency conditions, then taking action based on only partial planning would be ill-advised, and of low utility. Thus the overall utility of a particular partial plan can depend upon the context in which it is generated, and fixed performance profiles would not be sufficient to represent these dependencies. Conditional performance profiles, which can take into account such context dependencies, represent a first step towards addressing this representation problem.

The introduction of AI methods in real-time tasks also raises the possibility of tasks that provide periodic estimates of their remaining required execution time, which a scheduler could use to make dynamic adjustments to the task's resource allocation. These dynamic schedule changes could allow a system to adapt more effectively to uncertain, data-dependent search methods and changing environments. For example, a system might initially attempt to use a powerful model-based diagnosis method to plan treatment. However, if the model-based task initially evaluates the problem and indicates that it is expected to require several minutes to arrive at a treatment, the system might choose to first quickly retrieve a relevant treatment plan via case-based methods, in order to stabilize any emergency conditions.

Communication

The cooperative approach to real-time AI places the most stringent demands for extensions on existing real-time systems, because it involves dramatically new types of operations. For example, communication between real-time and non-real-time tasks is a major issue in the cooperative approach, while it is virtually nonexistent in embedded AI systems. In cooperative systems, the AI subsystem must be able to interact with the concurrently executing real-time control plan, either receiving feedback information (e.g., patient symptoms), downloading new plans (e.g., monitoring and treatment tasks), or communicating action arbitration information (e.g., which of several conflicting treatments should be used). The real-time support system must provide appropriate communication channels while also ensuring that those channels cannot interfere with ongoing guaranteed real-time tasks. For example, a system may be required to ensure that a real-time task will definitely receive an action arbitration message from a non-real-time AI task by a particular time, and it must be possible to provide that assurance to the system's AI methods despite ongoing schedule swaps and task variations.

Scenario Swapping (Mode Switching)

In the cooperative model of real-time AI, the AI subsystem generates a real-time task schedule that must then be rapidly swapped in for execution by the real-time subsystem, without violating any hard real-time deadlines. This capability requires the real-time system to support predictable-latency schedule substitutions. In combination with the precomputing concept described below for AI systems, it may also be necessary for the real-time subsystem to provide storage space for anticipated real-time schedules, and dynamic links between them. Thus, for example, an ICU task schedule used for monitoring a stable patient might itself trigger the action of swapping in a special schedule for treating shock when certain sensor readings are noted.

Support for this type of activity will require not only rapid schedule swaps, but the ability to reason about the temporal latencies that may arise during swaps, as well as the possibility of overlapping portions of task schedules during startup and shutdown phases of operation. In the previous example, the tasks used to treat shock might require an initializing process that loads data from the prior general-monitoring tasks. To accommodate that information transfer, tasks from both the general monitoring and special treatment schedules must be interleaved during the transition period between schedules.

Real-time Task Languages

Some cooperative real-time AI designs require the AI subsystem to download new real-time control tasks for predictable execution. Thus a task-specification language must be developed which is amenable both to manipulation by the AI subsystem (for planning) and to subsequent rapid execution by the real-time subsystem. This type of language may form a bridge between the knowledge and action representations common in AI planning systems and the more traditional systems-programming languages that real-time systems researchers have developed [40]. In the ICU domain, such a language would have to be capable of describing patient monitoring and treatment tasks, along with their projected effects on a patient (for planning) and their resource requirements (for scheduling). Providing both rigorous execution timing control and useful semantic flexibility will be a significant challenge. The work by Lyons *et al.* on the $\mathcal{RS}$ representation [49, 50] exemplifies a programming language model that combines aspects of real-time temporal specifications with AI-like planning models.

Enhanced Task Specifications

Because the computational tasks in a real-time AI system may be generated automatically, the planning system may be able to provide detailed information about those tasks to the scheduler. For example, an AI planner might be able to tell the scheduler about runtime dependencies and constraints between tasks, such as that a particular pair of tasks will never need to execute concurrently (i.e., they are mutually exclusive), and the scheduler might use this information when it is allocating tasks to processors. Or, the scheduler may be given timing information about a task other than just its worst-case execution time and required period (e.g., a probability distribution of arrival times for an event-driven task, or a maximum allowable invocation separation [55]). Advances in scheduling techniques that can take full advantage of this additional information should be able to schedule more tasks than simpler existing methods.

Non-real-time Tasks

The inclusion of non-real-time AI tasks in the same processing environment as guaranteed real-time tasks can introduce many complications relating to resource management, scheduling, and communications (as above). One primary concern with non-real-time AI tasks is that they are not usually cast as short, easily restartable computations. Instead, AI problem-solving methods may have very large runtimes and contain vast state information. For example, a model-based diagnosis method in the ICU domain could maintain a large open set of partially-hypothesized diagnoses as it searches for the best solution. Such tasks must be treated as ongoing, non-terminating tasks much like those found in traditional Unix-type operating systems. However, this capability is unusual for real-time operating systems, which may not even provide virtual memory.

4 Challenges for AI

Nearly all of the areas of current AI research could find applications in future real-time AI systems. However, we can pinpoint several areas of AI that may have significant impact on the development of real-time AI systems, because they are related to the fundamental operational constraints of these systems. Figure 3 summarizes the areas we have identified, while the following paragraphs provide descriptions and examples for each.

- Reduction of Search Variance.
- Incremental and Approximate Problem-solving.
- Customized Problem-solving.
- Precomputing.
- Representing, Planning, and Learning Reactions.
- Utility-based Modeling.
- Temporal Representation and Reasoning.
- Representation and Prediction of Processes.
- Concurrent Planning and Execution.
- Multi-agent Reasoning and Commitment.

Figure 3: Summary of AI challenges.

Reduction of Search Variance

Strosnider & Paul [70] have identified several techniques by which the variance inherent in search-based AI problem solving can be reduced, thus making problem-solving methods amenable to worst-case scheduling. These techniques include search-space pruning (removing portions of the space known to not contain a solution), search ordering (adjusting the order in which the search space is explored), and scoping (limiting the lookahead used to select operators). Other researchers are focusing on improved search algorithms such as RTA* [41]. If these techniques can be successfully applied to a particular domain's AI search problems, then the embedded-AI approach can provide real-time problem solving for the domain. In the ICU example, if we could use these techniques to manipulate the space of possible treatment plans such that the search for a treatment could be guaranteed to complete in a reasonably short amount of time, then the search could be scheduled and executed each time a change of patient status occurred.

Incremental and Approximate Problem-solving

Most current AI planning and problem-solving systems attempt to generate a complete and correct solution to a problem. In resource-bounded situations, it may be more effective to quickly generate an approximately correct solution. Incremental constraint-satisfaction techniques, iterative-deepening-style search strategies [41], dynamic programming [9], transformational planning [53], and approximate reasoning methods [14, 46] are currently being explored as ways to generate incremental and improving solutions. In the ICU system, these techniques could be used to quickly develop an approximate treatment plan (i.e. start to administer a dose of a particular drug) at the onset of an emergency, followed by improving this treatment as the patient stabilizes (i.e., decide to stop at 22.5 ccs).

Incremental improvement mechanisms allow an embedded-AI system to meet deadlines by simply interrupting the incremental algorithm when a deadline arrives. This approach, variously known as "any-time algorithms" [8], "imprecise computations" [47], "best-so-far methods" [70], and "increasing reward for increasing service" (IRIS) tasks [16], has gained even more proponents than names. However, significant advances are still necessary in characterizing task performance to develop the mappings between solution quality and service time ("performance profiles") that are needed to ensure that the interrupted algorithms will indeed produce results of acceptable quality by the time a deadline arrives. In addition, the process of composing larger systems from several incremental algorithms ("deliberation scheduling") is a growing area of research [4, 61], closely related to scheduling research in the real-time systems community.

Modifications to basic incremental methods may also lead to variants such as "dynamic expected runtime" tasks, which would occasionally re-evaluate their own expected runtimes and notify the real-time support system. Using this update information, an intelligent real-time control system could modify its task schedules and adjust its behavior to account for its current progress and expected future work.

Customized Problem-solving

A slightly different approach, under investigation by Garvey & Lesser [27], is to postulate a finite set of alternative computation methods for solving each particular problem a system might face. For example, the ICU controller might have complex causal model-based methods for analyzing sensor data and arriving at a diagnosis, as well a set of simpler rules that classify situations less accurately, but much more quickly. Using the "design-to-time" (DTT) methodology, selections are made from these alternative problem-solving techniques given *a priori* knowledge of the required resources and task deadlines. One significant weakness of this approach is the need for such *a priori* information. However, the discrete set of alternative methods used in DTT may be much more feasible to accurately characterize than the generalized incremental computations needed for deliberation scheduling. Research in the real-time systems community on "version selection" [51] and "load adjustment" [42] is closely related to DTT.

Precomputing

AI systems currently solve problems from scratch or use memory-based techniques to solve current problems based on past experience. A related technique that may improve real-time performance is to cache solutions for anticipated problems— that is, essentially creating a case memory based on reasoning about hypothetical situations, rather than on past practice [6]. This approach may include both compiling problem solutions and pre-prioritizing behavioral tradeoff decisions that may arise in the future. In the ICU domain, for example, the system might generate and store sample treatment plans for likely medical complications and trauma conditions, and then retrieve them for use in an emergency. As the plans take effect, they can then be tailored to the precise condition of the current patient.

Representing, Planning, and Learning Reactions

Reactive AI systems that perform little or no lookahead planning are well-suited to interacting with real-time domains because they avoid the high variance of planning and search activities. However, in order to be applicable to hard real-time domains, existing representations of reaction (e.g., RAPs [26], routines [1], and monitors [63]) must be extended to support predictable, provably-correct behavior. This requires enhanced capabilities to represent both the resource requirements of a reaction [56] and the functional behavior of a reaction [60].

In many types of real-time AI systems, intelligently adapting to dynamic environments requires the ability to create and use reactive plans that can immediately make necessary changes while a planner begins to determine longer-term actions. For planners that build reactive plans, significant areas of research include reaction planning mechanisms [39, 53, 56, 65] and formalisms which can support the proofs of guaranteed behavior that are necessary in hard real-time domains [40, 49, 50]. The inherent parallelism of reactive control systems is a particularly challenging aspect for AI planners, which have largely focused on sequential plan execution.

In addition, using learning methods to acquire reactions could provide improved reaction speed and coverage. Techniques that have proven useful in this area include compilation of reactive behaviors [64], "chunking" of previous action sequences [45], reinforcement learning [38, 59], and dynamic programming methods [3]. Such learning techniques allow a system to take necessary actions (e.g., increasing respirator rate) in emergencies (e.g., a decrease in blood pressure), if the action has been useful in the past.

Utility-based modeling

Given the dynamic and uncertain nature of real world domains (particularly as viewed through sensor readings), an intelligent control system must often make decisions based on incomplete information. To achieve optimal behavior according to some metric, the system might reason about probabilities and expected utilities [20, 25, 36, 72]. However, traditional decision theory often requires more information than is available in realistic domains. In these situations, a more qualitative model of uncertainty and decision-making is needed, including the ability to represent the growth of uncertainty as time passes. In the ICU case, such methods might lead to a decision to take certain actions based on their immediate beneficial effects (e.g., increasing respirator flow rate), even if the long term effects may be less clear (e.g., if left at too high a rate, this could eventually cause a respiratory problem).

Temporal Representation and Reasoning

Particularly in systems where all AI processing is not guaranteed to meet real-time deadlines, it becomes important for a system to have an explicit understanding of the temporal characteristics of its behaviors. In the ICU domain, for example, if administering a particular test and analyzing the resulting data takes several minutes, it may be important for the system to anticipate this delay and implement intermediary treatment actions, pending the results of the test. Current AI research on representations of time [2, 10, 32, 52, 71] generally involves complex constraint propagation mechanisms to maintain partial ordering relations between time intervals. In many cases, it may be possible to use simpler, more efficient temporal representations that can still provide the types of worst-case timing information required for real-time guarantees [57].

Representation and Prediction of Processes

In a medical diagnostic knowledge-based system, the inputs used in making the diagnosis are typically symptoms observed at a particular time. In contrast, in the ICU example the symptoms include a recognition of how the patient's health is changing over time, in response to ongoing physical processes and actions taken by the system. Being able to represent and reason about processes is crucial to predicting the possible future states of the patient and what to do about them. In addition, predicting how long a process should take (e.g., this IV should take about 20 minutes to drain) and comparing this to the actual behavior (e.g., after 10 minutes it is still 90% full) can be an important aspect of intelligent behavior and fault detection (e.g., the IV is not inserted correctly). This type of performance has been initially investigated by Cohen *et al.* [7, 33] using "envelopes" describing projected plan execution. Other techniques expected to be of use include extending planning systems to handle processes using quantitative [11, 62] and qualitative representations [15, 19], and modifying Bayesian techniques to cope with change over time.

Concurrent Planning and Execution

In the cooperative approach to real-time AI, the ability to concurrently plan and execute actions is crucial to successful performance [56, 65]. It is particularly useful to be able to begin executing plans before they can be completely fleshed out. For example, if the ICU patient is starting to enter a respiratory failure, the control system should immediately begin to increase oxygen flow, even before completely determining what other actions will be taken later. Techniques useful in enabling this sort of behavior include the use of iterative planning models (which can rapidly identify promising steps towards goals) [48, 53] and decision-theoretic control techniques (which can identify the most important steps to take).

Multi-agent Reasoning and Commitment

In many cases, deadlines and time constraints stem from commitments between agents. The ICU, for example, makes commitments to a patient and his or her family to monitor the patient's signs and intervene (to the extent possible) quickly enough to prevent the patient's condition from degrading significantly. It is this commitment that places such severe time constraints on automated diagnostic systems in the ICU. The same systems, if employed in a morbidity/mortality study, would likely be able to take more time to explain the progression of symptoms and outcomes. Even then, however, the systems must meet (possibly implicit) time commitments to provide results in time for a hearing, or publication.

While in some domains an agent will have little flexibility over the deadlines placed on a supporting knowledge-based system (e.g., the ICU patient), in other cases deadlines can be much more negotiable. It is important to understand how deadlines arise due to commitments between agents [68], how different types of relationships between the tasks at different agents influence the severity and flexibility of time commitments [12, 13], how commitments and goals can be revised, relaxed, and reassigned as unexpected events change what can be done when [6, 21, 23], and how abstraction can be used to generate the right degree of commitment under various circumstances [24].

5 Summary

Extending previous work [22, 56], we have identified and described three fundamental approaches to developing real-time AI systems. This paper improves upon the previous classification by distinguishing the various approaches according to their performance goals, in addition to their architectural organization of real-time and AI methods. With the field of real-time AI still in flux, choosing an appropriate system architecture for a particular application requires a clear description of the system's performance goals and design features. We hope that our classification of existing approaches to real-time AI will serve as a starting point for such descriptions.

By examining the functional constraints that these different architectures place on their components, we have pinpointed several challenging research areas that are critical to the development of next-generation real-time AI systems. As more complex, intelligent programs are applied to controlling systems in mission-critical domains, the need for more powerful and flexible real-time systems support technology is growing rapidly. In Section 3 we outlined several areas in systems support that will be important for the short-term development of next-generation intelligent real-time control systems. Likewise, we have found that the broad application of AI methods to real-time domains will require new approaches, differing from many of the traditional search-based techniques explored in the field. As discussed in Section 4, reactivity, incremental algorithms, decision-theoretic tradeoff methods, and other non-traditional techniques will be required to intelligently meet the demands of mission-critical environments. This paper may thus serve, in part, as a guideline for future research aimed at advancing the state of the art.

Existing systems represent a strong initial effort, but do not yet provide the comprehensive coordination of real-time reactivity and intelligent, deliberative behavior implied by the full sense of "real-time AI." The embedded-AI and embedded-real-time approaches place strong constraints on either the AI or real-time components, making them applicable to only limited domains. The cooperative approach to real-time AI removes some of those constraints, but current implementations remain largely master-slave systems in which the real-time subsystem is enslaved, simply executing plans sent from the AI subsystem. In the long term, we expect that the merging of real-time and AI research will lead to systems in which task planning, scheduling, and execution are more integrated, yet still retain the guaranteed performance characteristics required for mission-critical operations. Precisely how this will be accomplished is, of course, the subject of future work. However, it is clear that advances the areas we have pinpointed will lead us closer to the overall goal of building a flexible but predictable goal-directed mechanism capable of recognizing and adjusting to environmental dynamics and its own resource bounds.

Acknowledgements

The authors would like to acknowledge useful input from the many participants in the NSF-sponsored workshop on "Artificial Intelligence and Real-time" (Grant IRI-9216094) the Panel on Resource Bounded Systems at the June 1993 Workshop on Design Principles and Engineering of Knowledge Based Systems (sponsored by the Joint Directors of Laboratories), and contributors to a report being collated by Kang Shin[6]. Researchers involved in these efforts include: Ron Arkin, Mark Boddy, Chris Brown, B. Chandrasekaran, Su-Shing Chen, Bruce D'Ambrosio, Susan Davidson, Tom Dean, David Etherington, Oscar Firschein, Northrup Fowler, Alan Garvey, Rich Gerber, Matt Ginsberg, Moises Goldszmidt, Barbara Hayes-Roth, Connie Heitmeyer, Leslie Interrante, Farnam Jahanian, Paris Kanellakis, Bob Kohout, Jean-Louis Lassez, Insup Lee, Kwei-Jay Lin, Jane Liu, Doug Locke, Nancy Lynch, David Miller, Al Mok, Howard Moraff, Ragunathan Rajkumar, Krithi Ramamritham, Daniel Rochowiak, Paul Rosenbloom, Stan Rosenschein, Fred Schneider, Marcel Schoppers, Karsten Schwan, Liu Sha, Kang Shin, Sang Son, Jack Stankovic, Alex Stoyenko, Hide Tokuda, Abe Waksman, and Wei Zhao.

References

[1] P. E. Agre and D. Chapman, "Pengi: An Implementation of a Theory of Activity," in *Proc. National Conf. on Artificial Intelligence*, pp. 268–272. Morgan Kaufmann, 1987.

[2] J. F. Allen, "Maintaining Knowledge about Temporal Intervals," *Communications of the ACM*, vol. 26, no. 11, pp. 832–843, 1983.

[3] A. G. Barto, S. J. Bradtke, and S. P. Singh, "Real-time Learning and Control using Asynchronous Dynamic Programming," submitted to *Artificial Intelligence*, 1994.

[4] M. Boddy and T. Dean, "Solving Time-Dependent Planning Problems," in *Proc. Int'l Joint Conf. on Artificial Intelligence*, pp. 979–984, August 1989.

[5] R. A. Brooks, "A Robust Layered Control System for a Mobile Robot," *IEEE Journal of Robotics and Automation*, vol. RA-2, no. 1, pp. 14–22, March 1986.

[6] B. Chandrasekaran, R. Bhatnagar, and D. D. Sharma, "Real-Time Disturbance Control," *Communications of the ACM*, vol. 34, no. 8, pp. 33–47, August 1991.

[7] P. R. Cohen, M. L. Greenberg, D. M. Hart, and A. E. Howe, "Trial by Fire: Understanding the Design Requirements for Agents in Complex Environments," *AI Magazine*, vol. 10, no. 3, pp. 33–48, Fall 1989.

[8] T. Dean and M. Boddy, "An Analysis of Time-Dependent Planning," in *Proc. National Conf. on Artificial Intelligence*, pp. 49–54, 1988.

[9] T. Dean, L. P. Kaelbling, J. Kirman, and A. Nicholson, "Planning With Deadlines in Stochastic Domains," in *Proc. National Conf. on Artificial Intelligence*, pp. 574–579, 1993.

[6] To appear as a University of Michigan technical report.

[10] T. L. Dean, "Intractability and Time-Dependent Planning," in *Proceedings of the 1986 Workshop on Reasoning about Actions & Plans*, pp. 245–266. Morgan Kaufmann Publishers Inc., 1987.

[11] T. L. Dean and M. P. Wellman, *Planning and Control*, Morgan Kaufmann Publishers, 1991.

[12] K. S. Decker and V. R. Lesser, "Quantitative Modeling of Complex Environments," *Int'l J. of Intelligent Systems in Accounting, Finance, and Management*, vol. 2, no. 4, , December 1993.

[13] K. S. Decker and V. R. Lesser, "Designing a Family of Coordination Algorithms," Technical Report 94–14, University of Massachusetts, Department of Computer Science, 1994.

[14] K. S. Decker, V. R. Lesser, and R. C. Whitehair, "Extending a Blackboard Architecture for Approximate Processing," *Journal of Real-Time Systems*, vol. 2, no. 1/2, pp. 47–79, May 1990.

[15] G. F. DeJong, "Learning to Plan in Continuous Domains," *Artificial Intelligence*, vol. 65, no. 1, , January 1994.

[16] J. K. Dey, J. F. Kurose, and D. Towsley, "Efficient On-line Processor Scheduling for a Class of IRIS (Increasing Reward with Increasing Service) Real-Time Tasks," *Performance Evaluation Review*, vol. 21, no. 1, , June 1993.

[17] R. Dodhiawala, N. S. Sridharan, P. Raulefs, and C. Pickering, "Real-Time AI Systems: A Definition and An Architecture," in *Proc. Int'l Joint Conf. on Artificial Intelligence*, pp. 256–261, August 1989.

[18] R. B. Doorenbos, "Matching 100,000 Learned Rules," in *Proc. National Conf. on Artificial Intelligence*, pp. 290–296, 1993.

[19] B. Drabble, "EXCALIBUR: A Program for Planning and Reasoning with Processes," *Artificial Intelligence*, vol. 62, no. 1, , July 1993.

[20] D. Draper, S. Hanks, and D. Weld, "Probabilistic Planning with Information Gathering and Contingent Execution," Technical Report 93–12–04, University of Washington, December 1993.

[21] E. H. Durfee and V. R. Lesser, "Incremental Planning to Control a Time-Constrained, Blackboard-Based Problem Solver," *IEEE Trans. Aerospace and Electronic Systems*, vol. 24, no. 5, pp. 647–662, 1988.

[22] E. H. Durfee, "A Cooperative Approach to Planning for Real-Time Control," in *Proc. Workshop on Innovative Approaches to Planning, Scheduling and Control*, pp. 277–283, November 1990.

[23] E. H. Durfee and V. R. Lesser, "Partial Global Planning: A Coordination Framework for Distributed Hypothesis Formation," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 21, no. 5, pp. 1167–1183, 1991.

[24] E. H. Durfee and T. A. Montgomery, "Coordination as Distributed Search in a Hierarchical Behavior Space," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 21, no. 6, pp. 1363–1378, 1991.

[25] O. Etzione, “Embedding Decision-Analytic Control in a Learning Architecture,” *Artificial Intelligence*, vol. 49, pp. 129–159, 1991.

[26] R. J. Firby, “An Investigation into Reactive Planning in Complex Domains,” in *Proc. National Conf. on Artificial Intelligence*, pp. 202–206, 1987.

[27] A. Garvey and V. Lesser, “Design-to-time Real-Time Scheduling,” *IEEE Trans. Systems, Man, and Cybernetics*, vol. 23, no. 6, pp. 1491–1502, 1993.

[28] A. Garvey and V. Lesser, “A Survey of Research in Deliberative Real-Time Artificial Intelligence,” *Journal of Real-Time Systems*, vol. 6, no. 3, , May 1994.

[29] E. Gat, M. G. Slack, D. P. Miller, and R. J. Firby, “Path Planning and Execution Monitoring for a Planetary Rover,” in *Proc. IEEE Int'l Conf. on Robotics and Automation*, pp. 20–25, 1990.

[30] M. P. Georgeff and F. F. Ingrand, “Decision-Making in an Embedded Reasoning System,” in *Proc. Int'l Joint Conf. on Artificial Intelligence*, pp. 972–978, August 1989.

[31] J. R. Greenwood and J. P. Marsh, “Real-Time AI: Software Architecture Issues,” Technical report, Advanced Decision Systems, May 1987.

[32] S. Hanks, “Practical Temporal Projection,” in *Proc. National Conf. on Artificial Intelligence*, 1990.

[33] D. M. Hart, S. D. Anderson, and P. R. Cohen, “Envelopes as a Vehicle for Improving the Efficiency of Plan Execution,” in *Proc. Workshop on Innovative Approaches to Planning, Scheduling and Control*, pp. 71–76, November 1990.

[34] B. Hayes-Roth, “An Architecture for Adaptive Intelligent Systems,” accepted for publication in *Artificial Intelligence, Special Issue on Agents and Interactivity*, 1994.

[35] J. Hendler and A. Agrawala, “Mission Critical Planning: AI on the MARUTI Real-Time Operating System,” in *Proc. Workshop on Innovative Approaches to Planning, Scheduling and Control*, pp. 77–84, November 1990.

[36] A. Howe and P. Cohen, “Failure Recovery: A Model and Experiments.,” in *Proc. National Conf. on Artificial Intelligence*, pp. 801–808, 1991.

[37] F. F. Ingrand and M. P. Georgeff, “Managing Deliberation and Reasoning in Real-Time AI Systems,” in *Proc. Workshop on Innovative Approaches to Planning, Scheduling and Control*, pp. 284–291, November 1990.

[38] L. P. Kaelbling, *Learning in Embedded Systems*, MIT Press, 1994.

[39] L. P. Kaelbling and S. J. Rosenschein, “Action and Planning in Embedded Agents,” in *Robotics and Autonomous Systems 6*, pp. 35–48, 1990.

[40] R. C. Kohout, D. J. Musliner, and J. A. Hendler, “Grounding Dynamic Reaction on the Maruti Operating System,” Technical Report CS-TR-3231, University of Maryland Department of Computer Science, April 1994.

[41] R. E. Korf, "Depth-Limited Search for Real-Time Problem-Solving," *Journal of Real-Time Systems*, vol. 2, no. 1/2, pp. 7–24, May 1990.

[42] T.-W. Kuo and A. K. Mok, "Load Adjustment in Adaptive Real-Time Systems," in *Proc. Real-Time Systems Symposium*, pp. 160–170, December 1991.

[43] T. J. Laffey, P. A. Cox, J. L. Schmidt, S. M. Kao, and J. Y. Read, "Real-Time Knowledge-Based Systems," *AI Magazine*, vol. 9, no. 1, pp. 27–45, 1988.

[44] J. E. Laird and P. S. Rosenbloom, "Integrating Execution, Planning, and Learning in Soar for External Environments," in *Proc. National Conf. on Artificial Intelligence*, July 1990.

[45] J. E. Laird, A. Newell, and P. S. Rosenbloom, "SOAR: An Architecture for General Intelligence," *Artificial Intelligence*, vol. 33, pp. 1–64, 1987.

[46] V. R. Lesser, J. Pavlin, and E. Durfee, "Approximate Processing in Real-Time Problem Solving," *AI Magazine*, vol. 9, no. 1, pp. 49–61, 1988.

[47] J. W.-S. Liu, K.-J. Lin, and S. Natarajan, "Scheduling Real-Time, Periodic Jobs Using Imprecise Results," in *Proc. Real-Time Systems Symposium*, pp. 252–260, December 1987.

[48] D. M. Lyons, A. J. Hendriks, and S. Mehta, "Achieving Robustness by Casting Planning as Adaptation of a Reactive System," in *Proc. IEEE Int'l Conf. on Robotics and Automation*, pp. 198–203, April 1991.

[49] D. M. Lyons, "A Process-Based Approach to Task Plan Representation," in *Proc. IEEE Int'l Conf. on Robotics and Automation*, pp. 2142–2147, 1990.

[50] D. M. Lyons, S. Mehta, and P. S. Gopinath, "Robust Representation and Execution of Robot Plans," in *Proc. IEEE EUROMICRO Workshop on Real-Time*, pp. 34–41, June 1990.

[51] N. Malcolm and W. Zhao, "Version Selection Schemes for Hard Real-Time Communications," in *Proc. Real-Time Systems Symposium*, pp. 12–21, December 1991.

[52] D. McDermott, "A Temporal Logic For Reasoning About Processes and Plans," *Cognitive Science*, vol. 6, pp. 101–155, 1982.

[53] D. McDermott, "Transformational Planning of Reactive Behavior," Technical Report 941, Yale University Department of Computer Science, December 1992.

[54] D. P. Miller, *Planning by Search Through Simulations*, PhD thesis, Yale University, 1985.

[55] D. J. Musliner, "Scheduling Automatically-Generated Real-Time Monitoring Tasks," submitted to *Intelligent Systems Engineering: The International Journal of AI in Engineering*, 1994.

[56] D. J. Musliner, E. H. Durfee, and K. G. Shin, "CIRCA: A Cooperative Intelligent Real-Time Control Architecture," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 23, no. 6, pp. 1561–1574, 1993.

[57] D. J. Musliner, E. H. Durfee, and K. G. Shin, "World Modeling for the Dynamic Construction of Real-Time Control Plans," to appear in *Artificial Intelligence*, 1994.

[58] C. J. Paul, A. Acharya, B. Black, and J. K. Strosnider, “Reducing Problem-Solving Variance to Improve Predictability,” *Communications of the ACM*, vol. 34, no. 8, pp. 81–93, August 1991.

[59] A. Ram and J. C. Santamaría, “Multistrategy Learning in Reactive Control Systems for Autonomous Robotic Navigation,” *Informatica*, vol. 17, no. 4, pp. 347–369, 1993.

[60] S. J. Rosenschein and L. P. Kaelbling, “The Synthesis of Digital Machines with Provable Epistemic Properties,” in *Proc. Conf. Theoretical Aspects of Reasoning About Knowledge*, pp. 83–98, 1986.

[61] S. J. Russell and S. Zilberstein, “Composing Real-Time Systems,” in *Proc. Int'l Joint Conf. on Artificial Intelligence*, pp. 212–217, August 1991.

[62] R. M. Salter, “Planning in a Continuous Domain— An Introduction,” *Robotica*, vol. 1, pp. 85–93, 1983.

[63] J. C. Sanborn and J. A. Hendler, “A Model of Reaction for Planning in Dynamic Environments,” *Int'l Journal for Artificial Intelligence in Engineering*, vol. 3, no. 2, pp. 95–102, April 1988.

[64] M. Schoppers, “Universal Plans for Reactive Robots in Unpredictable Environments,” in *Proc. Int'l Joint Conf. on Artificial Intelligence*, pp. 1039–1046, 1987.

[65] M. Schoppers, “A Software Architecture for Hard Real-Time Execution of Automatically Synthesized Plans or Control Laws,” in *Proc. AIAA/NASA Conf. on Intelligent Robots in Field, Factory, Service, and Space*, pp. 768–775, March 1994.

[66] K. G. Shin and P. Ramanathan, “Real-Time Computing: A New Discipline of Computer Science and Engineering,” *Proceedings of the IEEE*, vol. 82, no. 1, pp. 6–24, January 1994.

[67] H. A. Simon, *Models of Bounded Rationality*, MIT Press, 1982.

[68] R. Smith, “The Contract Net Protocol: High-level Communication and Control in a Distributed Problem Solver,” *IEEE Trans. Computers*, vol. 29, , 1980.

[69] J. A. Stankovic, “Misconceptions about Real-Time Computing: A Serious Problem for Next-Generation Systems,” *IEEE Computer*, vol. 21, no. 10, pp. 10–19, October 1988.

[70] J. K. Strosnider and C. J. Paul, “A Structured View of Real-Time Problem Solving,” accepted for publication in *AI Magazine*, 1994.

[71] S. Vere, “Temporal Scope of Assertions and Window Cutoff,” in *Proc. Int'l Joint Conf. on Artificial Intelligence*, pp. 1055–1059, 1985.

[72] M. P. Wellman, “The STRIPS Assumption for Planning with Uncertainty,” in *Proc. AAAI Spring Symp. on Planning in Uncertain, Unpredictable, or Changing Environments*, 1990.

[73] S. Zilberstein and S. J. Russell, “Constructing Utility-Driven Real-Time Systems Using Anytime Algorithms,” in *Proc. IEEE Workshop on Imprecise and Approximate Computation*, pp. 6–10, December 1992.