Reasoning about Bounded Reactivity to Achieve Real-Time Guarantees *

David J. Musliner **Edmund H. Durfee** **Kang G. Shin**

Department of Electrical Engineering and Computer Science
The University of Michigan
Ann Arbor, Michigan 48109-2122
E-mail: djm@eecs.umich.edu

Introduction

Realistic intelligent control systems must recognize that, in addition to the processor limitations that make them subject to "bounded rationality" [Simon, 1982], sensor and actuator limitations also lead to "*bounded reactivity*." As part of our larger goal of developing intelligent real-time control systems [Musliner *et al.*, 1992], we are investigating techniques that allow a system to take its own bounded reactivity into account by *selectively* perceiving and reacting to conditions in the environment. These techniques include encapsulating sensing and action primitives within reactive structures that facilitate reasoning about resource constraints, and an environment modeling technique that allows the system to automatically derive the selected perceptions and reactions it must activate at any particular time.

Most research on "real-time AI" focuses either on restricted AI techniques that have predictable performance characteristics [Boddy and Dean, 1989] or on reactive systems that retain little of the power of traditional AI [Brooks, 1986]. Our research takes a different approach: to achieve a combination of unrestricted AI techniques with the ability to make hard performance guarantees, we propose CIRCA, a Cooperative Intelligent Real-time Control Architecture. Figure 1 illustrates the architecture, in which an AI subsystem (AIS) reasons about task-level problems that require its powerful but uncertain reasoning methods, while a separate real-time subsystem (RTS) uses its predictable performance characteristics to deal with control-level problems that require guaranteed response times. We have developed a scheduling module and a structured interface that allow the unconstrained AI subsystem to asynchronously direct the real-time subsystem without violating any response-time guarantees.

*The work reported in this paper was supported in part by the National Science Foundation under Grants DMC-8721492 and IRI-9158473, and by a NSF Graduate Fellowship. The opinions, findings, and recommendations expressed in this publication are those of the authors, and do not necessarily reflect the views of the NSF.

This paper focuses on the techniques we are developing for meeting performance goals with limited reactive resources, including sensors and actuators. Selective perception plays a key role in these techniques, restricting the conditions which the system senses, processes, and reacts to. We will discuss these techniques in the context of our example domain, in which a prototype implementation of CIRCA pilots a Hero 2000 robot through hallways using an aimed sonar sensor. We first present an overview of the architecture, including a description of our representation for reactive behaviors. We then discuss the environment model which the AIS uses to reason about the system's performance and to make tradeoffs among various performance dimensions, including sensory attention.

Architecture Overview

CIRCA meets the demands of real-time control within a bounded-reactivity system by guaranteeing that it will produce a precise, high confidence response in a timely fashion *to a limited set of inputs.* In other words, the architecture can sacrifice completeness of attention in order to achieve precision, confidence, and timeliness in its responses to environmental changes that it does observe.

The Real-Time Subsystem (RTS)

The RTS executes a cyclic schedule of simple test-action pairs (TAPs[1]) which have known worst-case execution times. Since the RTS performs no other functions, it can guarantee that the scheduled tests and actions are performed within predictable time bounds. The TAP structure provides a standard primitive with which the Scheduler and AIS can reason about the timing and resource characteristics of the RTS' behavior, in order to guarantee meeting deadlines and resource restrictions. Each TAP class has a fixed set of tests (or preconditions), a set of actions to take if all the tests return

[1] Not to be confused with Firby's RAPs [Firby, 1987], which are larger in scale and do not have predictable execution times.

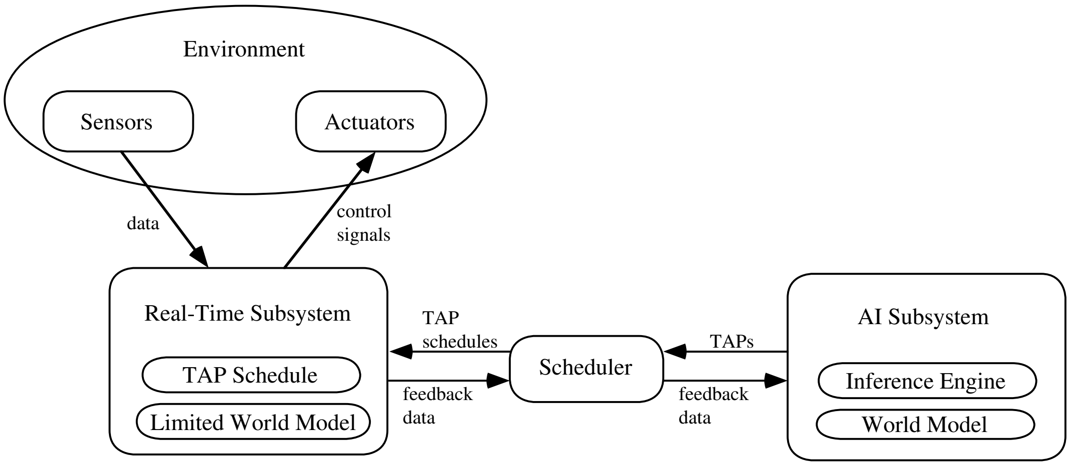

Figure 1: The Cooperative Intelligent Real-Time Control Architecture.

true, data about the sensing and actuating resources the TAP requires, and worst-case timing data on how long it takes to test the preconditions and execute the actions.

Specific TAP instances also have parameters that the AIS can set depending on the context in which the TAP is used. These parameters include frequency requirements or deadlines, which constrain the times at which a TAP must be invoked. TAP tests and actions may include acquiring sensor readings, performing limited data processing, and controlling the system actuators. Choosing TAPs and their frequencies implements selective perception and reaction by determining what environmental features the RTS is attending to, and what reactive behaviors it is supporting.

For example, Figure 2 shows an instance of the simple `stop-if-object-ahead` TAP class used in the hallway-following task. The TEST specifies that the TAP should only be executed if the robot is moving and the distance ahead of the robot, as sensed by sonar, is less than `*safety-distance*`. Thus the TAP explicitly focuses the sonar sensor on a selected condition, rather than trying to maintain a complete world model. If the tests return true, the robot is halted and the RTS sends a notification message to the AIS. Testing and executing this TAP takes a maximum of .13 seconds (TEST-TIME + ACTION-TIME), and the AIS has determined that it must be run at least every 1.5 seconds (MAX-PERIOD) to guarantee avoiding collisions with objects in front of the robot. Note that the frequency with which this TAP must be executed is dependent on the speed of the robot's motion: the AIS reasons about this and other variables to produce the parameters such as MAX-PERIOD. This is another example of selective perception, in which the system chooses to sense a condition at periodic intervals, rather than continuously.

The AI Subsystem (AIS) and Scheduler

CIRCA reasons about its bounded reactivity within the AIS and the Scheduler, which cooperate to decide which responses the RTS can and should guarantee. The AIS reasons about its goals and a model of the environment, and suggests sets of TAPs to the Scheduler, which attempts to build a TAP schedule. The Scheduler reasons about the maximum periods of the TAPs, their worst-case execution time and resource needs, and the resources available from the RTS. The Scheduler returns either a successful schedule that meets all the constraints, or some informative feedback if it fails to produce such a schedule. In that case, the AIS will modify the suggested set of TAPs, possibly by altering timing parameters, by choosing alternate TAPs to produce a desired behavior, or by actually dropping some TAPs altogether. In this way, the AIS and Scheduler reason about the real-time subsystem's bounded reactivity, and choose how to degrade performance to meet those limitations.

We have developed a graph model that represents the AIS' reasoning about the environment and the RTS behaviors. The graph model is the basis for CIRCA's selective perception: the AIS chooses TAPs to monitor conditions that the graph model indicates are critical to the system's goals [Musliner *et al.*, 1991]. Furthermore, the model shows how a subset of all reactions can isolate the guaranteed control level from the unguaranteed task level, so that the unpredictable AIS does not cause the RTS to violate hard deadlines.

The Graph Model of RTS/Environment Interaction

The directed graph model represents the *worst-case* behavior of the environment, and the actions which the RTS can take to avoid failure. The graph model has five elements (S, F, T_E, T_A, T_T):

```
TAP stop-if-object-ahead
    TEST: (and *moving* (< (get-sonar-reading-ahead) *safety-distance*))
    ACTION: (progn (halt) (notify-AIS 'halted))
    RESOURCES: (sonar base-motors)
    MAX-PERIOD: 1.5
    TEST-TIME: .1
    ACTION-TIME: .03
```

Figure 2: A `stop-if-object-ahead` TAP.

1. A finite set of "states" $S = \{S_1, S_2, ..., S_m\}$, where each state S_i represents a description of relevant features of the world.
2. A distinguished failure state F, which subsumes all states that violate domain constraints or control-level goals (e.g., system survival). The system strives to avoid the failure state.
3. A finite set of "event transitions" $T_E = \{T_{E1}, T_{E2}, ..., T_{En}\}$, that represent world occurrences as instantaneous state changes.
4. A finite set of "action transitions" $T_A = \{T_{A1}, T_{A2}, ..., T_{Ap}\}$, that represent actions performed by the RTS.
5. A finite set of "time transitions" $T_T = \{T_{T1}, T_{T2}, ..., T_{Tq}\}$, that represent the progression of time. We represent only the significant time transitions which lead to state changes.

Each transition $T_i \in T = T_E \cup T_A \cup T_T$ is a mapping between states; $T_i : S \rightarrow S$. The functions $D : T \rightarrow S$ and $R : T \rightarrow S$ determine the domain and range of a transition; $T_i : D(T_i) \rightarrow R(T_i)$.

An "event-closed" set of states $S_{EC} \subseteq S$ is defined as a connected set of states for which every event transition from every state in the set leads to a state that is also in the set. That is, $\forall T_{Ei} \in T_E \mid D(T_{Ei}) \notin S_{EC} \vee R(T_{Ei}) \in S_{EC}$. In other words, non-temporal events (as opposed to the mere progression of time) cannot move the system out of the event-closed set of states. An event-closed set of states that does not contain the failure state is called a "safe" set of states ($F \notin S_{safe}$). Note that a safe set of states can still lead to failure through time transitions (i.e., it is possible that $\exists T_{Ti} \in T_T \mid D(T_{Ti}) \in S_{safe} \wedge R(T_{Ti}) = F$). These time transitions to failure correspond exactly to violating the hard real-time domain constraints: if the system enters a new state because of an event, but fails to react to the new state before a hard deadline, it will have entered the failure state via a time transition: that is, by "waiting too long" to react, the system fails.

Figure 3 shows a portion of the graph model for the hallway-following task. Each labeled box in the graph model represents a state S_i. Solid single arrows represent event transitions T_{Ei}, dashed single arrows represent action transitions T_{Ai}, and double arrows represent time transitions T_{Ti}. For simplicity, the example shows only transitions which have a single possible outcome. In general, environmental uncertainty may allow transitions to lead to more than one possible new state. Such straightforward extensions to the modeling technique are addressed in [Musliner *et al.*, 1992].

In the figure, states {A,B,C,D,E} form a safe set of states S_{safe}: no event transitions lead out of the set. However, time transitions can still lead to failure. In the example, we can see this in the transition from state B to state C caused when an obstacle appears in the path; if the system waits too long to recognize the situation and take action, it will follow the time transition to F by colliding with the obstacle. But, if the system quickly detects the obstacle and halts, it can avoid a collision and transition to state E instead. Thus, if the system can guarantee that it will always preempt time transitions that lead from states in the safe set to the failure state, then *the system can remain in a safe set of states for an indefinite period of time without violating its control-level goals or the domain constraints.* The big "if" requires that the system provide the appropriate action transitions to stay within the safe set. CIRCA was designed to achieve just that, using the AIS and Scheduler to reason about which transitions to guarantee, and the RTS to implement those guaranteed transitions with TAPs. Selective perception is based on the graph model's representation of the causal behavior of the world [Simmons, 1990], showing which world conditions are critical [Musliner *et al.*, 1991] and how frequently they must be confirmed.

Accounting for Bounded Reactivity

The AIS can account for the limitations on sensor and actuator resources available to the RTS by modifying its graph model. For example, if the AIS attempts to guarantee responses to preempt both of the time transitions to failure shown in Figure 3, the Scheduler may indicate that the RTS does not have sufficient sensor resources to guarantee both responses (i.e., it cannot sense for both orientation and obstruction frequently enough). The AIS can then modify its model to decrease the sensor requirements, possibly by eliminating low probability transitions associated with the oversubscribed sensor. In this example, the AIS might stop considering the transition from B to C, eliminating the demand for sensors to check for state C. This solution trades off completeness against guaranteed timeliness, so that the system can no longer guarantee its safety

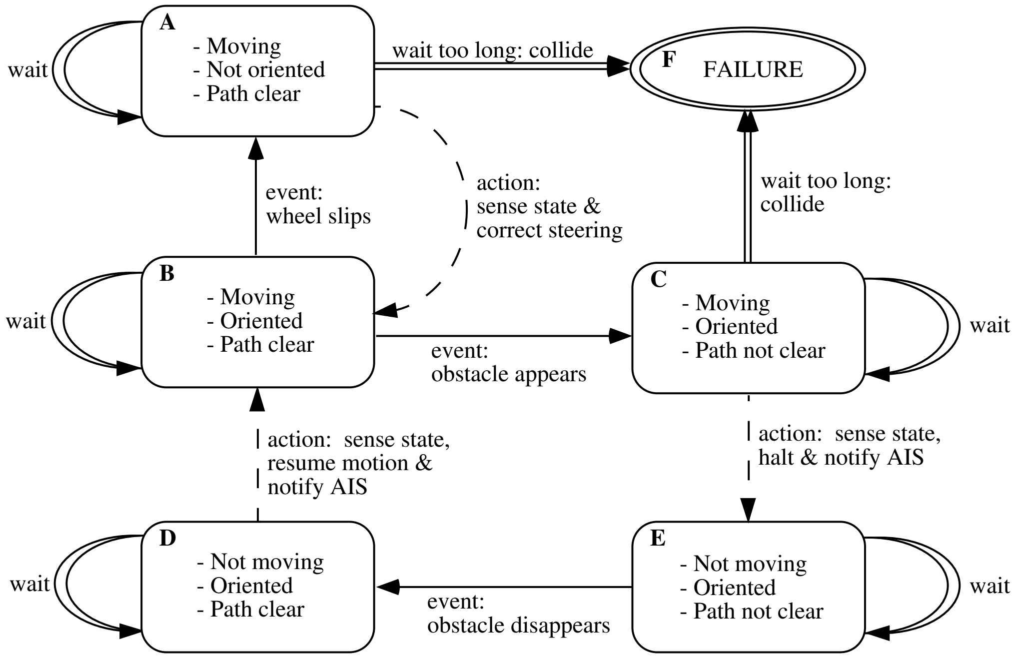

Figure 3: An example portion of the graph model of RTS/environment interactions.

from all known forms of failure. Perceptual selectivity is increased to meet the constraints of bounded reactivity.

As an alternative approach to modifying the model to account for resource limitations, the AIS might change a parameter which affects the deadlines associated with time transitions to failure. For example, if the robot's speed is decreased, the time before the transition from state C to F will be increased. Thus the RTS would not have to check for obstacles ahead as frequently, and the demands on the sensors would decrease.

This process of reasoning about resource constraints is CIRCA's mechanism for making tradeoffs in the completeness of its guaranteed responses. The system strives to guarantee to preempt all known transitions to failure; when it cannot, it modifies the modeled set of states and transitions, and thus alters the selected perceptions and reactions it guarantees.

Conclusion

We have implemented and tested preliminary versions of each CIRCA subsystem. We are in the process of refining the interface software, and will soon have a fully operational prototype system. We are also investigating techniques for automatically generating the graph model from a more compact representation, allowing the system designer to easily specify environment features.

In summary, CIRCA is an innovative architecture in which cooperating subsystems provide both the performance guarantees needed for real-time control and the powerful, unpredictable intelligence needed to address complex task-level problems. Through the TAP structure and the graph model of the environment, CIRCA is able to alter its selected perceptions and reactive behaviors to account for both the system's bounded reactivity and its real-time goals.

References

[Boddy and Dean, 1989] Mark Boddy and Thomas Dean. Solving time-dependent planning problems. In *Proc. Int'l Joint Conf. on Artificial Intelligence*, pages 979–984, August 1989.

[Brooks, 1986] Rodney A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2(1):14–22, March 1986.

[Firby, 1987] R. James Firby. An investigation into reactive planning in complex domains. In *Proc. National Conf. on Artificial Intelligence*, pages 202–206, 1987.

[Musliner *et al.*, 1991] David J. Musliner, Edmund H. Durfee, and Kang G. Shin. Execution monitoring and recovery planning with time. In *Conf. on Artificial Intelligence Applications*, pages 385–388, February 1991.

[Musliner *et al.*, 1992] David J. Musliner, Edmund H. Durfee, and Kang G. Shin. CIRCA: a cooperative intelligent real-time control architecture. *to appear in* IEEE Trans. Systems, Man, and Cybernetics, 1992.

[Simmons, 1990] Reid Simmons. An architecture for coordinating planning, sensing, and action. In *Proc. Workshop on Innovative Approaches to Planning, Scheduling and Control*, pages 292–297, November 1990.

[Simon, 1982] Herbert Alexander Simon. *Models of Bounded Rationality*. M. I. T. Press, 1982.