PREDICTIVE SUFFICIENCY AND THE USE OF STORED INTERNAL STATE

David J. Musliner
Institute for Advanced Computer Studies
The University of Maryland
College Park, MD 20742
musliner@umiacs.umd.edu

Edmund H. Durfee and **Kang G. Shin**
Dept. of EE & Computer Science
The University of Michigan
Ann Arbor, Michigan 48109-2122
{durfee,kgshin}@eecs.umich.edu

Abstract

In all embedded computing systems, some delay exists between sensing and acting. By choosing an action based on sensed data, a system is essentially predicting that there will be no significant changes in the world during this delay. However, the dynamic and uncertain nature of the real world can make these predictions incorrect, and thus a system may execute inappropriate actions. Making systems more reactive by decreasing the gap between sensing and action leaves less time for predictions to err, but still provides no principled assurance that they will be correct.

Using the concept of *predictive sufficiency* described in this paper, a system can prove that its predictions are valid, and that it will never execute inappropriate actions. In the context of our CIRCA system, we also show how predictive sufficiency allows a system to guarantee worst-case response times to changes in its environment. Using predictive sufficiency, CIRCA is able to build real-time reactive control plans which provide a sound basis for performance guarantees that are unavailable with other reactive systems.

Introduction

Traditional AI planning systems[3,10,15] have been criticized because they may spend large amounts of time building a plan that is out-of-date before it can be used, and thus the actions that the plan chooses may be inappropriate. For example, consider an intelligent autonomous vehicle that is waiting at a red light. When the light changes to green, the vehicle's sensors detect the change and, after some further processing, the system decides to move through the intersection and on to its destination. But, if the system spent too much time planning its entire route, the light may have changed back to red, and the plan's first action would be "inappropriate."

In response to this critique, researchers have developed reactive systems[1,2,4,6,13] that perform little or no lookahead planning, instead choosing actions based on current sensor inputs. One goal of this behavior is to keep the selected actions appropriate to the current situation: because no planning is done, an action can be chosen quickly once sensor readings determine the current situation.

However, because computations can only occur at some finite speed, there will always be some delay between sensing and action. During this "sense/act gap," sensed information is stored in the system, either explicitly in memory modules or implicitly in the communication and processing mechanisms of the system. By choosing an action based on that stored information, the system makes an implicit *prediction* that the stored information will continue to provide a sufficiently accurate representation of the world.[5]

Because real-world systems are dynamic and somewhat uncertain, such predictions are inherently risky. Gat[5] suggested that these predictions and the associated stored internal state are useful only at higher levels of abstraction. We argue that, because the gap between sensing and action is inevitable, it is not the abstraction level but the magnitude of this delay (and the requisite prediction) that is critical. Systems in dynamic worlds must be "real-time," in the sense that the utility of the system's computations depends not only on their result, but on when that result is produced.[14] To guarantee correct performance, an intelligent real-time system must ensure that the actions it chooses are appropriate for the actual current state of the world, not just the state of the world that was last sensed.

The work reported in this paper was supported in part by the National Science Foundation under Grants IRI-9209031 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.

Rather than solving the real-time problem, reactive systems simply operate in a "coincidently real-time" manner[7]— they function as quickly as possible, in the hopes that the sense/act gap will be reduced so much that significant world changes cannot occur during the gap. In this paper, we present a more rigorous approach to dealing with the sense/act gap. Our approach consists of *proving* that significant world changes cannot cause a particular selected action to be inappropriate, by verifying that the predictions spanning the sense/act gap are valid.

In the next section, we lay the foundations for this proof by defining the "interval of predictive sufficiency," or the time during which an observation provides sufficient evidence to accurately predict the value of some proposition. In the following section, we illustrate how explicit reasoning about predictive sufficiency can be implemented, with examples from CIRCA, the Cooperative Intelligent Real-time Control Architecture.[8,9] We describe how CIRCA uses predictive sufficiency while building real-time reactive control plans, to guarantee that the system will never choose inappropriate actions or miss real-time reaction deadlines. This paper concludes with sections discussing the type of knowledge that is required for reasoning about predictive sufficiency, and pointing out future directions for this research.

Defining Predictive Sufficiency

To accurately describe the concept of predictive sufficiency, we must begin with some notation. We will use a simple temporally-qualified modal logic to describe the state of a control system's knowledge. The logical statement $K(p[t_i], t_j)$ indicates that the system knows, at time t_j, that the proposition p holds at time t_i. For convenience, we will also use statements of the form $K(p[t_\alpha, t_\beta], t_j)$, indicating that the system knows, at time t_j, that p holds continuously over the time interval from t_α to t_β.

A control system's operations can be generally expressed as the acquisition of a sensory observation, the logical deduction of what that observation means about the state of the world at the time the observation was made, the deduction of the predictions that the observation allows the system to make about the world following the observation, and the selection of an action based on that knowledge. In our notation, we have:

$$
\begin{array}{rl}
O[t_i] & \\
\downarrow & \text{interpret} \\
\forall p \in P : K(p[t_i], t_j) & \\
\downarrow & \text{predict} \\
\forall q \in Q : K(q[t_{q\alpha}, t_{q\beta}], t_k) & \\
\downarrow & \text{select} \\
a[t_{a\alpha}, t_{a\beta}] &
\end{array}
$$

where $O[t_i]$ is a sensory observation made at time t_i, P is the set of propositions which can be inferred about the world at time t_i from the observation, and Q is the set of propositions that can be predicted over the respective intervals $[t_{q\alpha}, t_{q\beta}]$. These intervals are the "intervals of predictive sufficiency," during which the observation O is sufficient to predict the value of the propositions Q. The time t_j is the time by which the system has derived its knowledge of P, and t_k is the time by which the system knows Q. Following those deductions, the action a is chosen and executed during the time interval $[t_{a\alpha}, t_{a\beta}]$.

We first use the concept of predictive sufficiency to show how an action can be guaranteed to be appropriate when it is executed. The key to avoiding an inappropriate action is to ensure that the value of the propositions used to choose an action will remain unchanged long enough to keep the action appropriate. This can be achieved by making action choices based on propositions whose intervals of predictive sufficiency cover the time during which the action's preconditions are necessary. More formally, suppose the action a requires a set of propositions R to hold during the respective intervals $[t_{ra}, t_{rb}]$. If $R \subseteq Q$ and $\forall r \in R : (t_{r\alpha} \leq t_{ra}) \wedge (t_{r\beta} \geq t_{rb})$, then the intervals of predictive sufficiency that are supported by the observation O ensure that the required propositions will hold as necessary.

For example, in the stoplight scenario described earlier, the vehicle agent will at some point make an observation confirming the proposition "the light is green" (P). This proposition alone is not sufficient to justify crossing the intersection, because there is no guarantee that, at the time t_j when P is known, the light is *still* green. The knowledge resulting directly from interpreting sensor readings can only describe past states of the world. However, if the system knows some information about the domain's dynamic behavior, it can derive additional propositions that describe the current and future worlds. In this example, the system might know that the traffic signal will switch to yellow for at least five seconds before it turns red. So, although the system does not know if the light is still green, it can conclude that, for at least five seconds after the light was seen to be green, the light must be either green or yellow, and the intersection will be "safe" to cross (Q). If the agent is sure that the time it takes to infer these propositions from its observations and cross the intersection is less than five seconds, it can guarantee that it will never be in the intersection during a red light.

Thus the addition of domain modeling information has allowed the system to make explicit predictions about the future state of the world, based

on stored sensor readings. Given further information about the agent's own performance, these predictions are then shown to be sufficient to justify certain actions. This example illustrates how predictive sufficiency can cover the sense/act gap, avoiding inappropriate actions.

Implementing Predictive Sufficiency

In this section, we provide a high-level description of CIRCA and show how the prototype implementation of the architecture explicitly reasons about predictive sufficiency and makes guarantees about its behavior. Note that we do not claim this implementation is ideal; it serves only as a useful testbed to demonstrate the concepts of predictive sufficiency. More details on CIRCA are available in related publications.[8,9]

Figure 1 illustrates the architecture, in which an AI subsystem (AIS) and Scheduler cooperate to strategically plan and schedule a set of reactive behaviors that will cope with a particular expected domain situation. The parallel real-time subsystem (RTS) is guaranteed to accurately execute the behavior schedules, comprised of simple situation-response rules. In this paper, we are focusing on how the prototype AIS explicitly reasons about the sense/act gap and predictive sufficiency while planning reactions. Note that this lookahead planning is performed while previously-planned reactions are already executing on the RTS, so the planning process can be viewed as "off-line." To show how CIRCA uses predictive sufficiency, we must first briefly describe the system's world modeling techniques, which it uses to reason about the behavior of the world and the actions that the system should take to achieve its goals.

In the prototype implementation, the world model takes the form of a directed graph in which nodes represent possible states of the world and arcs represent instantaneous transitions between states. The status of ongoing processes in the world is explicitly encoded into the representation of a state. Important changes in process status thus correspond to transitions between states. The model distinguishes three types of state changes: *action transitions*, performed deliberately by the system's reactions; *event transitions*, due to external world occurrences; and *temporal transitions*, due to the passage of time and ongoing processes. Timing information is associated with each transition, representing constraints on how long the world must remain in a state until the transition may occur. We now illustrate how this model is used by the AIS to explicitly reason about the sense/act gaps that will occur when planned behaviors are executing on the RTS, and how the system guarantees that those gaps will not lead to inappropriate actions.

Avoiding Inappropriate Actions

Figure 2 shows an example portion of the graph-based world model for the stoplight scenario described above. Within the state descriptions, the model shows that the stoplight can take on its three signal colors, Red, Yellow, and Green. In the Yellow and Green states, it is safe for the agent to cross ("Safe2X"), but not in the Red state. In this simple example, we have abstracted out all of the agent's own state except for the indication of whether it has crossed the intersection or not. The different states of the traffic signal are connected by temporal transitions (double arrows) indicating that, as time passes, the signal will transition to subsequent states. Each temporal transition is labeled with the minimum possible delay before the transition occurs, perhaps derived from the agent's previous experience with this traffic signal. For example, the transition between the Red and Green states indicates that the signal will stay red for at least 60 seconds before turning green.

When planning reactions to operate in this domain, CIRCA does not build an enumeration of possible world states and then plan actions; instead, it dynamically constructs the graph model and the plan of actions together in a single depth-first search process, essentially similar to a forward-chaining STRIPS planner.[10] This process operates on a stack of world model states, examining each state in turn and planning actions that achieve goals and preempt temporal transitions that lead to failure.

To begin the planning process, the initial states are pushed onto the state stack. Then, as long as the stack is not empty, the system pops a state off the stack and considers it the *current state*. The system simulates all of the event transitions and temporal transitions that apply to the current state, yielding either new states that have not been examined yet or states that have already been processed (i.e., states for which actions have already been planned). New states are pushed onto the state stack, while old states are simply updated with the information that they have a new source state. The system then chooses an action to take in the current state, as determined by a heuristic scoring function.

For example, if the system is told that the "red" state $\mathcal{A}$ is its initial condition, it will first consider the applicable event and temporal transitions, pushing the new "green" state $\mathcal{B}$ onto the stack. The system will then try to plan an action for state $\mathcal{A}$; since the state is not safe for crossing, the only applicable action is `no-op` (shown as a dashed line in

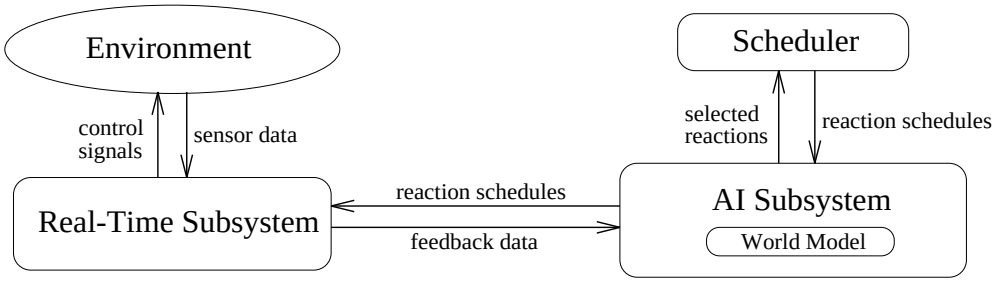

Figure 1: Overview of CIRCA.

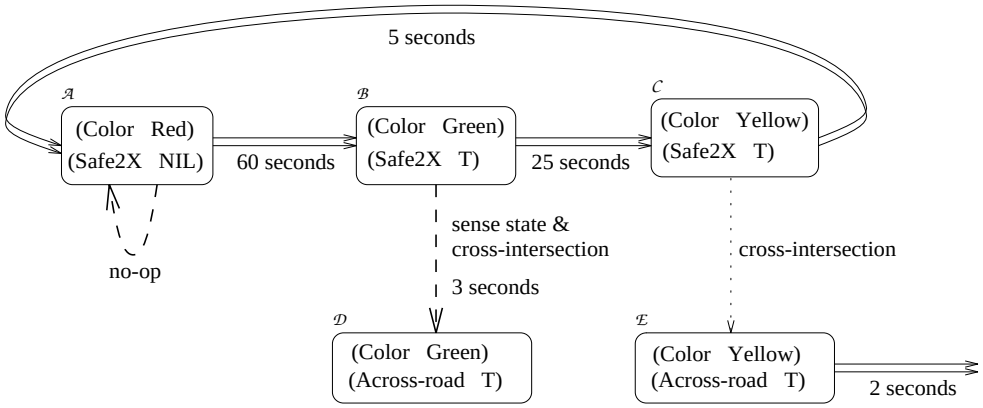

Figure 2: An abstracted portion of the world model for the stoplight scenario.

Figure 2). The system will then mark state $\mathcal{A}$ as processed, pop state $\mathcal{B}$ off the stack, and derive the new successor state $\mathcal{C}$ via the temporal transition indicating that the light will change to yellow. Again an action is chosen for the current state, but this time the `cross-intersection` action is chosen because it is applicable (Green is safe to cross) and because it leads to the desired result. So at this point CIRCA has planned a simple reaction indicating that, when the light is green, the agent should cross. But the system has not yet shown why this action is guaranteed to be appropriate when executed; it has not yet addressed the sense/act gap, and the possibility that the light will change before the `cross-intersection` action is completed.

CIRCA addresses these issues by ensuring that the propositions used to satisfy the action's preconditions are covered by intervals of predictive sufficiency. The system knows the worst-case execution time of all of its sensing and action primitives, as well as their combinations. Thus the system knows exactly how long it will take, in the worst case, to detect the green light and cross the intersection (here, three seconds). To check for predictive sufficiency, the system must look for other domain processes that may be occurring during the action (i.e., transitions to other states). In this case, the system has recognized, based on domain knowledge, that there can be a temporal transition leading from the green state $\mathcal{B}$ to the yellow state $\mathcal{C}$ after a minimum of 25 seconds.

As noted above, CIRCA does not know how long the light has been green when it is observed; therefore, in the worst case, it is assumed that the temporal transition to the yellow state $\mathcal{C}$ occurs at the same time the system initiates the transition to cross the intersection. This corresponds to the "ghost" action transition in the figure (the dotted line), showing that the action planned for state $\mathcal{B}$ may actually be applied to state $\mathcal{C}$, leading to a new state $\mathcal{E}$ where the signal is yellow, but there is now a minimum of only two seconds before a temporal transition leads to a red light state.

In this process of looking at transitions out of the state for which the action is planned, CIRCA has shown that, although alternate results are possible, the precondition of the action ("safe2X") is known to hold for five seconds. This is the interval of predictive sufficiency: seeing a green light allows the system to guarantee at least five more seconds of safe crossing time. Because the process of sensing the green light and then crossing the street takes no more than three seconds, the interval of predictive sufficiency is long enough to cover the sense/act gap. Therefore, CIRCA can plan this action and guarantee that it will only

be executed in appropriate situations*.

When CIRCA continues the planning process and tries to choose an action for the yellow state $\mathcal{C}$, it finds that the `cross-intersection` action is applicable and leads to the desired state. However, when the system tries to ensure that the "safe2X" precondition can be predicted to hold while the action is executed, it finds that a temporal transition leaving state $\mathcal{C}$ leads to the red state $\mathcal{A}$, which is "unsafe2X." Therefore, since the system does not know how much time may have passed in the yellow state $\mathcal{C}$ before the state was detected, and the subsequent state does not satisfy the action's preconditions, the action is rejected. In summary, CIRCA has used its explicit understanding of predictive sufficiency to derive a common rule of thumb used by drivers who glance at a traffic signal: if the light is green, go ahead and cross; if the light is yellow, do not start crossing, because the light may turn red too soon.

An interesting feature of this approach to avoiding inappropriate actions is that it requires no information about how frequently a particular sensory observation is being acquired— the example said nothing about how often the system checks to see if the light is green. If the system never even checks to see if the light is green, and thus never takes the `cross-intersection` action, it will never perform an inappropriate action. Clearly, this type of proof is only useful for goals that have no deadline. For real-time goals, that require response-time guarantees, this method is not sufficient.

To describe CIRCA's approach to meeting such real-time deadlines, we first introduce a more complex application domain.

The Puma Domain

The stoplight domain was used above for its intuitive simplicity; CIRCA has also been applied to a much larger robot control problem, illustrated by the simulation image in Figure 3. The Puma is assigned the task of packing parts arriving on the conveyor belt into the nearby box. Once at the end of the belt, each part remains motionless until the next part arrives, at which time it will be pushed off the end of the belt (unless the robot picks it up first). If a part falls off the belt because the robot does not pick it up in time, the system is considered to have failed. Thus, the arriving parts impose hard deadlines on the robot's responses; it must always pick up arriving parts before they fall off the conveyor.

The Puma is also responsible for reacting to an emergency alert light. If the light goes on, the system has only a limited time to push the button next to the light, or the system fails. This portion of the domain represents a completely asynchronous interrupt with a hard deadline on its service time.

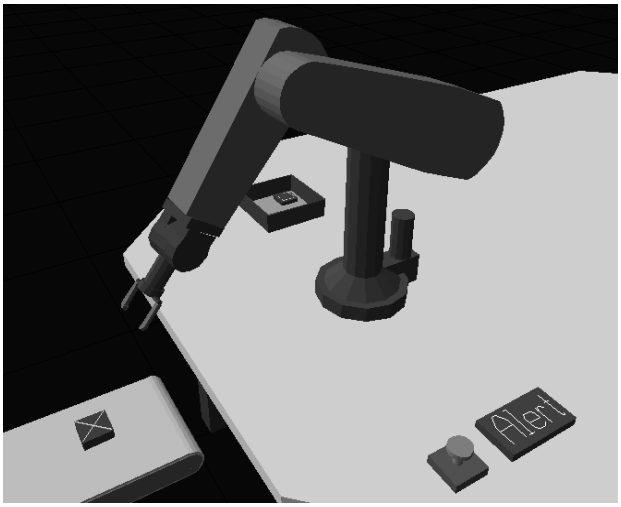

Figure 3: The Puma domain, with two hard real-time deadline constraints.

Real-Time Response Guarantees

To deal with the hard deadlines in the Puma domain, the planning methods described above are not sufficient— they do not ensure that reactions will be timely, but rather that they will never be inappropriate. As we shall see, CIRCA must merge even more knowledge with its sensing information to guarantee timely responses that meet hard deadlines.

Figure 4 illustrates a small portion of the world model for the Puma domain†, showing the representation of the hard deadline on picking up arriving parts. Parts are known to be spaced apart on the conveyor by at least some minimum distance. After a part arrives, the conveyor belt is considered to be "busy" for some amount of time (corresponding to the minimum part spacing) before the next part may arrive. Thus, from state $\mathcal{A}$ (where CONVEYOR-STATUS is BUSY) there is a temporal transition to state $\mathcal{B}$ (where CONVEYOR-STATUS is FREE), tagged with the value $min\Delta = 10$ (seconds) to indicate that state $\mathcal{A}$ must persist at least that long before the transition to state $\mathcal{B}$. From state $\mathcal{B}$, an event transition represents the fact that a part may arrive at any time, leading to state $\mathcal{C}$. The potential failure resulting from the part falling off the conveyor is represented by the temporal transition out of state $\mathcal{C}$, also tagged with $min\Delta = 10$: if the next part arrives

*CIRCA currently only supports this test for preconditions that are required over the entire duration of an action.

†The full domain model includes more state features and hundreds of states and transitions.

while this part is still on the conveyor, failure will occur.

To understand CIRCA's approach to making response-time guarantees, let us examine the planner's operation when it is considering state $\mathcal{C}$. The first phase of the planning process finds applicable event and temporal transitions, and recognizes that there is a potential temporal transition to failure. Since the failure is defined to be catastrophic, CIRCA realizes that it must preempt the temporal transition. That is, CIRCA decides it must execute some action that will definitely occur before the earliest time the temporal transition to failure can occur. A simple lookahead shows that the action `pickup-part-from-conveyor` will successfully avoid the failure. Now the only challenge is to ensure that the action will happen quickly enough. To ensure that the transition to failure is preempted, CIRCA commits to repeatedly executing a reaction that checks for the conditions of state $\mathcal{C}$ and implements the chosen action, at least frequently enough to ensure that the action will be completed before failure can occur. That is, CIRCA decides how quickly it must poll the sensors to detect the imminent failure and prevent it.

It is fairly obvious that, to guarantee that the system will simply detect the potential failure represented by state $\mathcal{C}$, which has a minimum possible duration ($mindur(P)$) of 10 seconds, CIRCA must test for the state at least once every 10 seconds. However, detecting the state $\mathcal{C}$ is not sufficient: the system must be able to *finish* the action of picking up the part before it can fall off the conveyor. In the terms introduced previously, the interval of predictive sufficiency during which the part is known to remain on the conveyor must cover the chosen action, in addition to its preconditions. To provide this predictive sufficiency, CIRCA relies on its additional knowledge about the frequency with which CIRCA itself will be obtaining sensory information. For example, if the period of the repeated observations is $\rho(O)$ seconds, then an observation in which the condition does hold, following an observation in which the condition does not hold, indicates that the change of state must have occurred in the last $\rho(O)$ seconds. Therefore, the condition must continue to hold for at least $mindur(P) - \rho(O)$ seconds.

Thus we have a modified interval of predictive sufficiency, based on both knowledge of the domain and knowledge about the ongoing performance of the reactive system itself. The AIS actually reasons about the performance of the reactive system it is designing to derive the predictive sufficiency of the observations it plans to make. To guarantee that every real-time reaction will be checked and executed before its corresponding deadline, CIRCA must show that the predictive sufficiency of the observations covers the sense/act gap and the duration of the chosen action. That is, $mindur(P) - \rho(O) > t_{a\beta} - t_i$. In our Puma domain example, if the `pickup-part-from-conveyor` action takes 3 seconds, we have $10 - \rho(O) > 3$, so that $\rho(O) < 7$. If CIRCA can guarantee to execute the reaction that tests for state $\mathcal{C}$ and picks up the part at least once every 7 seconds, it can guarantee that it will not drop any parts off the conveyor[‡].

Making this reaction frequency guarantee is the job of CIRCA's Scheduler module (see Figure 1). The AIS uses the methods described above to derive frequency requirements for mission-critical reactions, and sends those reactions to the Scheduler. The Scheduler examines the capacity of the RTS to see if the available resources are sufficient to meet those requirements: if so, a schedule of reaction executions is returned to the AIS. If the RTS resources are not sufficient to guarantee the reaction rates specified by the AIS, the Scheduler will return an error message to the AIS, indicating that some performance tradeoff will be required in this overconstrained domain.

Knowledge Requirements

As we have noted, predictive sufficiency can only be established by combining immediate sensor information with additional knowledge about the domain. The basic form of the required knowledge is the "minimum duration" of some condition. That is, the system must know that some sensed state of the environment always persists for some minimum amount of time. In the stoplight domain, for example, the system must know the minimum duration of each signal color. In general, this type of knowledge might be acquired in one of two ways.

First, the system might have previous experience with the domain (or similar domains), and be able to extrapolate from that experience the requisite minimum durations. Experienced drivers know that no green light lasts for less than 5 seconds. Learning and past experience can thus play a key role in reasoning about predictive sufficiency.

Second, knowledge of minimum durations may also be derived from simple first principles, given precursor knowledge of the maximum rate of related (underlying) processes. For example, in the Puma domain, the minimum duration of the (CONVEYOR-STATUS BUSY) condition is determined by the maximum part arrival rate, which in turn is based on the conveyor belt speed and the spacing between parts. So if the system knows that parts must be at least ten inches

[‡]At least, not from this particular part of the state space.

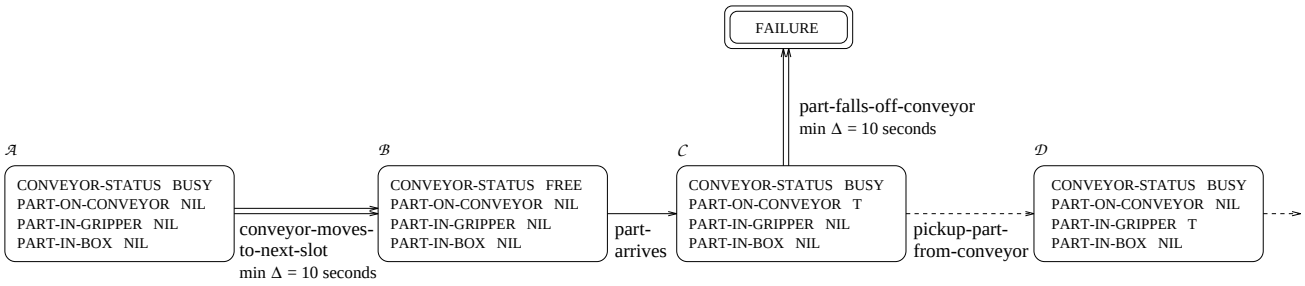

Figure 4: A small, abstracted portion of the Puma domain model.

apart and that the belt is moving at one inch per second, then the maximum part arrival rate is six parts per minute, and the minimum duration of the (CONVEYOR-STATUS BUSY) condition is ten seconds.

Currently, CIRCA makes no effort to learn minimum-duration knowledge itself, and it has only rudimentary, domain-specific methods to derive that knowledge from process rates. Instead, our focus has been on having CIRCA use that knowledge to reason about predictive sufficiency, and investigating the effects of explicitly dealing with the sense/act gap.

Conclusion

We have argued that all computing systems must make predictions about how the state of the world will evolve during the delay between sensing and action. The intuition behind the trend toward reactive systems has been that reducing this delay simplifies (but does not eliminate) prediction. In this paper, we have described how this intuition is really attempting to capture implicitly the concept of *predictive sufficiency*. By explicitly representing and reasoning about predictive sufficiency, we can determine exactly how long a gap between sensing and acting is allowable within a system, given its environment and its capabilities.

Predictive sufficiency is a critical concept for embedded agents, because it permits a system to make guarantees about its behaviors. We have shown how CIRCA implements predictive sufficiency to guarantee that it will not execute inappropriate actions and that it will react to its environment frequently enough to meet real-time deadlines.

Explicitly reasoning about predictive sufficiency also allows us to break away from the mind-set that decreasing the delay between sensing and acting is always desirable. Specifically, knowing the predictive sufficiency of an observation may allow a system to avoid some sensor polling by caching sensory data. No sensor readings need to be taken as long as a previous observation's interval of predictive sufficiency remains in force. We are investigating ways in which CIRCA can use its explicit knowledge of predictive sufficiency to design sensor caching schemes that maximize the use it gets out of each observation, reducing the frequency of costly observations without compromising the system's performance guarantees.

Our investigation of predictive sufficiency is a first step towards a more complete understanding of exactly when stored internal state is useful, and when it can lead to invalid predictions and failures. We hope to unify this approach with the epistemic proofs of Rosenschein and Kaelbling[11,12] to establish a full theory of the correspondence between a system's internal state, its predictions, and the world. This theory would allow strong prescriptive statements about when and how to use stored internal state.

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] 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.

[3] R. E. Fikes and N. J. Nilsson, "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving," *Artificial Intelligence*, vol. 2, pp. 189–208, 1971.

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

[5] E. Gat, "On the Role of Stored Internal State in the Control of Autonomous Mobile Robots," *AI Magazine*, vol. 14, no. 1, pp. 64–73, Spring 1993.

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

[7] 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.

[8] D. J. Musliner, *CIRCA: The Cooperative Intelligent Real-Time Control Architecture*, PhD thesis, The University of Michigan, Ann Arbor, MI, September 1993. Also available as CSE-TR-175-93.

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

[10] N. J. Nilsson, *Principles of Artificial Intelligence*, Tioga Press, Palo Alto, CA., 1980.

[11] S. J. Rosenschein, "Synthesizing Information-Tracking Automata from Environment Descriptions," Technical Report 2, Teleos Research, July 1989.

[12] 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.

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

[14] 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.

[15] D. Wilkins, "Domain-Independent Planning: Representation and Plan Generation," *Artificial Intelligence*, vol. 22, no. 3, pp. 269–301, April 1984.