The Hybrid Planning-Scheduling System in CIRCA-II

Ella M. Atkins
Department of Aerospace Engineering
University of Maryland
College Park, MD 20742
atkins@eng.umd.edu

ABSTRACT

Autonomous operation in a dynamic environment requires accurate and timely response. A *real-time control plan* specifies actions plus a resource schedule to guarantee real-time constraints will be met during plan execution. The Cooperative Intelligent Real-time Control Architecture (CIRCA-II) employs an integrated planning-scheduling system to create and execute real-time control plans. In this paper, we describe CIRCA-II and its application to real-time domains.

Introduction

Autonomous behavior in complex real-world systems requires accurate and timely reactions to environmental events. These reactions must prevent all catastrophic failures such as loss-of-life and should ultimately achieve mission goals such as arriving at a destination on time. Timely and accurate responses for a complex domain may require a significant amount of computational resources, regardless of whether such responses are pre-programmed or dynamically selected as the agent acts within its environment. As processor speed and algorithm efficiency increase, it is tempting to presume that resource limitations are not an issue because they can always be combated with a bigger, faster system. However, the exponentially-complex search-based planning and scheduling algorithms typically utilized to impart "intelligence" to a complex autonomous system can quickly consume all such resources, as can the storage and retrieval-time requirements for reactions in strictly plan-execution systems. Additionally, hardware upgrades are not easily performed in unfriendly, resource-limiting environments (e.g., space, underwater).

In this paper, we present our definition of a real-time control plan and argue that such plans are required to satisfy hard real-time execution requirements in time-constrained environments. The Cooperative Intelligent Real-time Control Architecture (CIRCA) (Musliner, Durfee, and Shin 1995) and, more recently, CIRCA-II (Atkins 1999) explicitly combine distinct planning and scheduling algorithms into a single system in order to produce hard real-time control plans that achieve mission goals while providing absolute safety guarantees in time constrained environments. The state-space planner specifies a set of actions required to guarantee safety and achieve goals, then the distance-constrained real-time scheduler places the safety-preserving action subset in a cyclic schedule based on their worst-case execution properties and planner-specified separation constraints (e.g., deadlines). In this paper, we overview the CIRCA-II architecture and describe its combination of planning and scheduling algorithms into a single system capable of creating and executing real-time control plans.

Real-time Control Plans

Depending on research focus, the term *plan* may refer to either a sequence of actions or else a policy that applies to a group of world states. Due to our veritable obsession with hard real-time plan execution, our plans must include more than constructs for matching actions to states. Figure 1 illustrates two of the most popular interpretations of a plan. Figure 1a shows a STRIPS plan (Fikes and Nilsson, 1987), a sequential action format produced by many popular state-space and plan-space systems. This specification is appropriate when actions must be strictly executed in a predefined sequence. The STRIPS plan structure does not rely on active sensing during plan execution, implying there can be no uncertainty about when or in what order actions should execute. Figure 1b illustrates a policy representation such as that generated by a traditional Markov Decision Process (MDP) (Boutilier, Dean, and Hanks, 1999). In this model, there is uncertainty regarding the exact progression of states that will be encountered, so the set of current state features must be sensed and

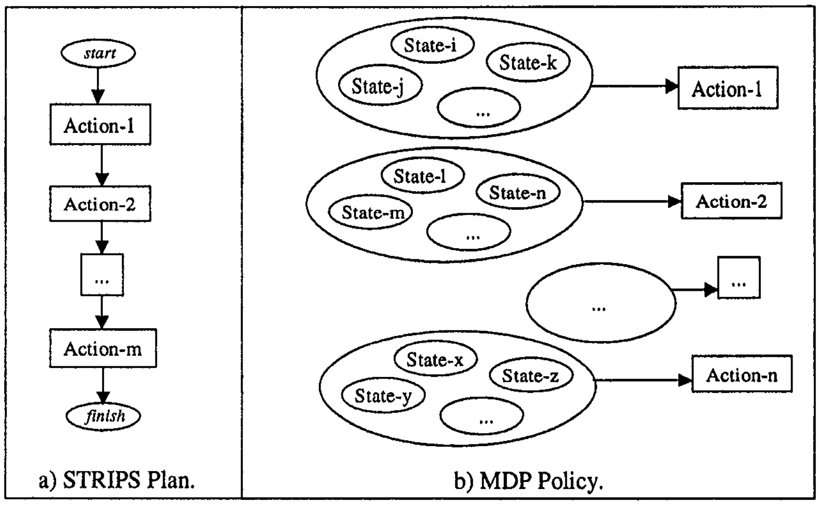

Figure 1: Traditional Plan Types.

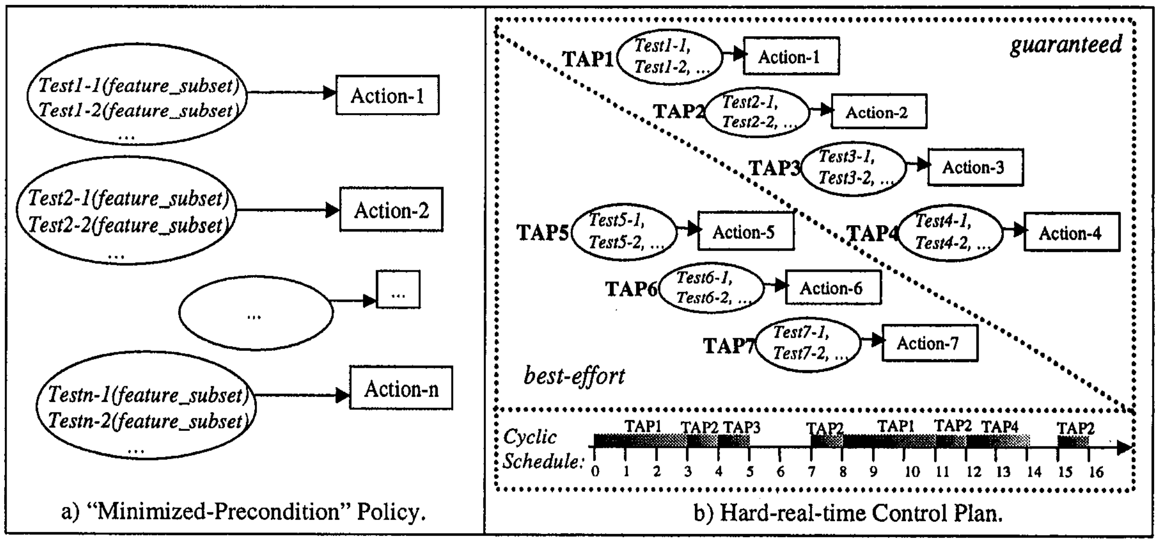

Figure 2: Evolution of the Real-time Control Plan.

matched to the correct action to execute next. As a result, reaction times to environmental events are a function of the total time required to identify the current state, find the appropriate action, then execute that action.

Because the progression of world states may not be known during the advance planning stage, a plan execution system may require state feature sensing to select appropriate actions during plan execution. However, dynamic and dangerous environments also require that the complete sense-act loop execute in hard real-time for failure-avoidance purposes. To define our notion of real-time control plan, consider an MDP policy as the initial representation. Now, to increase efficiency for matching the current world state to an action, consider a new format in which the policy is post-processed so that a set of general "preconditions", not fully-instantiated states, is used to uniquely match each reachable state to a policy action. This "minimized precondition" policy representation is shown in Figure 2a.

In a policy where the exact sequence of states cannot be predicted, the "minimized-preconditions" for executing each action must be checked periodically, with each action executing whenever its preconditions match. Otherwise, the action may never execute in a state where it has been planned. This plan structure suggests a loop over the precondition-action pairs to identify and execute the proper action for each state. A cycle through the plan-loop will not execute instantaneously, so each action's preconditions must be tested with sufficient frequency to guarantee avoiding any failures that might occur should action execution delay too long.

If all actions are required for failure-avoidance and all actions have the same real-time execution deadlines for failure-avoidance, then the best we could do is to cycle through the plan-loop as-is. However, typically, only certain actions are required for failure-avoidance while others are used only for goal-achievement. We attach to each action the worst-case timing requirements for guaranteed failure-avoidance, and classify all actions with specified worst-case timings as "guaranteed" while all others are "best-effort", as illustrated in Figure 2b. Now, if all guaranteed actions have the same worst-case timing requirements, we can execute the "plan-loop" over all guaranteed actions, inserting best-effort actions into slack time intervals when available. However, in general, the guaranteed actions may have a very diverse set of real-time requirements. Thus, instead of looping over each action in the guaranteed set, we may maximize our ability to guarantee that all execute in time by explicitly scheduling these actions in accordance with their resource requirements and real-time deadlines.

Figure 2b includes a cyclic schedule that specifies the "plan-loop" for the set of guaranteed actions for this plan. We define a *task* as the combination of the minimized-precondition feature tests for the action as well as the action itself. For guaranteed performance, this schedule must be built assuming worst-case task resource consumption, and must verify that all real-time constraints for the associated action will be met during execution. In CIRCA-II, we define a *real-time control plan* as the Figure 2b combination of a minimized-precondition task set and cyclic task schedule that guarantees real-time failure-avoidance during plan execution.

CIRCA-II Architecture

The Cooperative Intelligent Real-time Control Architecture (CIRCA) (Musliner et al, 1995) distinctly separates planning, scheduling, and plan-execution processes such that plans are developed, scheduled, then executed in hard real-time when required for safety guarantees. Using this approach, only the plan execution part of the system must execute in strict accordance with deadlines, thereby allowing the planner and scheduler to reason *about* real-time rather than *in* real-time. To build real-time plans, CIRCA's planner employs a time-dependent state transition model and a representation for system "failure" such that plans contain two classes actions: "guaranteed" with hard deadlines for failure avoidance, and "soft real-time" with best-effort execution for goal achievement.

The original CIRCA architecture has evolved into a second-generation system (CIRCA-II) (Atkins 1999), illustrated in Figure 3. CIRCA-II incorporates a probabilistic planning algorithm along with a plan cache to facilitate hard real-time response when a single real-time control plan is incapable of guaranteeing safety in both probable and unlikely situations. At the highest level, the architecture is divided into a Planning Subsystem and a Plan-Execution Subsystem. We have partitioned CIRCA-II tasks such that the Planning Subsystem includes all processes for which we cannot easily define reasonable worst-case execution properties. At best, CIRCA-II will be able to complete *Planning Subsystem* operations in *coincidental real-time*. Thus, we relegate the majority of planning and scheduling operations to occur offline before the system ever enters its "dangerous" environment. Conversely, safety-critical tasks in the *Plan-Execution Subsystem* require hard real-time execution. This module includes a Plan Executor that is responsible for executing each single control plan in accordance with real-time constraints, along with a Plan Dispatcher that manages a Plan Cache used for real-time response to unplanned-for situations (Atkins et. al., 1997).

A primary objective of maintaining separate planning and scheduling algorithms is to utilize existing scheduling algorithms with minimal modification. In particular, the planner should be told whether or not the current plan is schedulable, and if it isn't, which task is judged to be the most costly "bottleneck". If the plan is found schedulable by the resource allocation analyzer then its entire value is redeemed. However, if the plan is unschedulable, the interface module points out a "costly" task to use as a target to remove during subsequent planner backtracking.

We require a common format for plan transmission between planner and scheduler modules. For each planned task $T_i \in T_{total}$, where T_{total} contains all tasks in the plan, the planner outputs the triplet (g_i, P_i, V_i) to the scheduler. g_i is the "guarantee flag" that indicates whether task T_i is guaranteed $(g_i = 1)$ or best-effort $(g_i = 0)$. P_i is the maximum period of T_i required to guarantee

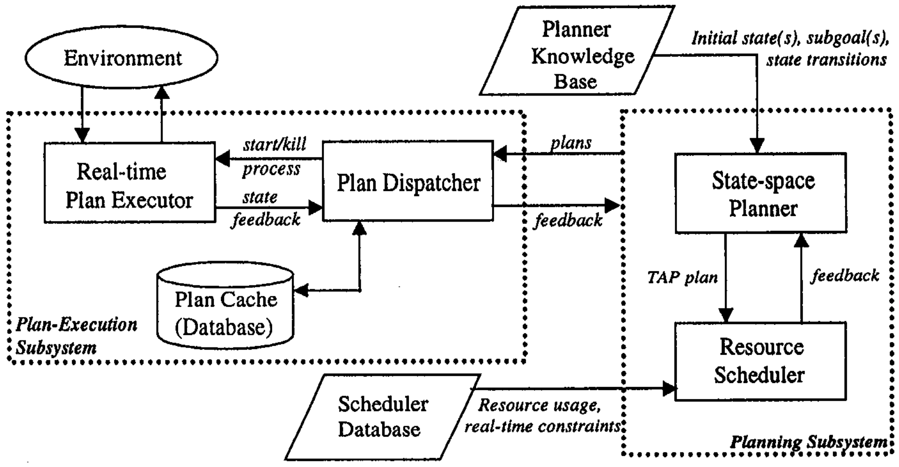

Figure 3: CIRCA-II Architecture.

safety when $g_i = 1$, and V_i is the "priority" value of task T_i. The scheduler attempts to fit all tasks within a real-time cyclic schedule, then it returns a success/failure status along with task resource utilization values. If the scheduler succeeds, the plan is downloaded to the cache. Otherwise, a "bottleneck" task is defined based on the weighted combination of task priority and utilizations, and the planner backtracks to find a schedulable plan as described in (Atkins et. al, 1999).

CIRCA-II Applications

Maintaining close ties with an application domain ensures that a system has practical use and, in parallel, addresses key automation issues within that specific application. CIRCA was originally demonstrated with an office robot and a simulated robot assembly task (Musliner, Durfee, and Shin 1995). More recently, CIRCA-II has been tasked with developing and updating mission flight plans for simulated Uninhabited Aerial Vehicle (UAV) and Uninhabited Combat Aerial Vehicle (UCAV) missions (Atkins 1999). To-date, tasks have involved a fixed set of flight maneuvers and simple weapons control operations, and CIRCA-II performance has been analyzed by assessing its ability to succeed in this hard real-time environment. Although CIRCA-II has succeeded in the development and execution of real-time control plans for fully-automated flight, we are still seeking alternate planning techniques that might permit more expressive numerical representations to facilitate the interface with the complex, nonlinear flight dynamics inherently present for any UAV. We look forward to learning about the latest developments in constraint-based planning and its combination with other algorithms during this workshop.

References

E. M. Atkins, "Plan Generation and Real-time Execution with Application to Safe, Autonomous Flight," *Ph.D. Dissertation,* University of Michigan, 1999.

E. M. Atkins, T. F. Abdelzaher, K. G. Shin, and E. H. Durfee, "Planning and Resource Allocation for Hard Real-time, Fault-Tolerant Plan Execution," in: *Proceedings of the Third International Conference on Autonomous Agents,* Seattle, Washington (1999) 244-251.

E. M. Atkins, E. H. Durfee, and K. G. Shin, Detecting and Reacting to Unplanned-for World States, in: *Proceedings of the Fourteenth National Conference on Artificial Intelligence,* Providence, Rhode Island, (1997) 571-576.

C. Boutilier, T. Dean, and S. Hanks, "Decision-Theoretic Planning: Structural Assumptions and Computational Leverage," *Journal of Artificial Intelligence Research (JAIR), 11(1999) 1-94.*

R. E. Fikes and N. J. Nilsson, "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving," *Artificial Intelligence* 2(3-4)(1971) 189-208.

D. J. Musliner, E. H. Durfee, and K. G. Shin, "World Modeling for the Dynamic Construction of Real-Time Control Plans," *Artificial Intelligence,* 74(1) (1995) 83-127.