Knowledge Representation for Real-time Plan Development

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

ABSTRACT

The Cooperative Intelligent Real-time Control Architecture employs an integrated planning-scheduling system to create plans that guarantee system safety in complex real-time environments. The planner uses state transitions with discrete-valued features to build and schedule plans that maintain system safety while achieving mission goals. The product, a *real-time control plan*, specifies actions plus a resource schedule to guarantee that real-time constraints will be met during plan execution. In this paper, we overview the production and use of a real-time control plan, then we discuss challenges associated with modeling complex dynamic environments in any state-space planning system.

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. Additionally, the level of intelligence actually achieved is a function of planning knowledge accuracy and completeness.

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 safety guarantees in time-constrained environments. The state-space planner specifies a set of actions required to guarantee safety and achieve goals, then the real-time scheduler places the safety-preserving action subset in a cyclic schedule based on their worst-case execution properties and planner-specified task separation constraints. These real-time control plans are then executed as scheduled by a plan execution module.

Instead of discussing the design and operation of CIRCA or CIRCA-II, this paper focuses on how to define the planner/scheduler domain-specific "input" (i.e., knowledge base) and "output" (i.e., real-time control plan) for a complex dynamic domain. First, we present our definition of a *real-time control plan* to illustrate the product we require from the planner. We have used this plan representation in CIRCA/CIRCA-II for experiments in a variety of domains, most recently an Uninhabited Aerial Vehicle (UAV). Next, we describe the input domain knowledge required to actually create such plans. Because systems such as the UAV are typically described using complex nonlinear continuous dynamic equations of motion, developing a sufficient representation for actions and state feature values is a difficult task. We define a *hybrid real-time system* as a domain (e.g., UAV) in which state feature evolution over time is naturally described by a combination of continuous dynamic motion and discrete events. We conclude with a discussion of challenges we face when using CIRCA-II to control such a hybrid system along with open issues to be addressed in future work.

Output: 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

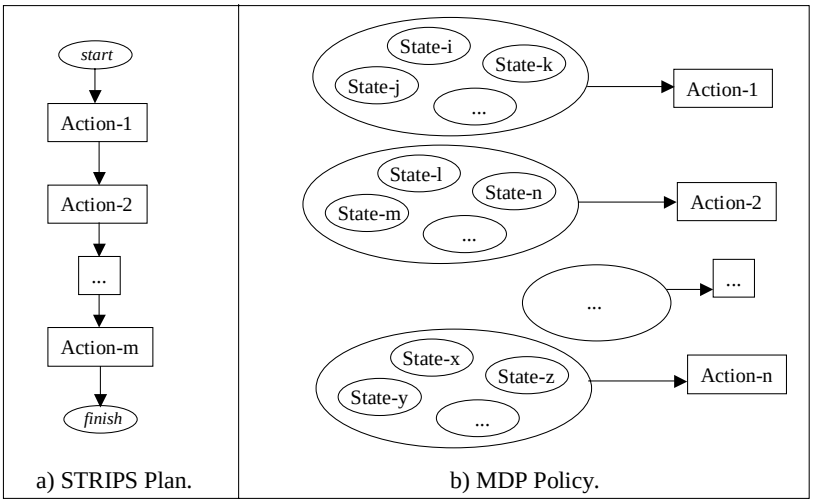

Figure 1: Traditional Plan Types.

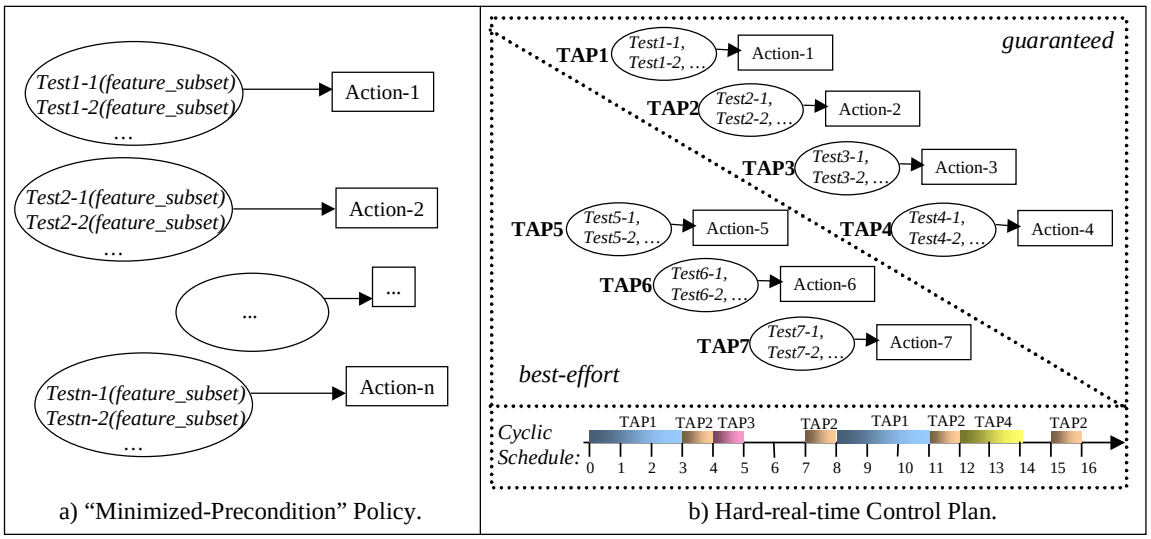

Figure 2: Evolution of the Real-time Control Plan.

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 may always be 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 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 offline plan development, a real-world plan execution system may require

state feature sensing to select appropriate actions. However, dynamic and dangerous environments also require that the complete sense-act loop execute in hard real-time to guarantee failure-avoidance. To define our notion of real-time control plan, consider an MDP policy as the initial representation. Now, to increase execution efficiency for matching the current world state to the correct policy 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 during plan execution. This "minimized precondition" policy representation is shown in Figure 2a.[1]

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 we must structure the plan so that each action's preconditions will 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 and 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.

Input: Knowledge Base

The primary reason to create a *real-time control plan* is to guarantee safe operation while a system/vehicle interacts with a dangerous, dynamic environment. Generally, AI planners are defined by the search strategies and heuristics employed to develop plans. Ultimately, however, plan *quality* is a function of domain knowledge accuracy and completeness.[2] For traditional planners, a knowledge base contains a set of discrete state features and values along with transitions that describe how the world changes as actions are executed. A large but finite set of world states may be enumerated from this information, and associated transitions provide a simple mechanism for describing discrete events (e.g., (STACK A B) from the “Blocks World” (Russell and Norvig, 1995)). This state-space representation has been adopted by planning researchers and has led to systems that reason efficiently about complex high-level tasks and their interactions.

CIRCA and CIRCA-II adopt this basic model for encoding domain knowledge and include transitions to describe both controlled actions and exogenous events. Due to its focus on real-time issues, a CIRCA knowledge base also contains

[1] The decision-tree-based minimal precondition generation algorithm in CIRCA and CIRCA-II prevents the existence of multiple choice points for all states expanded during planning. All other “unexpected” states are identified during plan execution as described in (Atkins, 1999).

[2] This statement holds for static and dynamic (i.e., learned) knowledge, given that plan quality may change over time as new experiences are integrated.

transition timing information, including delays[3] before exogenous events may occur and delays between initiating an action and the associated state change. This information along with worst-case action execution properties allow development of the real-time control plans described above.

Hybrid Systems for Plan Development

Real-time agents typically move within their environment, either on the Earth's surface (e.g., office robot, automobile) or in three-dimensional space (e.g., UAV). Such motion is inherently continuous in nature, but symbolic planning researchers have discovered clever ways of circumventing this problem. For example, A* search during route planning (Russell and Norvig, 1995) allows identification of the shortest path from a location A to location B. The key to the "shortest route" search problem is the existence of a "road map" to define route segments (i.e., actions) and their associated distances and traversal times (i.e., costs).

Figure 3: Excerpt from an IFR Flight Chart.

[3] Transition delays are specified statistically in CIRCA-II.

We have taken an analogous approach in CIRCA-II. For UAV simulations, we have extrapolated the "road map" idea to IFR (Instrument Flight Rules) flight routes. Figure 3 shows an IFR chart for the Miami area, with route segments connecting circled "waypoints".

Unfortunately, many systems operate in environments where clearly-defined routes are not available. For example, off-road driving makes traditional road maps irrelevant, and the upcoming transition to aircraft "free-flight" will limit the utility of IFR charts for route planning. To-date, we have either specified a choice of routes for the CIRCA planner (e.g., segments from an IFR flight chart) or else ignored the routing problem completely (e.g., specified action "go to location x" without deeper investigation). If specific routing details are not required at the state-space planning level (e.g., "go to destination x" is sufficient), then a separate trajectory generation module can be used to supplement the planner. This module would take as input proposed start and destination locations, and would feed back a trajectory, minimum transit time, and maximum resource consumption required for that path. In this fashion, complete real-time plans are developed. This method is desirable because it allows the planner to exclusively reason about its area of expertise: discrete events, including fixed initial and goal destinations (e.g., airports). The use of a dynamics-based trajectory generator is also attractive because it allows tractable consideration of continuous operating regions (e.g., aircraft free-flight) rather than pre-determined fixed routes.

Although a good first step, we believe this simple combination is not generally sufficient for autonomous vehicle operation, specifically in situations where the precise path between initial and destination locations is relevant for high-level deliberation. For example, consider a UCAV (Uninhabited Combat Aerial Vehicle) mission in which the "goal destination" is a missile target, along with additional goal of aircraft survival. Combat survival involves evading enemy weapons when required (e.g., with aggressive maneuvers) followed by a return to the nominal goal path, as was demonstrated with CIRCA-II in simulation (Atkins, 1999). However, survival following battle damage requires knowledge of the degree to which the UCAV can still fly. As a simple example, consider damage that results in engine failure. In this situation, the UCAV will most likely be unable to reach its goal, thus an alternate trajectory based on "best reachable emergency landing site" must be computed. The

continuous variable values representing aircraft best glide ratio, altitude, airspeed, etc. are required to determine the subsequent contingency plan, and the overall "goal" switches from a specific destination to selecting and landing safely at a "reachable" destination. However, the planner does not have sufficient knowledge to distinguish between reachable and unreachable goal destinations without a more information about vehicle dynamics. Conversely, the basic trajectory generator can determine whether any particular site is "reachable" but does not have the knowledge to select "desirable" versus "undesirable" goal destinations.

To address such a situation, we are working to develop a more tightly integrated symbolic planning — trajectory generation approach. Recently, hybrid systems researchers (Egerstedt et. al.) with roots in the controls community have adopted finite state automata for discrete events along with dynamics-based trajectory generation algorithms to effectively develop and follow paths from initial to goal locations. Because such systems were developed from the "bottom up" (i.e., dynamics → controller → trajectory generation → reactive finite automata), the bulk of research to-date has employed a set of mode switch transitions to define how the trajectory should be generated, and goal destinations are either fixed or else generated implicitly by potential field maps. By using the fundamental equations of motion, route segment shapes and traversal times are easily recomputed as vehicle dynamics (e.g., maximum engine power/thrust) and environmental properties (e.g., mud depth for off-road driving) change. We believe the finite automata currently used by trajectory generation experts will guide us in the direction of a more expressive symbolic planning – trajectory generation interface, and we hope to incorporate such techniques as tools for next-generation planning algorithms.

References

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

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.*

M. Egerstedt, T. Koo, F. Hoffmann, and S. Sastry. "Path Planning and Flight Controller Scheduling for an Autonomous Helicopter," in *Hybrid Systems: Computation and Control*, Lecture Notes in Computer Science 1569, Springer-Verlag, Berkeley, CA, 1999.

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.

S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, Prentice-Hall, Englewood Cliffs, New Jersey, (1995).