Working Notes of the AAAI Spring Symposium on Model-Based Validation of Intelligence
Stanford, CA March 2001

Monte-Carlo Simulation for Automatic Synthesis of Verified Real-time Controllers
Extended Abstract

Christopher W. Geib and Robert P. Goldman and David J. Musliner
Honeywell Laboratories
{geib,goldman,musliner}@htc.honeywell.com

Introduction

This extended abstract outlines work in progress on the use of Monte Carlo simulation in the automatic construction of verified real-time controllers. This is part of ongoing work on the CIRCA architecture. As such, we will first briefly outline the CIRCA architecture and its method for constructing controllers and then outline the problem and our solution approach.

Background on CIRCA

The CIRCA architecture is intended to provide real-time, intelligent, verifiable control for autonomously-operating systems. It has been applied to real-time planning and control problems in several domains including mobile robotics, simulated autonomous aircraft, space probe challenge problems (Musliner & Goldman 1997) and controlling a fixed-wing model aircraft (Atkins *et al.* 1998). To build controllers for these kinds of domains, CIRCA must be able both to carry out unrestricted (i.e., intractable) computation and support hard real-time response guarantees.

To this end, the CIRCA architecture has three central components. First, the adaptive mission planner decomposes the mission into more manageable subtasks that can be planned in detail. Second, the *Controller Synthesis Module* (CSM) (see Figure 1) builds and verifies individual controllers for the subtasks provided by the mission planner. Third, operating concurrently with the mission planner and the CSM the *real-time subsystem* (RTS) reactively executes the generated controllers and enforces guaranteed response times. The focus of this paper is the CSM and the efficient generation of verifiable controllers.

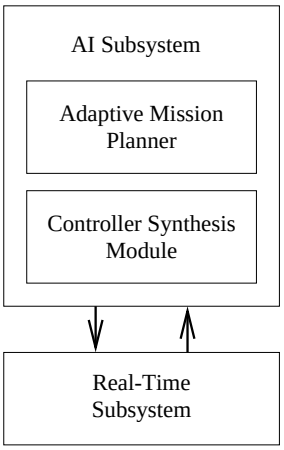

Figure 1: Basic CIRCA architecture.

To build discrete reactive controllers the CSM takes as input a description of the processes in the world and the available control actions represented as a set of state transitions with worst case time characteristics. From this description, CIRCA incrementally constructs a controller and checks it for correctness using a timed automaton verifier.

The real-time controllers that CIRCA builds sense features of the system's state (internal and external) and execute reactions based on the current state. The CIRCA RTS runs a *memoryless* reactive controller. This means the controller synthesis problem is the process of *choosing a control action for each reachable state (feature-value assignment) of the system.* This is not as simple as it sounds. The set of reachable states is not fixed, since the correct choice of control actions can render some states (un)reachable.

Since the CSM only builds *safe* controllers, a critical issue is making failure states unreachable. This is ac-

complished by the process of *preemption*. A transition t is preempted in a state s if and only if some other transition t' from s must occur before t could possibly occur. In controller synthesis, the CSM preempts a failure state by choosing a control action for the prior state that is fast enough to guarantee that it will occur before a transition to the failure state. In the abstract, the controller synthesis algorithm is as follows:

1. Choose a reachable state (at the start of controller synthesis, only the initial states are reachable).
2. Choose a control action (an action or a reliable temporal) for that state.
3. Invoke the verifier to confirm that the (partial) controller is safe.
4. If the controller is *not* safe, use information from the verifier to direct backtracking.
5. If the controller *is* safe, recompute the set of reachable states.
6. If there are no unplanned reachable states (reachable states for which a control action has not been chosen), terminate successfully.
7. If unplanned reachable states remain, loop to step 1.

This is only a sketch of the actual controller synthesis algorithm. The interested reader is referred to (Musliner, Durfee, & Shin 1995; Goldman *et al.* 1997) for a more detailed description of the algorithm and related issues. However, even this high level algorithm is sufficient to illustrate the problem we are addressing.

The Problem

The problem with this algorithm lies in the requirement that it plan actions for *all* reachable states of the world. This actually presents two constraints on controller synthesis: bounded rationality and bounded reactivity.(Musliner, Durfee, & Shin 1993)

First consider bounded rationality. Even within the small subproblems that the mission planner produces there may still be more world states than it is reasonable for the system to plan for. There may also be possibilities that are unreasonable to plan for. Consider the case where there are two events, each of which is unlikely. While it may be reasonable for CIRCA to plan for each of these unlikely events individually, is it reasonable for CIRCA to plan for the case where *both* of these very unlikely events happen?

Second, even if we have the time to build a controller for all of the world states, this controller may not be able to make the kind of guarantees that we want. There may simply be too many states for the controller to monitor. In this case, while the controller was looking for one very unlikely event a critical transition to failure is happening. Thus even if the controller synthesis algorithm could determine actions for each of the states, trade offs would have to be made. The controller would not have time to monitor for all of the states and still maintain its real-time guarantees.

Simply put, in some domains, we don't have the time to build or even execute controllers that cover all the possible states of the world. The question then is, if we cannot consider all of the world states, which should we consider? Given the controller synthesis algorithm, we can view this as two simple algorithmic questions: 1) How should we order the reachable states of the world to plan actions? 2) When should we stop considering states of the world?

The intuitive answers to these questions are: 1) plan for the most likely states first, and 2) stop when the cost of planning for the state exceeds the expected value of having the plan.

Solution Details

With this approach, the central problem becomes one of determining which world states are the most likely given the current controller. We are not the first people to look at this problem of ordering states that CIRCA considers on the basis of the state's likelihood. Atkins, Li and others (Li *et al.* 1999) have been looking at finding a closed form solution for calculating the state probabilities. Generating these closed form solutions requires making simplifying assumptions, and results in approximate solutions.

We believe that Monte-Carlo simulation of the world and controller will have a number of advantages over the closed form solution. Monte-Carlo simulation will allow us to more accurately model the semantics of the real-time executive. It will allow us to move to incrementally computing the state probabilities reducing the runtime of the computation. Finally we believe Monte-Carlo simulation will produce a more accurate approximation of the probabilities in the case of some transition types.

As in Li's work, Monte-Carlo simulation requires a transition rate distribution for each transition. Rather than having a single worst-case time for each transition, we now assume we have a function that provides the probability of the transition as a function of time. The simulation is done by starting in the "initial state" and running fixed time length simulations of the system. We will simulate both the world's transitions and the actions of the controller to determine two statistics.

First, we compute the "time in state". That is, by doing a timed simulation of the world and controller we will be able to collect the amount of time spent in each state. Since we know the number of simulation runs and the total time of each run, we can compute the time in each of the frontier states and divided this by the total simulated time giving a time-weighted probability of being in one of the unplanned states. This statistic is used to order the unplanned states for consideration by the CSM

Second, we compute an absolute probability of ever reaching a state. For each run we mark the states that the simulation traverses. Each state is marked only once per simulation run (even if the simulation enters the state multiple times). Again by dividing by the total number of runs we are provided with the probability that a run will ever enter a given state.

The simulation algorithm is:

1. Start in the initial state.
2. If the state has been planned for
   - Compute the time for the controller to act.
   - For each other transition out of the state, sample from the transition rate distribution to determine how long it will take for the transition to happen.
   - Choose the action or transition with the minimum time to transition and execute it.
   - Remove the time for the transition from the clock.
3. Else If the state is unplanned for or is a failure state
   - Mark the state as visited on this run.
   - Add the time remaining in the run to the state's counter
   - End the run.
4. If there is more time loop to step 2

A large number of individual simulation runs are combined by the running counters and the final probabilities are generated by dividing the cumulative statistics against the known totals. These statistics can then be used to order the unplanned states to consider the most likely first. Further, if the absolute probability of entering a failure state rises above a chosen threshold we trigger backtracking in the controller synthesis algorithm.

As a whole, this approach will move CIRCA away from absolute guarantees that no failure state will be reached to probabilistic guarantees of no failure. By the time of the symposium we hope to have results of this approach to report.

Acknowledgments

This material is based upon work supported by the Space and Naval Warfare Systems Center – San Diego under Contract No. N66001–00–C–8039. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA, the U.S. Government, or the Space and Naval Warfare Systems Center – San Diego.

References

Atkins, E. M.; Miller, R. H.; VanPelt, T.; an d W. B. Ribbens, K. D. S.; Washabaugh, P. D.; and Bernstein, D. S. 1998. Solus: An autonomous aircraft for flight control and trajectory planning research. In *Proceedings of the American Control Conference (ACC)*, volume 2, 689–693.

Goldman, R. P.; Musliner, D. J.; Krebsbach, K. D.; and Boddy, M. S. 1997. Dynamic abstraction planning. In *Proc. National Conf. on Artificial Intelligence*, 680–686.

Li, H.; Atkins, E.; Durfee, E.; and Shin, K. 1999. Resource allocation for a limited real-time agent using a temporal probabilistic world model. forthcoming.

Musliner, D. J., and Goldman, R. P. 1997. CIRCA and the Cassini Saturn orbit insertion: Solving a prepositioning problem. In *Working Notes of the NASA Workshop on Planning and Scheduling for Space*.

Musliner, D. J.; Durfee, E. H.; and Shin, K. G. 1993. CIRCA: a cooperative intelligent real-time control ar-

chitecture. *IEEE Trans. Systems, Man, and Cybernetics* 23(6):1561–1574.

Musliner, D. J.; Durfee, E. H.; and Shin, K. G. 1995. World modeling for the dynamic construction of real-time control plans. *Artificial Intelligence* 74(1):83–127.