Appears in Working Notes of the AAAI Fall Symposium on Model-Directed Autonomous Systems
Cambridge, Massachusetts, November 1997

The Link Between Distributed Planning and Abstraction

David J. Musliner, Mark S. Boddy, Robert P. Goldman, Kurt D. Krebsbach
Automated Reasoning Group
Honeywell Technology Center
3660 Technology Drive
Minneapolis, MN 55418
{musliner,boddy,goldman,krebsbac}@htc.honeywell.com

Abstract

The Distributed CIRCA (D-CIRCA) project is focused on developing planning technology to concurrently build and execute real-time control plans for multiple cooperating autonomous agents. In this paper, we demonstrate how the goals of the D-CIRCA program relating to distributed, cooperating agents motivate our recent work on Dynamic Abstraction Planning (DAP). We contend that problems of incomplete knowledge, limited communication bandwidth, and state-space explosion can all be addressed by abstraction. We present a high-level description of the DAP technique and conclude by outlining several issues for future work.

Introduction

The recent explosion of interest in "agents" has focused more attention than ever before on Distributed AI and the problems of multi-agent intelligent systems. We argue that some of the problems inherent in distributed multi-agent systems can be reduced to single-agent abstraction issues. We then present a new technique, Dynamic Abstraction Planning (DAP) (Goldman *et al.* 1997), that automates decisions about which elements of a domain representation to abstract. Due to space constraints, we cannot include the technical details of DAP here, but instead outline the technique at a conceptual level. See (Goldman *et al.* 1997) for details on DAP. We conclude with a discussion of several remaining issues and ideas about abstraction, distribution, and D-CIRCA.

Background: The D-CIRCA Project

Early work on the Cooperative Intelligent Real-Time Control Architecture (CIRCA) (Musliner, Durfee, & Shin 1993; 1995) focused on building an intelligent control system for a single agent, allowing that agent to provide real-time response guarantees while also using complex planning algorithms. The resulting architecture, illustrated in Figure 1, combines planning and scheduling modules that build guaranteed, executable plans with a real-time execution subsystem for predictably executing and enforcing the planned behavior.

The current Distributed CIRCA (D-CIRCA) project seeks to extend these concepts of guaranteed safety and predictable performance into multi-agent domains such as cooperating teams of autonomous aircraft (see Figure 2). D-CIRCA agents will communicate to allocate tasks and build executable real-time plans that achieve overall team goals. D-CIRCA will enforce both the *logical correctness* of coordinated multi-agent behaviors and the *timeliness* of those behaviors, ensuring that coordinated actions achieve their goals and preserve overall system safety. While executing their plans, D-CIRCA agents will respond to ongoing events in real-time, invoking safety-preserving reactions and/or triggering dynamic replanning tailored to the current context.

This dual capability is distinctly different from typical distributed AI systems. Most DAI research evaluates collaboration and coordination methods based primarily on logical correctness and solution efficiency, ignoring the issues of behavioral synchronization and reaction timing required for *guaranteed* performance by a multi-agent system. Systems based on the D-CIRCA architecture can be applied to mission-critical distributed domains with confidence, and will provide advance feedback when the available multi-agent resources are insufficient to deal with the anticipated behavior of the domain.

Linking Distribution and Abstraction

Consider the following aspects of a multi-agent system, when viewed from the perspective of a single agent within that system:

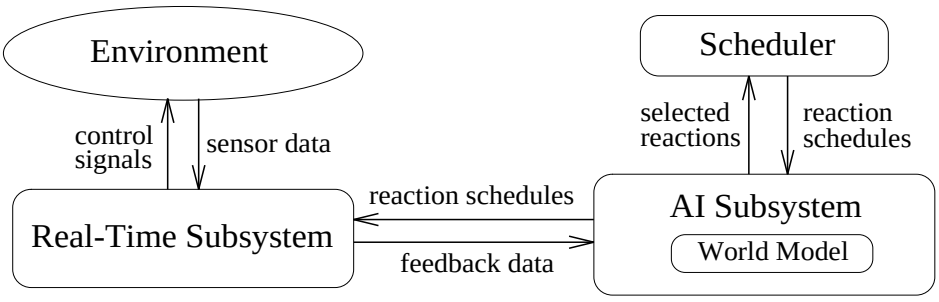

Figure 1: CIRCA combines concurrent planning, scheduling, and real-time execution.

Partial Information — The agent cannot know everything about its environment or about the internal state of other active agents.

Incomplete Control — The agent cannot perfectly control all aspects of the domain or the actions of other agents.

Limited Time — The agent has a limited amount of time to reason about and react to various situations to maintain safety and achieve its goals.

The original CIRCA system was designed to address the latter two issues explicitly, but made strong assumptions about the world being "fully observable." To make guarantees that CIRCA would detect and react to all environmental hazards, it was assumed that the system could sense and deliberate about all modeled world features. In a distributed environment, there are several reasons to use a partially-observable model, including:

Local Views— Distributed systems are often motivated by the need for different agents to have heterogeneous sensors, locations (and hence fields of view), and other distinguishing capabilities.

Limited Communication— Agents cannot share all of their knowledge.

Bounded Rationality — Even if they could share all of their knowledge, agents cannot afford the state-space explosion associated with reasoning about the complete domain model.

An important observation motivating our research on Dynamic Abstraction Planning is that *a single agent capable of making performance guarantees based on an abstracted world model can successfully address each of the above distribution issues.*

By "abstraction," we mean the deliberate omission of certain pieces of detailed information from an agent's world model, and hence from its consideration. An abstracted world model is essentially indistinguishable from an incomplete world model, a local view, a partially-shared model, or a "partially considered" model. Distributed observability reduces to partial observability for a single agent, so our first challenge is to build a new CIRCA that retains its unique guaranteed performance characteristics while using abstract world models.

Dynamic Abstraction Planning

The original CIRCA planning system builds reaction plans based on a world model and a set of formally-defined safety conditions that must be satisfied by feasible plans (Musliner, Durfee, & Shin 1995). CIRCA plans by generating a nondeterministic finite automaton (NFA) from user-supplied transition descriptions that implicitly define the set of reachable states. Beginning from a set of designated start states, the planner enumerates the reachable states and assigns to each state either an action transition or `no-op`. Actions are selected to *preempt* transitions that lead to failure states and to move towards states that satisfy as many goals as possible. System safety is guaranteed by planning action transitions that preempt *all* transitions to failure, making the failure state unreachable (Musliner, Durfee, & Shin 1995).

The DAP technique omits detail from the state representation, reducing both the size of the state space that must be explored to produce a plan, and the size of the resulting plan itself. The DAP method provides three important new advantages:

1. The abstraction method does not compromise safety-preserving guarantees: the world model used for planning is reduced, but not in ways that affect the system's ability to make rigorous statements about the safety assurances of plans it is building.
2. The method is fully automatic, and dynamically determines the appropriate level of abstraction during the planning process itself.
3. The method uses different levels of abstraction in

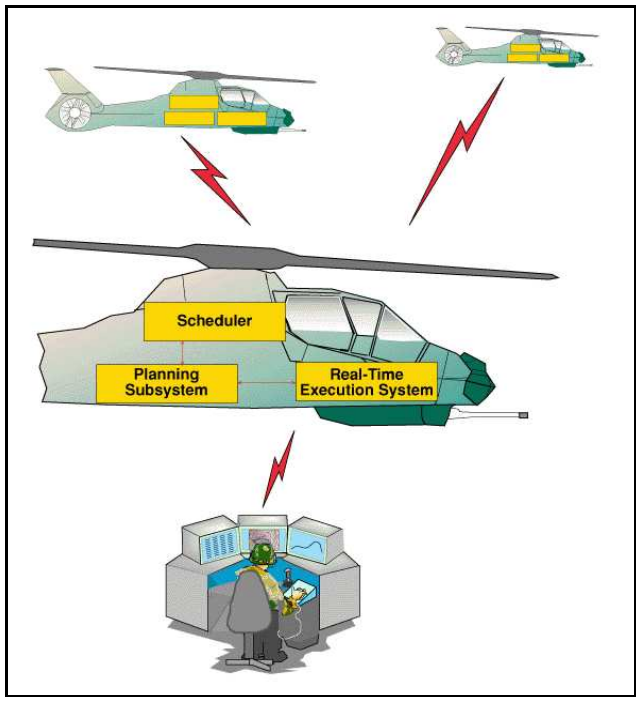

Figure 2: Multiple D-CIRCA agents control a team of autonomous rotorcraft under supervisory control.

different parts of the search space, individually adjusting how much detail is omitted at each step.

The intuition behind DAP is fairly simple: in some situations, certain world features are important, while in other situations those same features are not important. An optimal state space representation would capture only the important features for any particular state. By ignoring certain features, the planner can reason about *abstract states* that correspond to *sets* of "base-level" states, and thus can avoid enumerating the individual base-level states.

Thus, CIRCA augmented with DAP works as follows: rather than always using all of the available features to describe world states, we let the planner dynamically decide, for each new world state, the level of description that is necessary and desirable. DAP allows a planner to search for useful state space abstractions at the same time it is searching for a plan. Of course, during the planning process the system might realize that an abstract state that has already been reasoned about is not sufficiently detailed. For example, this occurs when the state description is not sufficiently refined to indicate whether a desirable action can, in fact, be executed (e.g., because the abstract state description does not specify values for all of the features in the action's preconditions). In such situations, the planner must be able to dynamically increase the precision of that abstract state description by including one or more of the omitted features. We call this process of adding detail a "split" or "refinement." Figure 3 illustrates a split which reduces the set of states from which failure is reachable, so that further planning can select actions to preempt failure.

The implemented DAP planner begins with a maximally-abstracted world model and iteratively either splits an abstract state or plans an action for a state until it has arrived at a feasible plan that avoids failure and achieves the system goals. When all states have an action specified, planning is complete.

Other Issues and Future Directions

In this section, we touch briefly on several issues that have arisen in the course of this work, including extensions involving:

- The world model manipulated by the planner.
- Interactions between the planner and the Scheduler.
- Distributed planning, and planning for distributed execution.

World Model

CIRCA's original world model relies on a set of simplifying assumptions that were tailored to make planning for real-time guarantees feasible. As we consider CIRCA for a wider set of applications, we are identifying useful ways of relaxing these assumptions and

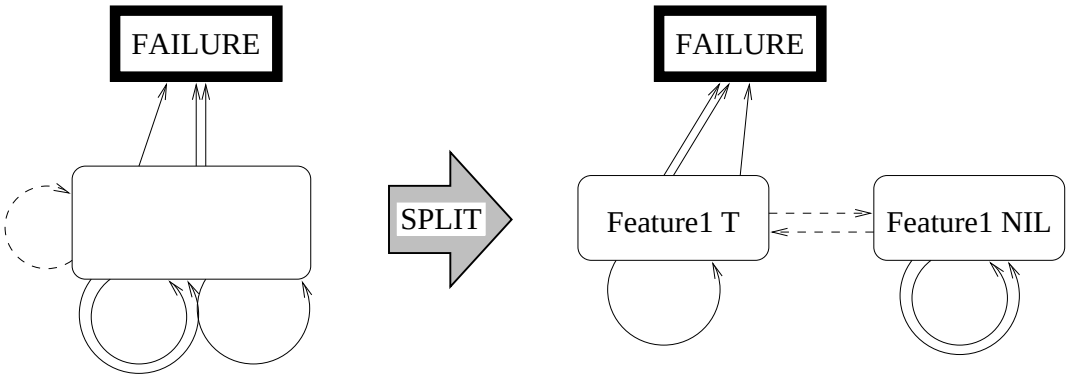

Figure 3: The DAP splitting operation.

extending the model.[1] For example, while CIRCA has the unusual ability to plan against external threats, it cannot (yet) take advantage of reliable *processes* in making its performance guarantees. The primitive actions that CIRCA currently uses in its guaranteed real-time plans are assumed to be atomic, of short duration, and executed only sequentially. Capturing reliable actions and external processes that have a longer duration will allow the system to build plans where concurrent actions and processes are involved in preempting failures. To draw an example from Gat (1996), suppose that a device needs to be warmed up for a period of time before it is used. To build a plan that relies on this device to prevent a failure, CIRCA needs to be able to represent a reliable process (i.e., warming up), with an upper bound on its duration. This representation will allow the planner to calculate a time by which the expected effect will definitely have happened. The existing temporal model already takes into account some upper bounds— the delays on primitive actions. We plan to expand the model to include new, *reliable* temporal transitions with upper bounds on their time of completion, together with state-encoding of the progress of those processes. Our initial investigations indicate that this modification will not require extensive changes to the planning engine.

Planner/Scheduler Interactions

The interactions between the CIRCA state-space planner and the Scheduler module are crucial to the effective operation of the system. CIRCA plans are safely executable only when the Scheduler can construct a schedule of planned reactions (*Test-Action Pairs* (Musliner, Durfee, & Shin 1993)) such that the minimum delays required by the planner are guaranteed by the schedule. In the current system, the planner builds a complete plan using assumptions about achievable action delays, and then the plan is handed off to the Scheduler, which tries to build a schedule. If it fails, the planner backtracks to try a different plan.

There are several ways to improve this situation. The first and simplest way is to make the interaction between planner and Scheduler more frequent, so that the Scheduler is incrementally producing schedules as the planner makes action choices. Tightly coupling scheduling with the planning decisions will allow the system to detect infeasible action choices earlier, avoiding the inefficient backtracking caused by the old batch mode. Second, the planner could be extended to include, as an explicit choice, the addition of a transition solely to prune the size of the final reachable state space and thus reduce the number of actions that must be scheduled. While the current planning heuristics choose the prune the search space for several reasons, they do not yet do so simply on the basis of making the scheduling process easier. Third, the planner could make action choices based on scheduling concerns; for example, it could prefer actions that are already in the schedule, rather than forcing the addition of a new test and action.

McVey *et al.* (1997) have recently investigated the introduction of a Schedule Manager to make more explicit the strategies and interactions that the planner could have with the Scheduler. One advantage of the Schedule Manager concept is the ability to offload iterative search for certain types of structured plan relaxations, allowing the planner to focus on model-based reasoning that the Scheduler cannot accomplish. Much work remains in defining useful feedback that can help

[1] One strong influence in this area has been Erann Gat's paper "News From the Trenches: An Overview of Unmanned Spacecraft for AI" (Gat 1996). See (Musliner & Goldman 1997) for more discussion related to this paper.

guide the planner in refining its plans when the Scheduler cannot find a feasible solution.

Distributed Planning

Building a system in which CIRCA agents communicate and interact as they construct plans and take actions involves many complex issues. For example, actions may be synchronized through commonly-visible world features, or through explicit communication regarding a state transition that would otherwise be unobservable. This communication must be added to the schedule during the planning process.

There are several ways to approach concurrent planning. Unsynchronized planning would involve CIRCA agents planning independently, perhaps knowing what the others are doing, perhaps not. In a sufficiently loosely-coupled environment, this might be feasible. Synchronized planning is a more powerful (and more difficult) approach. In synchronized planning, CIRCA agents communicate about their respective plans as they are being built. For example, CIRCA agents might ask for a communication action to be added (as above), or for a particular action not to be taken. Or, CIRCA agents might ask that a given action be delayed to enforce an ordering on actions taken by different agents. Our current intent is to implement a limited form of synchronized planning.

Acknowledgments: This work was supported by the Defense Advanced Research Projects Agency under contract DAAK60-94-C-0040-P0006.

References

Gat, E. 1996. News from the trenches: An overview of unmanned spacecraft for AI. In Nourbakhsh, I., ed., *AAAI Technical Report SSS-96-04: Planning with Incomplete Information for Robot Problems.*

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.

McVey, C.; Durfee, E. H.; Atkins, E. M.; and Shin, K. G. 1997. Development of iterative real-time scheduler to planner feedback. In *Proc. Int'l Joint Conf. on Artificial Intelligence (to appear).*

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 (to appear).*

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