Synthesizing Protection Monitors from Causal Structure

Glen A. Reece*
Department of Artificial Intelligence
University of Edinburgh
80 South Bridge
Edinburgh EH1 1HN, Scotland
G.Reece@ed.ac.uk

Austin Tate
Artificial Intelligence Applications Institute
University of Edinburgh
80 South Bridge
Edinburgh EH1 1HN, Scotland
A.Tate@ed.ac.uk

Abstract

Protection monitors synthesized from plan causal structure provide execution systems with information necessary to detect potential failures early during execution. By detecting early, the execution system is able to address these problems and keep the execution on track. When the execution system finds that the necessary repairs are beyond its capabilities, early detection gives the planning system additional time to suggest a repair. This paper discusses how protection monitors are synthesized directly from plan causal structure, and the options which are available to an execution system when protection violations occur.

Introduction

Experience in planning for execution in realistic domains tells us that we cannot consistently generate plans that will succeed because of the uncertainty which is inevitably present. A planning system is not able to determine all possible interventions *a priori*, and the model of the world which it uses to base assumptions on is destined to be out of date. The situated agent which must carry out the plans generated by the planning system also has uncertainty to contend with. It is neither in total control of the environment in which it is situated, nor necessarily alone. Thus, we must be able to manage uncertainty during execution.

Many execution systems take a simplistic approach to monitoring. That is, they simply try to test preconditions and postconditions of actions to determine when execution is not going to plan as a way of managing this uncertainty. This is a reasonable mechanism for doing so in some domains. However, as it has been shown (Doyle, Atkinson, & Doshi 1986), such an approach would not be reasonable in domains where actions take long periods of time to complete. For example, if an action N-1 completed successfully at time t1 and its effects were required by an action N-19 at t1+7days, then the fact that some event(s) had taken place which nullified one or more of the required effects between t1 and t1+7days would not be detected until the preconditions of the action were verified at t1+7days.

In order to detect and resolve such problems, an execution system must actively monitor actions during their execution and subsequently up to the point where their conditions are required. Passive monitoring, or only checking the pre- and post-conditions, informs the execution system not to attempt to perform certain actions due to failed preconditions, or that certain actions have not produced all of their expected effects so something else must be done. Active monitoring informs the execution system on how a particular action is progressing to achieve its effects. However, active monitors as they have been defined (Sanborn & Hendler 1988; Hart, Anderson, & Cohen 1990) in the past do not address the whole monitoring picture. In addition to monitoring the progress of an action, we need active monitors to detect protection violations during protection intervals. Such violations typically manifest themselves as later failures thus, possessing the ability to detect them provides an early warning for potential failures.

In order to comprehensively provide execution monitoring in complex and dynamic environments an execution agent must be able to: (1) monitor preconditions and essential postconditions, (2) actively monitor for protection violations, (3) actively monitor situations which are known to cause failures, (4) actively monitor for potential beneficial opportunities, and (5) actively monitor the progress of an action during its execution. Our focus in this paper is to detail how protection violations can be detected early during execution and how protection monitors are synthesized directly from plan causal structure.

In the next section, we describe what is meant by the causal structure of a plan. Section presents a model of plan execution which is based on having casual structure information available. Section discusses the process of how execution monitors are synthesized from causal structure and how they become activated, and in Section we discuss what happens when a vi-

*This research was sponsored by an Augmentation Award for Science and Engineering Research Training from the US Air Force Office of Scientific Research under contract F49620-93-1-0436. The parent contract is part of the ARPA-Rome Laboratory Planning Initiative.

olation of a condition is detected during a protection interval.

Plan Causal Structure

Causal structure is a high level representation of information about a plan which states the relationship between the purposes of actions with respect to the goals or sub-goals they achieve for some later point in the plan. This information may be used by a planner to detect and correct conflicts between solutions to sub-problems when higher level plans are refined to greater levels of detail.

Various forms of causal structure representations are found in most planning systems for a variety of purposes. During plan generation its main use is for interaction detection and correction. The representations include Goal Structure (or GOST) (Tate 1977; Currie & Tate 1991), causal links (McAllester & Rosenblitt 1991), protection intervals (Sussman 1975), and plan rational (Wilkins 1984) to name a few.

During the generative planning process a causal structure table is maintained to record what facts have to be true at any point in the plan network and the possible "contributors" that can make them true. A contributor in this sense is a node in the plan network whose effects are required elsewhere in the network to satisfy a condition of another node. The planning system is able to plan without choosing one of the (possibly multiple) contributors until it is forced to by interaction of constraints. The causal structure is used to detect important interactions (ignoring unimportant side effects) and can be used to find the small number of alternative temporal constraints to be added to the plan to overcome each interaction. This "Question Answering" procedure (Tate 1977) is the basis for work by Chapman on the Modal Truth Criterion (Chapman 1987). Multiple interactions arising at the same time further restrict the possible solutions and a minimal set of temporal constraints can be proposed.

We believe that this plan causal structure can be extended to represent information which an execution agent can use to effectively monitor action execution and detect protection violations (Tate 1984). Causal structure statements represent precisely the outcome of any operation which should be monitored (i.e., protected). If lower level failures can be detected and corrected while preserving the stated higher level causal structure, the fault need not be reported to a higher level (e.g., a planning system).

A Model of Plan Execution Monitoring Based on Causal Structure

An execution agent is given a plan generated by a planner together with information on what the individual plan steps achieve, by what time, and for which subsequent tasks (the causal structure). It must supervise the execution of actions (based on a capabilities data base which might be trivial or quite complex in nature). It should use any available monitoring capabilities to monitor the execution of each action to ensure (as far as possible) that it achieves its purpose(s).

When failures occur, recovery steps may be taken which might be of various types:

- Recovery procedures for the effector chosen (e.g., reset and repeat).
- Recovery procedures for the action type chosen (e.g., generic procedures for ensuring that an action can be successfully accomplished by passing it to a special purpose effector or skilled supervisor).
- Recovery procedures for the particular failed action (e.g., by procedural methods, etc.).

Recovery on failures can be simple or complex depending on the local intelligence of the effectors chosen, the closeness of coupling of actions in the domain, the predictability of error outcomes, etc. When a failure is found which cannot be locally recovered from within the given causal structure constraints (of required outcomes, resource usage or time limits), the execution agent must prepare a statement of the failure and changed plan circumstances to communicate back to the planner (which can then be used to suggest a plan repair).

As shown in Figure 1 (from (Tate 1984)), an activity can be executed as soon as all the incoming causal structure requirements are satisfied (by any potential "contributor" if there are several alternatives). A decision on the allocation to a particular effector must then be made. The activity and any associated constraint information is then passed to the effector.

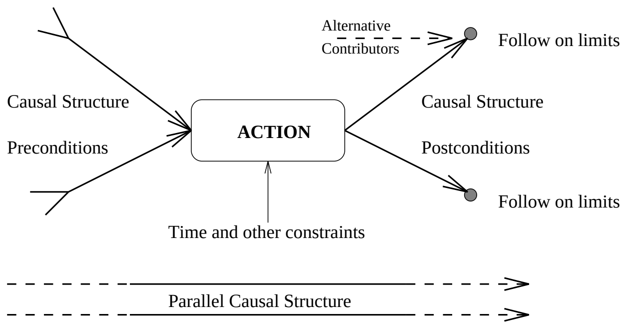

Figure 1: Execution from the viewpoint of a single action.

The relevant effector executes the action and its controller must report when the activity is completed. Time-out conditions related to the time limits for the follow on actions to the causal structure outcomes are used to prevent the system hanging up on effector controller failure.

The condition monitor is triggered to check all associated causal structure outcomes of the activity. This same model of execution and condition monitoring applies when the activity involves the use of a sensor to capture information needed at some point in the plan. The causal structure outcomes in such a case my contain variables which will be bound to definite values when the condition is checked.

If failures occur, local recovery is possible (by either the effector or by using procedural methods accessible to the execution agent) within the given resource or time limits set for the follow on activities resultant on each monitored outcome. The parallel causal structure (i.e., postconditions of actions before the failed activity which are required later in the plan) provides a guide to the local recovery system on what should be preserved if the local recovery is to avoid interference with other important parts of the existing plan. Any interference with such parallel causal structure should be reported to the execution agent as it must be re-considered by the planner to work out the actual effect on the plan.

Monitor Synthesis and Activation

The model described in Section , is the basis of the execution monitoring functionality of the Reactive Execution Agent (REA) design proposed in (Reece 1992). The REA is designed to handle multiple, simultaneously executing plans and to possess the ability to monitor conditions between plan executions. This design utilizes a communication protocol called Inter-Agent Communication Language (IACL) to transmit information between the execution agent and the planner. Tasks are specified by a planning system (in the form of synthesize messages) to the REA which are then carried out using a more detailed model of the execution environment than is available to the planner. The REA executes the plans by choosing the appropriate activities to achieve the various sub-tasks within the plans, using its knowledge about the particular resources under its control. It communicates with the environment in which it is situated by executing the activities within the plans and responding to failures fed back from the environment. Such failures may be due to the inappropriateness of a particular activity, or because the desired effect of an activity was not achieved due to an unforeseen event.

```
(causal-structure
 ((CSTR-7  (AT GT1) DELTA (N-6-1) (N-3))
  (CSTR-24 (AT GT2) DELTA (N-5-1) (N-3))
  (CSTR-6  (STATUS GT1) AVAIL (N-6-1) (N-3))
  (CSTR-23 (STATUS GT2) AVAIL (N-5-1) (N-3))
  (CSTR-14 (AT C5) DELTA (N-8) (N-3))
  (CSTR-9  (STATUS GT2) ABYSS (N-4-2) (N-4-1))
  (CSTR-8  (AT GT2) ABYSS (N-4-2) (N-4-1))
  (CSTR-18 (STATUS GT2) ABYSS (N-4-4) (N-4-1))
  (CSTR-17 (AT GT2) DELTA (N-4-4) (N-4-3))
  (CSTR-13 (AT GT2) DELTA (N-8) (N-4-3))
  (CSTR-12 (AT GT2) DELTA (N-8) (N-4-4))
  (CSTR-16 (STATUS GT2) BARNACLE (N-5-2) (N-5-1))
  (CSTR-15 (AT GT2) BARNACLE (N-5-2) (N-5-1))
  (CSTR-26 (STATUS GT2) BARNACLE (N-5-4) (N-5-1))
  (CSTR-22 (AT GT2) DELTA (N-4-1) (N-5-3))
  (CSTR-25 (AT GT2) DELTA (N-5-4) (N-5-3))
  (CSTR-21 (STATUS GT2) AVAIL (N-4-1) (N-5-4))
  (CSTR-5  (STATUS GT1) CALYPSO (N-6-2) (N-6-1))
  (CSTR-4  (AT GT1) CALYPSO (N-6-2) (N-6-1))
  (CSTR-20 (STATUS GT1) CALYPSO (N-6-4) (N-6-1))
  (CSTR-19 (AT GT1) DELTA (N-6-4) (N-6-3))
  (CSTR-11 (AT GT1) DELTA (N-8) (N-6-3))
  (CSTR-10 (AT GT1) DELTA (N-8) (N-6-4)))))
```

Figure 2: Causal structure information from a synthesize message

When the planner has generated a plan it intends to execute, it sends a synthesize message that contains the actions of the plan, commitment information, ordering constraints, and plan causal structure. This information is then used by the REA to synthesize a Task-Directive object which it can execute. The causal structure information contained in a synthesize message (see an example in Figure 2[1]) is used by the REA to synthesize monitor objects which actively monitor for protection violations during the execution of the Task-Directive.

Each CSTR, or causal structure record, provides the execution agent with important monitoring information as follows:

(<Tag> <Pattern> <Value> <R-Node> <C-Node(s)>)

The *tag* provides a reference to the planning system for use when a failure has occurred which cannot be addressed locally by the REA. The *pattern* specifies the exact property which is to be protected for the range *C-Node(s)* to *R-Node*. The *R-Node* is the node in the plan network which *requires* the pattern to have the *Value*, and the *C-Node(s)* field specifies one or more *contributors* of the value.

The causal structure record contains all the information necessary to synthesize a monitor object. The mapping of information contained in the CSTR to the monitor object is shown in Figure 3.

The complexity of protection monitors comes from deciding when the monitor should be active. Basically, a protection monitor is active only while the REA *intends* to execute the associated Task-Directive to which it belongs. The monitors become active immediately upon synthesis of the Task-Directive and are removed when either they expire or all actions of the Task-Directive have been executed. During the "life" of a protection monitor it could find itself in one of three states—activated, inactivated, and expired.

[1]This example is from a logistics transportation domain in which people are moved by ground transports (GTs) between towns on the fictional island of Pacifica (Reece & Tate 1993; Reece *et al.* 1993)

Monitor-Slot	Value	CSTR-info
NAME	MONITOR-25	
TAG	CSTR-25	TAG
TASK-DIRECTIVE	#<TD-1>	
SCHEMA	#<NODE-9>	
EXPECTED-VALUE	DELTA	VALUE
KNOWN-CONTRIBUTORS	(8)	C-NODE(s)
BEING-MONITORED	(AT GT2)	PATTERN
RANGE-START	8	
RANGE-END	9	R-NODE

Figure 3: Monitor object created from CSTR-25

When a synthesize message is received by the REA and a Task-Directive object is being synthesized, any causal structure information is used to synthesize protection monitors and associated with that Task-Directive. All protection monitors are initially in the inactivated state when they are synthesized. For example, the causal structure information shown in Figure 2 is used to synthesize protection monitors for a 15 node plan giving the coverage shown in Figure 4. A single monitor (e.g., M25) is synthesized for each causal structure record (e.g., CSTR-25).

When the REA begins to execute any Task-Directive from its agenda the state of the monitors can change. What a protection monitor object is concerned about is when the REA's world model is updated with new information. When updates occur a set of activation-rules are applied to each protection monitor to determine if it should change its state. These rules determine whether a monitor is to be activated or has expired.

Protection monitors become activated when execution has progressed to the point where the monitor's range is valid. Once a monitor is in the activated state it remains in that state until either what is being monitored by the object does not have the value it expected (in which case it is a violation), or execution has progressed past the range-end of the monitor (in which case it has expired). Once a monitor has expired it is removed from contention and is no longer considered when the activation-rules are applied. Protection violation will be further discussed in Section .

An advantage of this approach to activation is that violations can be detected across Task-Directives so the planner can improve the probability that the assumptions it makes about the future will be valid by protecting them. That is, if the planner "knew" that it was, for the time being, only going to execute a portion of the plan which it was working on, it could submit that plan to the REA with causal structure that would essentially protect the effects of that plan until the remaining portion of the plan could be executed. This requirement stems from the need to plan and/or execute particular "phases" of plans only to specified levels when *authority* is given to do so (Tate 1993). For example, we see in Figure 5, that the planner has a plan that it wishes to execute. However, in the first instance it is only given authority to execute actions N-1 and N-2. So, it sends a synthesize message to the

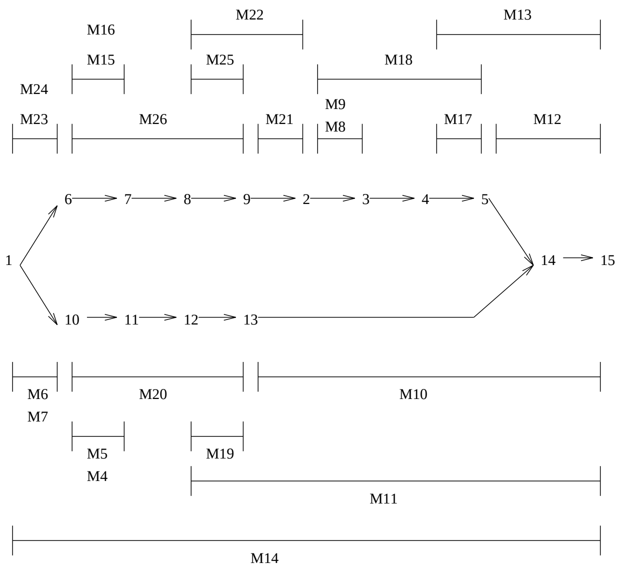

Figure 4: Causal Structure Coverage of a Plan by Protection Monitors

REA with causal structure telling it that condition-1 would be introduced by action N-1 which is required by action N-2 (and some action N-3 in the future), and condition-2 would be introduced by action N-2 which would be required by some action N-4 in the future. The REA would then use the causal structure information to monitor condition-1 and condition-2 until actions N-3 and N-4 were executed.

Protection Violation

When updates are made to the REA's world model, the activated protection monitors are examined to determine whether a violation has occurred. When the world model is updated with new information a process within the REA is notified and informed which information changes. This process then initiates the determination of whether violations have occurred.

Each activated protection monitor that is monitoring the type of condition which changed in the world model is examined to see if the value it expects the condition to have is the same as its new value. This examination only takes place if all of the contributors of the condition have been executed (i.e., the condition should exist). If one or more contributors remain to be executed then it is likely that a premature violation has occurred, so this potential violation is ignored and the monitor remains activated. If it is the same then the monitor remains activated otherwise, if it is not the same then a violation has occurred and must

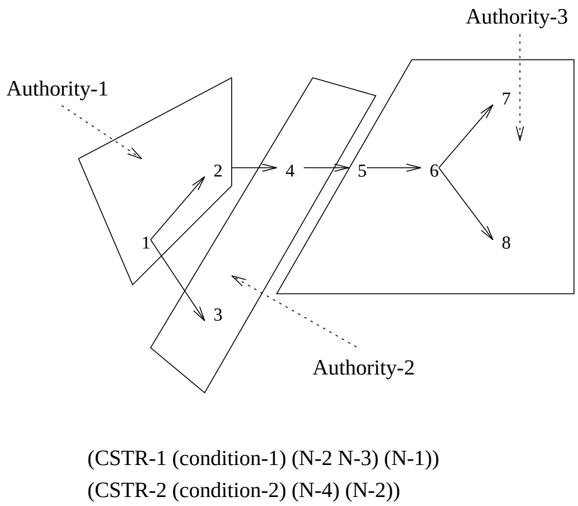

Figure 5: Planner Authority Levels

be examined further.

The planner uses the causal structure to prevent plan state interactions, but the execution agent does not have the ability to prevent things from happening. Not everything in the environment is under the agent's control and some other agent might have interfered.

When a violation has occurred several considerations must be made. First, did the contributor (or last known contributor in the case of multiple alternative contributors) fail? If so, then the violation can be ignored since it was the failure to produce the condition and not any interaction in the environment. This type of failure is handled by another component of the REA. If the violation was not due to a failure then there must have been something acting in the environment which caused the violation. In this situation one of three things could be done—reintroduction, repair, or failure.

Reintroduction is the process of executing another action which will yield the same condition that was violated. However, there are several issues which must be addressed here. This can only occur if the execution agent's representation is rich enough to allow for it to perform the reasoning required to find such a candidate. Then there is the issue of what interactions the introduction of this new action could cause. It could have effects which would interact with other actions waiting to execute and cause additional violations to occur. Reintroduction allows the system to detect and correct a causal structure violation before it manifests itself as a failure in the executing plan.

The second way to address the violation is through repair initiated by the planner. In this case, the REA would communicate with the planner to inform it that the preconditions of a particular node (i.e., the range-end node) were not going to be satisfied when it is eligible to execute. This would make it the planner's responsibility to generate a repair and communicate that back to the REA so the violation was removed. The REA would then ignore any future violation detections by that particular monitor. Repair also allows the system to detect and correct a causal violation, but does so with the assistance of the planning system. It provides an early warning system so the planning system can help the REA to avert possible future execution failures.

The third measure that could be taken to address the violation would be to report total failure of the Task-Directive. Though drastic, it could save resources which could be used by other Task-Directives the REA intends to execute. Total failure is only an option when the time remaining before the need to execute the action requiring the condition is so small as for it not to be reasonable to expect the planner to generate a repair in time.

Discussion

To actively monitor for protection violations during execution a planning system must provide the causal structure of the plan to the execution agent. We have shown how valuable execution monitoring information can be synthesized from this causal structure. This information allows an execution agent to detect (and possibly correct) potential failures before they manifest themselves as actual failures in a executing plan. It also provides a means for developing an early warning system so a planning system can assist the execution agent to avert execution failures by suggesting repairs.

But, just how much casual structure is enough? The causal structure sent to the REA is dependent upon the domain description given to the planner when the plan is generated. Therefore, the more causal structure information available, the more likely it is that the REA will be able to monitor the plan's execution using this approach. The limitation is that this approach is only as good as the sensing capabilities of the REA. As defined, the basic approach comes at no cost since these monitors are triggered from updates to the REA's world model and are not actively sensing the environment. The reality however, is that this approach is most effective when sensors are used often to keep the REA's world model up-to-date. To that end, research is continuing to provide the REA with the capability to actively sense the environment (keeping the world model up-to-date) to improve the utility of the approach presented here. The issue then will be is the ability to monitor *during* protection intervals to detect problems early worth the cost.

The utility of the approach presented here cannot be fully realized until many open issues of how a planning systems can use such information have been addressed. It is one thing to know where a plan has failed, and another to be able to repair the plan from that point.

However, some interesting research on the use of causal structure information in plan reuse and modification is being done to address such issues (Kambhampati 1990). Kambhampati uses a *validation structure* to represent the internal dependencies of a plan and then uses that structure to help in modifying plans to suit new situations. Though not exactly what its needed to allow a planner to repair a plan based upon the information from the approach presented in this paper, it is a step in the right direction. Hopefully, this approach will be seen to complement his work and the work of others addressing the issues of plan repair.

Conclusion

The value of using causal structure information in planning systems has been widely recognized. However, its utility in execution systems has not received much attention. The benefits of providing such information to execution systems are realized during deliberation and while reacting to change. The planning system is able to reduce the uncertainty of the information in its model of the world by tasking an execution system to monitor conditions it expects to be valid in the future. The execution system is able to avert potential failures by identifying them sooner thus, giving it more time to make repairs.

Acknowledgments

This research has benefited from discussions with members of the O-Plan Project Team at Edinburgh. The authors would also like to thank the reviewers for their comments and suggestions regarding this paper.

References

Chapman, D. 1987. Planning for Conjunctive Goals. *Artificial Intelligence* 32:333–377.

Currie, K., and Tate, A. 1991. O-Plan: the Open Planning Architecture. *Artificial Intelligence* 51(1).

Doyle, R.; Atkinson, D.; and Doshi, R. 1986. Generating perception requests and expectations to verify the execution of plans. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-86).* Philadelphia, PA: Morgan Kaufmann.

Hart, D.; Anderson, S.; and Cohen, P. 1990. Envelopes as a Vehicle for Improving the Efficiency of Plan Execution. In *Proceedings of DARPA Workshop on Innovative Approaches to Planning Scheduling and Control*, 71–76. San Diego, California: Morgan Kaufmann.

Kambhampati, S. 1990. A Theory of Plan Modification. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-90)*, 176–182. Boston, MA.: Morgan Kaufmann.

McAllester, D., and Rosenblitt, D. 1991. Systematic Nonlinear Planning. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-91).*

Reece, G. A., and Tate, A. 1993. The Pacifica NEO Scenario. Technical Report ARPA-RL/O-Plan2/TR/3, Artificial Intelligence Applications Institute.

Reece, G.; Tate, A.; Brown, D.; and Hoffman, M. 1993. The PRECiS Environment. In *Proceedings of National Conference on Artificial Intelligence (AAAI-93) DARPA-RL Planning Inititive Workshop.* Available as ARPA-RL/O-Plan2/TR/11 from the Artifical Intelligence Applications Institute.

Reece, G. A. 1992. Reactive Execution in a Command, Planning, and Control Environment. Technical Report 121, Department of Artificial Intelligence, University of Edinburgh, Scotland.

Sanborn, J., and Hendler, J. 1988. A Model of Reaction for Planning in Dynamic Environments. *Artificial Intelligence in Engineering* 3(2):95–102.

Sussman, G. 1975. *A Model of Skill Acquisition.* Elsevier Scientific.

Tate, A. 1977. Generating Project Networks. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-77).*

Tate, A. 1984. Planning and Condition Monitoring in a FMS. In *Proceedings of International Conference on Flexible Automation Systems.* Available as AIAI-TR-2 from the Artifical Intelligence Applications Institute.

Tate, A. 1993. Authority Management Coordination between Task Assignment Planning and Execution. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI-93) Workshop on Knowlege-based Production Planning, Scheduling and Control.* Available as AIAI-TR-133 from the Artifical Intelligence Applications Institute.

Wilkins, D. 1984. Domain-Independent Planning: Representation and Plan Generation. *Artificial Intelligence* 22(3):269–301. Available as SRI-TR (May 1983) from SRI International.