Prevention Techniques for a Temporal Planner*

John C. Hogge
Artificial Intelligence Laboratory
Texas Instruments
P.O. Box 655474 MS 238
Dallas, Texas 75265

Abstract

Research in domain independent planning has concentrated on making desired facts (goals) become true, with less attention on preventing undesired facts (negative goals) from becoming true. This document presents preliminary work towards a temporal-based model of prevention, based on Allen and Koomen's temporal planner. The temporal model permits more expressive prevention problems and a more powerful prevention-oriented planner. Negative goals are temporally constrained—for instance, we can direct the planner to prevent a fact F from occurring before, after, or during a specific goal, between two temporally separated goals, etc. The planner can find solutions which allow F to become true at any time outside of the specified interval or which never allow F to become true. The temporal model permits the idea of *delaying* F after the interval, as well as *terminating* it before the interval. This paper describes a set of prevention techniques which have been implemented in a temporal planner and discusses necessary requirements for a more complete preventive planner.

1 Introduction

Research in domain independent planning has concentrated on making desired facts become true, with less attention on preventing undesired facts from becoming true. [McDermott, 78] formulated prevention as a policy influencing the achievement of a goal, while [Fikes, Hart & Nilsson, 72] proposed extending STRIPS to allow *negative goals* solvable through an operator's delete list. This document presents preliminary work towards a temporal-based model of prevention, based on the temporal planner of [Allen and Koomen, 83]. The temporal model permits more expressive prevention problems and a more powerful prevention-oriented planner. Negative goals are temporally constrained—for instance, we can direct the planner to prevent a fact F from occurring before, after, or during a specific goal, between two temporally separated goals, etc. The planner can find solutions which allow F to become true at any time outside of the specified interval or which never allow F to become true. (Non-temporal planners are restricted to the latter set of solutions.) The temporal model permits the idea of *delaying* F after the interval, as well as *terminating* it before the interval.

Section 2 overviews the temporal planner for which we have formulated the prevention techniques. (The planner, prevention techniques, and a set of examples are implemented in Common Lisp and are publically available.) Section 3 specifies how prevention problems are posed in our planner. Section 4 describes the criteria for detecting that something must be done to prevent some undesired fact, while Section 5 presents a set of prevention techniques. Section 6 describes the search strategy which employs these techniques.

*This research was conducted at the Qualitative Reasoning Group, Department of Computer Science, the University of Illinois at Urbana-Champaign, 1304 W. Springfield Avenue, Urbana, Illinois 61801. It was supported by the Office of Naval Research, Contract No. N00014-85-K-0225.

Table 1: Seven Possible Values of Interval Relations and their Inverses

Value	Description	Inverse	Description
:<	before	:>	after
:M	meets	:MI	met by
:O	overlaps	:OI	overlapped by
:S	starts	:SI	started by
:F	finishes	:FI	finished by
:D	during	:DI	encloses
:=	equals	:=	equals

2 Overview of the Temporal Logic and Planner

Allen's temporal logic defines an *interval* as a discrete portion of time, often corresponding to the period of time over which a fact holds. The *temporal relation* between any two intervals is a disjunction of the thirteen possible values given in Table 1. For instance, the logic might know that one interval can occur either before, after, or overlapping another interval, as shown in Figure 1.

Throughout this paper, pattern variables are denoted by symbols starting with ?. Temporal intervals are denoted by symbols starting with $. Temporal relations are written as lists; for example, if $INTERVAL1 can occur before, after, or overlapping $INTERVAL2, we write

```
$INTERVAL1 (:< :> :O) $INTERVAL2.
```

The logic maintains a transitive closure over temporal relations. For instance, $INTERVAL1 (:=) $INTERVAL2 and $INTERVAL1 (:< :=) $INTERVAL3 implies $INTERVAL2 (:< :=) $INTERVAL3.

[Allen and Koomen, 83]'s planner temporally qualifies

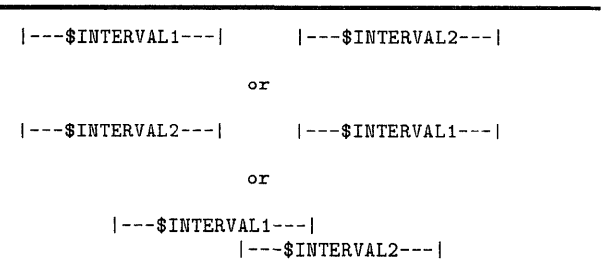

Figure 1: The Temporal Relation $INTERVAL1 (:< :> :O) $INTERVAL2.

the given and goal facts of each partial plan with intervals and a set of temporal relations among them. We have extended the planner to handle domain-dependent, temporally qualified causal rules for modeling complex (interacting) operator effects. For instance, a rule might assert some condition when two applied operators overlap in time. A rule's antecedents are comprised of a set of given facts and temporal conditions which must hold among them. Its consequents are a set of new given facts and temporal constraints among antecedents and consequents.

3 Prevention Definition

For the purposes of this paper, *prevention* is the process of planning for goals while keeping *undesired facts* from holding over specified time intervals. Problem descriptions include *prevention specifications* — pairs of prevention unification patterns, their time intervals, and temporal constraints on these intervals. Pattern variables are universally quantified. For instance, suppose our problem description includes a prevention specification

```
(ON A ?ANYTHING) $A-NOT-ON-ANYTHING,
```

and temporal constraints positioning `$A-NOT-ON-ANYTHING` with respect to intervals of given and goal facts. The planner solves the problem, preventing any fact unifying with `(ON A ?ANYTHING)` from temporally intersecting `$A-NOT-ON-ANYTHING`. When such intersections or *violations* are detected at a search node (using the criteria described in Section 4), our planner backtracks and uses the techniques presented in Section 5 to prevent the violation.

4 Detecting Prevention Violations

A fact of a given search node violates a prevention specification if it passes two tests. First, the fact must unify with the prevention pattern with none of its variables being bound to constants in the pattern (though variables in the pattern can be bound to constants of the fact). For instance, `(AT ?SOME-POT LOCATION)` tests negative against pattern `(AT POT1 LOCATION)`, while `(AT POT1 LOCATION)` tests positive against pattern `(AT ?SOME-POT LOCATION)`. Clearly, when preventing `(AT POT1 LOCATION)`, we can not assume that `(AT ?SOME-POT LOCATION)` should be prevented since we do not know whether `?SOME-POT` refers to `POT1`. Contrarily, to prevent `(AT ?SOME-POT LOCATION)` for all `?SOME-POT`, `(AT POT1 LOCATION)` must be prevented.

The second test for detecting violations depends on whether the search node is a partial or finished plan. For partial plans, the fact's interval must *definitely intersect* the prevention interval, meaning their temporal relation is a subset of `(:S :SI :F :FI :D :DI :O :OI :=)`. For finished plans, the fact's interval must *possibly intersect* the prevention interval, meaning their temporal relation has a non-zero intersection with `(:S :SI :F :FI :D :DI :O :OI :=)`. The distinction is important. If a prevention violation is possible (but not definite) in a partial plan, acting to correct the violation loses generality, since the final solution might involve a combination of operators and constraints which make the violation impossible. In a finished plan, no further constraints will be added, so it is reasonable to prevent a possible violation.

5 Prevention Techniques

In order to retain completeness, a planner searches every possible way of solving a goal through applicable operators. Likewise, when preventing a fact from intersecting an interval, our planner attempts to retain completeness by searching all possible ways it knows of accomplishing the prevention. While our prevention techniques are not complete, they work in many situations. The techniques are described in the following three sections. In a nutshell, we prevent a fact from intersecting an interval by either terminating it before the interval (Section 5.1), delaying it past the interval (Section 5.2), or preventing it from being asserted as a rule consequent (Section 5.3).

5.1 Preventing via Termination

This section describes three techniques for preventing a fact's interval from intersecting another interval by constraining (terminating) its endpoint such that the fact is before `(:<)` or meets `(:M)` the interval.

Termination of Rule Antecedents If the fact to be terminated is the consequent of a rule, we can indirectly terminate it before or meeting `(:< :MI)` the prevention interval by terminating one of the rule's antecedents. The technique can only be applied to antecedents which *terminate* the consequent, meaning `$ANTECEDENT (:> :MI :SI :F :FI :DI :OI :=) $CONSEQUENT` holds. This temporal condition says that `$ANTECEDENT`'s endpoint does not occur sooner than `$CONSEQUENT`'s endpoint. We assume that by ending the antecedent sooner, the consequent is ended sooner. Under this definition, the constraint `$ANTECEDENT (:< :M) $SOME-INTERVAL` results in the constraint `$CONSEQUENT (:< :M) $SOME-INTERVAL`, by transitivity. Consider the following rule:

```
Antecedents:
  (HEAT-PATH ?src ?dst) $heat-path,
  (> (TEMPERATURE ?src)
     (TEMPERATURE ?dst)) $temp-difference
Antecedent Temporal Conditions:
  There exists $heat-flow such that
    $heat-path (:SI :FI :DI :=) $heat-flow and
    $temp-difference (:SI :FI :DI :=) $heat-flow
Consequents:
  (HEAT-FLOW ?src ?dst) $heat-flow
```

Since $HEAT-PATH (:SI :FI :DI :=) $HEAT-FLOW meets the termination criteria, we can terminate $HEAT-FLOW before some interval by terminating $HEAT-PATH before that interval.

Determining which antecedents of a rule can terminate which consequents is a difficult problem which we've only partially solved. Since a rule may have implicit consequent temporal constraints inferable through transitivity, we take the transitive closure of rule constraints before checking for the termination condition (:> :MI :SI :F :FI :DI :OI :=). This is still insufficient, since sets of interacting rules in a domain can have implicit temporal constraints not apparent when examining each rule's local temporal constraints.

We also make several simplified assumptions. First, we assume that one can consistently terminate a consequent by terminating only one of the antecedents. As a counter example, consider the following rule:

```
Antecedents:  A $A, B $B
Antecedent Temporal Conditions:  $A (:=) $B
Consequents:  C $A
```

The rule is triggered when two facts A and B are temporally equal. Under these circumstances, C is asserted over A's interval. Since the consequent is temporally equal to both antecedents, our definition says it can be terminated by terminating either antecedent. However, it is not generally true that terminating $A necessarily causes a corresponding termination of $B and vice versa. We would expect to have to find a means of terminating both intervals in a coordinated manner such that their temporal relation value (:=) is maintained.

Another assumption is that intervals do not have fixed durations. If an antecedent were to meet a consequent and the consequent were known to hold over a fixed duration, terminating the antecedent would cause the consequent to terminate sooner.

When terminating a consequent by terminating one of its antecedents, we can apply any of the three methods described in this section. For instance, we can terminate a consequent by terminating one of its antecedent's antecedents.

Unfortunately, our approach misses some problem solutions through overconstraint. The ideal approach would terminate the antecedent as late as possible such that the consequent meets or occurs before the prevention interval, without necessarily asserting (:< :M) from antecedent to the prevention interval. Such an implementation probably requires a more expressive temporal logic with a notion of metric durations, such as [Dean & McDermott, 87]. Our temporal logic does not give us a way of expressing what changes to the antecedent's endpoint cause the consequent to miss the prevention interval. For instance, shortening an antecedent by two minutes might cause a consequent to just miss the prevention interval, without having the antecedent entirely miss the prevention interval.

Termination by Constraining an Applied Operator If the fact to be terminated was a precondition or effect of an applied operator and the operator has temporal control over its endpoint, we can simply assert the constraint that the fact is before or meets the prevention interval. For instance, a simple blocksworld MOVE operator would be able to control the endpoint of its precondition (ON ?OBJECT ?SURFACE), which meets effect (HOLDING ?OBJECT), which meets effect (PUTDOWN ?OBJECT ?NEW-SURFACE), and it could likewise control the endpoint of the effects. Other operators might not be able to control such endpoints—for instance, a doctor can "bring a patient back to life," but that act does not control how long the patient lives. We provide such control as a user-definable parameter in operator definitions.

The ability to constrain applied operators allows us to solve the following prevention problem. Suppose our domain has the previous MOVE operator. Our problem specifies given (ON A C) $ON-AC, goal (ON A B) $ON-AB, prevention specification (HOLDING ?X) $NOT-HOLDING-ANYTHING, and the following constraints:

```
$ON-AC (:<) $NOT-HOLDING-ANYTHING,
$NOT-HOLDING-ANYTHING (:<) $ON-AB
```

Sometime between (ON A C) and (ON A B), there is an interval over which we do not want to be holding anything (but we could be picking up things or putting down things). The goal (ON A B) is solved by applying MOVE, introducing effect (HOLDING A) which can possibly intersect $NOT-HOLDING-ANYTHING. To prevent the intersection, we constrain the endpoint of (HOLDING ?X) to occur before or meeting $NOT-HOLDING-ANYTHING. Figure 2 shows the resultant plan and one set of the possible temporal constraints.

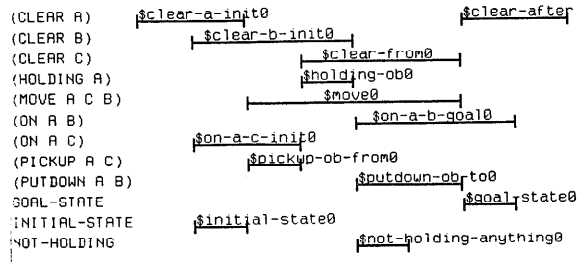

Figure 2: Plan to move A, preventing HOLDING during $NOT-HOLDING-ANYTHING

Termination by Applying an Operator The final technique for terminating a fact before or meeting some interval is to apply an operator. The operator must have a precondition unifying with the fact and must be defined to have control over the precondition's endpoint, permitting us to assert the constraint. For instance, the blocksworld MOVE operator discussed above could be applied to terminate any fact unifying with precondition (ON ?OBJECT ?SURFACE) by moving ?OBJECT to some other surface. (The

MOVE operator's definition would specify that it has control over the interval endpoint of the precondition.) This technique is similar to what [Fikes, Hart & Nilsson, 72] formulated for STRIPS, with endpoint controllable preconditions instead of delete lists.

5.2 Preventing via Delay

This section presents three techniques for preventing a fact from intersecting an interval by constraining (delaying) its startpoint such that the fact is after or met-by (:> :MI) the interval. Since these techniques correspond to the three termination techniques presented in Section 5.1, they are presented by comparison.

Delay of Rule Antecedents If the fact to be delayed is the consequent of a rule, we can indirectly delay it after or met-by (:> :MI) the prevention interval by delaying one of the rule's antecedents. The technique can only be applied to antecedents which can *delay* the consequent, meaning `$ANTECEDENT (:< :M :S :SI :FI :DI :O :=) $CONSEQUENT` holds. Determining whether an antecedent can delay a consequent is performed in a similar manner to determining whether an antecedent can terminate a consequent. Searching possible ways of delaying an antecedent is done similarly by recursively applying the three techniques presented in this section. For instance, we can delay a consequent by delaying one of its antecedent's antecedents.

Delay by Constraining an Applied Operator If the fact to be delayed was a precondition or effect of an applied operator and the operator has control over its startpoint, we can simply assert the constraint that the fact is after or met-by the prevention interval. The operator definition specifies whether an operator controls the startpoint of precondition and effect intervals. As an example of delay by operator constraint, Figure 3 shows another solution to the plan described in Figure 2. This solution performs the prevention by constraining the startpoint of operator effect `(HOLDING ?X)` to occur after or met-by `$NOT-HOLDING-ANYTHING`.

Delay by Applying an Operator The final technique for delaying a fact after or met-by some interval is to apply an operator. The operator must have a precondition unifying with the fact and must be defined to have control over the precondition's startpoint. Unlike the termination operators described in Section 5.1, delay operators are less useful for modeling domains. While termination operators terminate a condition which already holds, delay operators delay the start of a condition which will inevitably hold. Such inevitability is rare in a domain— if an action can delay the start of a condition, it may keep it from ever occurring. Instead, such conditions should be modeled as rule consequents, which are subject to the prevention techniques of Section 5.3 as well as delay techniques.

5.3 Preventing Rule Consequents

One technique for preventing a fact from intersecting an interval is to prevent the fact from ever being asserted. This technique can be restricted to facts which are the consequents of rules, excluding the effects of applied operators, since our planner is complete in searching possible

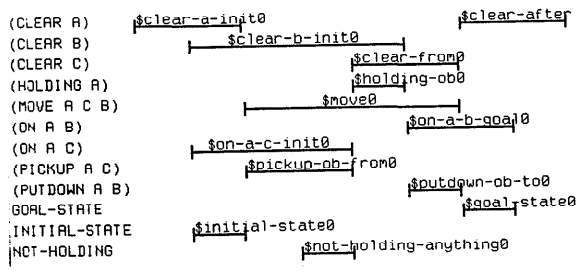

Figure 3: Another plan to move A, preventing HOLDING during $NOT-HOLDING-ANYTHING

combinations of operators which solve the goals. In other words, if a search node's operator asserts an effect which violates a prevention interval, we do not have to backtrack for a solution which avoids applying the operator, since such solutions are already being searched.

Preventing Assertion of Rule Antecedents One way of preventing a rule consequent is to prevent one of the rule's antecedents from being asserted. For each antecedent which was itself the consequent of a previous rule, we recursively apply the techniques for preventing rule consequents to the previous rule. Preventing a rule consequent requires (for completeness) searching all possible ways of interrupting the chain of inference which led to the assertion of any one of its antecedents. The rule at the top of the chain can only be prevented by upsetting its temporal conditions, since its antecedent facts are either initially given or the effects of operators.

Preventing Temporal Conditions A rule can also be prevented by achieving the opposite of one of its temporal conditions. For instance, for the temporal condition

`$INTERVAL1 (:S :F :D :=) $INTERVAL2`

we search ways of constraining `$INTERVAL1` and `$INTERVAL2` to values `:SI, :FI, :DI, :<, :>, :M, :MI, :O,` or `:OI`. A given value can be achieved by manipulating one of the two antecedent's intervals. Values `:<` and `:>` can be achieved through the termination and delay techniques presented in sections 5.1 and 5.2, respectively. For instance, `$a :< $b` can be achieved either by terminating `$a` such that `$a :< $b` or by delaying `$b` such that `$a :< $b`. The other eleven values require more restrictive techniques. Whereas the termination techniques move an interval's endpoint backwards (sooner) in time, and the delay techniques move an interval's startpoint forwards (later) in time, these values require bidirectional control over endpoints and startpoints, allowing us to move them forwards or backwards.

For instance, when achieving `$a :M $b`, if we could only move `$a`'s endpoint backwards, we should not be able to act in situations where `$a :(<) $b`.

Our bidirectional control techniques are similar to the termination and delay techniques, except that the technique for constraining a rule consequent by constraining an antecedent is more restrictive. Section 5.1 explained that if an antecedent terminates a consequent through temporal constraints defined in the rule, the consequent can be prevented over an interval by terminating the antecedent over that interval. The definition of terminatability was

<antecedent> (:> :MI :SI :F :FI
:DI :OI :=) <consequent>.

The definition we use for bidirectional endpoint control [1] is:

IF <antecedent> (:F :FI :=) <consequent> THEN
<consequent>'s endpoint can be controlled by controlling <antecedent>'s endpoint.
IF <antecedent> (:MI) <consequent> THEN
<consequent>'s endpoint can be controlled by controlling <antecedent>'s startpoint.

In other words, <consequent>'s endpoint must be known to occur at either <antecedent>'s startpoint or endpoint. Despite the restrictive temporal conditions, this control technique is often effective since many domain rules have constraints of the form <antecedent> (:=) <consequent>, which satisfies the restriction.

This control technique has limitations. For instance, if the rule specified that <antecedent> (:DI) <consequent>, the consequent is sandwiched between the startpoint and endpoint of the antecedent. Therefore, achieving control of both startpoint and endpoint of the antecedent would achieve control over the startpoint and endpoint of the consequent. While we could improve the search strategy to achieve such simultaneous control, the temporal logic does not give us a way of expressing what changes to the antecedent's startpoint and endpoint cause the desired constraint on the consequent's startpoint or endpoint.

For a full description of how the bidirectional control techniques are used, refer to [Hogge, 87b]. For brevity, we will just describe one case. In order to achieve `$A1 (:S) $A2`, one has to constrain the two intervals' startpoints to happen simultaneously (expressed as the constraint `(:S :SI :=)`) and to constrain their endpoints such that `$A1`'s endpoint happens before `$A2`'s endpoint (expressed as the constraint `(:O :S :D)`). We must search these two requirements independently, queueing search nodes which constrain the startpoints and others which constraint the endpoints. (Notice that meeting both requirements results in the desired value `:S`, since `:S` is the intersection of `(:S :SI :=)` and `(:O :S :D)`.) It would be less general to assume that actions must be performed to constrain both the startpoints and endpoints; for instance, the act of constraining the endpoints of `$A1` and `$A2` might fire some rule which causes their startpoints to be constrained as desired.

Our techniques for preventing temporal conditions assume that a temporal constraint between two intervals can be achieved by constraining one of the two intervals. It would require a richer temporal logic (with a notion of duration) to be able to express the need to constrain both intervals: for instance, to achieve `$A1 (:<) $A2` one might have to terminate `$A1` *and* delay `$A2`.

[1] For bidirectional startpoint control, substitute `:S` for `:F`, `:SI` for `:FI`, and `:M` for `:MI`.

6 Prevention Search Strategy

We have implemented a search strategy for correcting prevention violations, using the techniques presented in previous sections. (See [Hogge, 87b] for details of the algorithm.) The strategy allows the techniques to use each other in a recursive fashion. For example, a fact might be prevented over an interval by preventing it from being asserted as a rule consequent, accomplished by preventing an antecedent of the rule from being asserted by another rule, accomplished by thwarting a temporal condition of the rule, accomplished by applying an operator to terminate an antecedent such that the condition does not hold. The strategy's complexity is bounded by the number of operators, the lengths of inference chains, and the number of possible relation values in the inverse of each rule's temporal conditions.

Our search strategy suffers from one source of incompleteness which appears difficult to correct. When applying an operator to accomplish some temporal constraint (such as terminating an interval or controlling its endpoint), our implementation only backtracks to the parent of the current search node. Thus, we sometimes miss the solution since the parent node might be too constrained (through the presence of other operators) whereas one of its ancestor nodes might not be. As a trivial example, suppose a rule asserts a contradiction if more than two MOVE operators are used in a plan. If two moves have been applied at node W and we must introduce a third to carry out some prevention, introducing it into W will cause a contradiction. Thus, completeness requires trying *every* ancestor of W. Besides increasing the complexity, this is problematic since the intervals to be constrained may not exist early enough in the search tree. Since the existence of intervals depends on the order in which operators were applied during search, completeness would require reconstructing the search with different operator orderings.

7 Summary

The following summarizes our techniques for preventing a fact F from intersecting an interval in a temporal planner:

1. Constrain (terminate) F's interval before or meeting `(:< :M)` the prevention interval.
   - 1a. If F is the consequent of a rule, accomplish the constraint by constraining an antecedent.
   - 1b. If F was a precondition or effect of an applied operator, and the operator definition specifies *endpoint control*, simply assert the constraint.
   - 1c. Apply an operator which has an *endpoint controlled* precondition unifying with F and assert the constraint.
2. Constrain (delay) F's interval after or met-by `(:> :MI)` the prevention interval.
   - 2a. If F is the consequent of a rule, accomplish the constraint by constraining an antecedent.

2b. If F was a precondition or effect of an applied operator, and the operator definition specifies *startpoint control*, simply assert the constraint.

2c. Apply an operator which has a *startpoint controlled* precondition unifying with F and assert the constraint.

3. If F is the consequent of a rule, prevent F from being asserted by preventing the rule from firing. Backtrack to the search node before the rule fired and:

3a. upset its temporal preconditions, or

3b. prevent assertion of an antecedent by preventing the rule which asserted it as a consequent.

Further work in prevention should make use of more expressive temporal representations to solve the shortcomings of our techniques and should address the backtracking problem of our search strategy. A useful extension would be to allow prevention specifications (temporally constrained negative preconditions) in operator definitions.

8 Acknowledgements

Brian Falkenhainer posed a problem which got me interested in prevention. Thanks to Ken Forbus and QRG for your support. The Office of Naval Research supported this project through Contract No. N00014-85-K-0225.

References

[Allen, 83] Allen, J.F., "Maintaining Knowledge about Temporal Intervals", *Communications of the ACM*, vol. 26, pp. 832-843.

[Allen and Koomen, 83] Allen, J.F. and Koomen, J.A., "Planning Using a Temporal World Model", *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pp. 741-747.

[Dean & McDermott, 87] Dean, T.L. and McDermott, D.V., "Temporal Data Base Management", *Artificial Intelligence*, vol. 32, pp. 1-55.

[Fikes, Hart & Nilsson, 72] Fikes, R., Hart, P., and Nilsson, N., "Some New Directions in Robot Problem Solving," *Machine Intelligence* 7 (1972), pp. 405-430.

[Hogge, 87a] Hogge, J.C., "TIME and TPLAN User's Manual", University of Illinois Department of Computer Science Technical Report UIUCDCS-R-87-1366, September 1987.

[Hogge, 87b] Hogge, J.C., "TPLAN: A Temporal Interval-Based Planner with Novel Extensions", University of Illinois Department of Computer Science Technical Report UIUCDCS-R-87-1367, September 1987.

[Hogge, 87c] Hogge, J.C., "The Compilation of Planning Operators from Qualitative Process Theory Models.", University of Illinois Department of Computer Science Technical Report UIUCDCS-R-87-1368, September 1987.

[McDermott, 78] McDermott, D., "Planning and Acting", *Cognitive Science*, vol. 2, pp. 71-109.