Planning Using a Temporal World Model

James F. Allen and Johannes A. Koomen
Computer Science Deparunent, University of Rochester
Rochester, NY 14627 U.S.A.

Abstract

Current problem solving systems are constrained in their applicability by inadequate world models. We suggest a world model based on a temporal logic. This approach allows the problem solver to gather constraints on the ordering of actions without having to commit to an ordering when a conflict is detected. As such, it generalizes the work on nonlinear planning by Sacerdoti and Tate. In addition, it allows more general descriptions of actions that may occur simultaneously or overlap, and appears promising in supporting reasoning about external events and actions caused by other agents.

1. Introduction

Current problem solving systems are constrained in their applicability by inadequate world models. In particular, in most systems, the model of time is such that actions must be considered to be instantaneous, and only one action can occur at a time. This is the case in state-space based systems such as those of Fikes and Nilsson [1971], as well as in other systems based on the situation calculus [McCarthy, 1968]. In addition, these systems can only consider domains in which changes are made only as a result of the planner's actions, and the goals that can be described are confined to a single time instant. Thus, one couldn't express a goal such as "Put block A on B, and then later move A to C."

Recent work has extended these models in a few directions. Sacerdoti [1977], for example, allows for partial ordering of actions in his plans, but retains a simple world model. As a result, actions are still viewed as instantaneous, for two actions either occur simultaneously or one must be strictly before the other. No possibility is allowed that actions might overlap in any way. McDermott [1978] allows constraints on the solution of a problem of the form: "Don't violate goal X during the solution." Vere [1981] allows events not caused by the planner provided that there is a reasonable estimate of the date at which the event will occur.

The preparation of this paper was supported in part by the Defense Advanced Research Projects Agency under Grant N00014-82K-0193, and in part by the National Science Foundation under Grants 1ST 8210564 and MCS-8209971. Many thanks to Rich Pelavin and Henry Kautz for comments on the initial drafts.

We propose a formalism that incorporates these extensions and relaxes most of the other restrictions discussed above. The world model consists of all the planner's knowledge of the past, present, and future, expressed in a temporal logic. In simulating the effects of an action, the state of the world is not updated temporally as in previous systems. Instead, the planner's knowledge, primarily the predictions about the future, are updated. To draw a loose analogy to state-space based planners, the states in this model are states of the planner's knowledge and are independent of the temporal aspects of the world.

Given this approach, a *plan* is a collection of assertions viewed as an abstract partial simulation of the future, including actions the planner intends to take as well as other predicted actions, events, and states. In a coherent plan, most but not necessarily all of these events and states are causally related. A *goal* is a partial description of the world desired. This description is not confined to a specific instant of time. It might consist of a sequence of states (e.g., get block A on B, then later get A on C), restrictions (e.g., never let ON(B,C) be true), or any other set of facts expressible in the temporal logic.

We will not suggest any new methods for problem solving here. Our current concern is simply to investigate the consequences of the more general world model. In fact, we consider it a major asset of this representation that it can be used with existing problem solving methods (e.g., means-end analysis, decomposition, etc.). We will discuss some issues in this area later in the paper.

We will use a STRIPs like action formalism (as in [Nilsson, 1980]), except that the preconditions and effects will be temporally qualified. The temporal representation used is that described in [Allen, 1981]. The basic unit is that of a temporal interval, and intervals can be related by any of seven primitive relations and their inverses. These are summarized in Figure 1.

Usually, the precise relationship between two intervals is not known. In such cases, we can express a disjunclion of the primitive relations. For example, the fact that intervals A and B are disjoint can be expressed by asserting that A is *before, meets,* is *met by,* or is *after B*. This is generally summarized with the notation:

A (< m mi >) B.

Relation	Symbol	Symbol for Inverse	Pictoral Example
X *before* Y	<	>	XXX YYY
X *equal* Y	=	=	XXX YYY
X *meets* Y	m	mi	XXXYYY
X *overlaps* Y	o	oi	XXX YYY
X *during* Y	d	di	XXX YYYYYY
X starts Y	s	si	XXX YYYYY
X finishes Y	f	fi	XXX YYYYY

Figure 1: The Thirteen Possible Relationships

Another complex relationship that will be useful later is the notion that one interval A wholly contains another interval B. We shall assert A *contains* B as an abbreviation for

A (di si fi) B.

A computationally effective inference procedure has been developed based on constraint propagation. There is not space to discuss it here, but see [Allen, 1981). The inferences made by this system are simply those derivable from the transitivity behavior of the relations. A typical example of such an inference rule is:

If A *during* B and B *meets* C
then A is *before* C.

All of the examples following have been implemented and run in a prototype planning system.

II. A Simple Example

This problem consists simply of stacking three blocks into a tower. There is one type of action required: that of stacking two blocks. Nilsson [1980] formalizes STACK as follows:

```
STACK(x,y)
    Preconditions: CLEAR(x), CLEAR(y)
    Effects: ON(x,y)
    DeleteList: CLEAR(y)
```

The same action in our formalism is expressed as an axiom:

If STACK(*x*,*y*) occurs over time Sxy, then
  CLEAR(*y*) holds over time Cy,
    such that Sxy *finishes* Cy; and
  ON(*x*,*y*) holds over time Oxy,
    such that Sxy *meets* Oxy; and
  CLEAR(*x*) holds over time Cx,
    such that Sxy *during* Cx.

Note that the deletion information is implicit in the temporal annotation. The precondition CLEAR(y), for instance, is constrained to terminate at the end of the slacking action. This is not the only form of precondition expressible in our system. We might have preconditions that overlap the action, continue after the action, or even hold during a part of the action.

When the temporal intervals associated wiih an instance of an action are added to the temporal reasoner, other relationships are automatically derived. Tor instance, from the fact that *Sxy finishes Cy* and Sxy *meets* Oxy, the relation *Cy meets* Oxy will be inferred. In the examples below, we shall only mention such inferential behavior if it is crucial to the example.

'The problem is to take three clear blocks A. B, and C, on a table and construct a tower with A on B and B on C. The initial description of the problem consists simply of the initial state and the goal state. Introducing interval I for the time of the initial stale, and (J for the goal stale, we can describe the situation as follows:

I *before* G.
CLEAR(A) holds over time Ca1, such that
  Ca1 *contains* I.
CLEAR(B) holds over time Cb1, such that
  Cb1 *contains* I.
CLEAR(C) holds over time Cc1, such that
  Cc1 *contains* I.
ON(A,B) holds over time Oab, such that
  Oab *contains* G.
ON(B,C) holds over time Obc, such that
  Obc *contains* G.

Planning is initiated on those assertions that have no causal explanation. Tor the purposes of this paper, this means that the assertion is not the effect of an action nor true at the initial time I. Thus we have two subgoals to achieve: ON(A,B) and ON(B.C). We use Sacerdotfs strategy for conjunctive subgoals [1977] and attempt to achieve each goal independently. The stack action is applicable for achieving goals of form ON(x,y). Thus we will introduce two stacking actions.

'The action STACK(A,B) is added with its effect ON(A,B) set to hold over Oab. This results in the following new facts being added (facts irrelevant for the example are omitted):

STACK(A,B) occurs over time Sab, such that
  Sab (> mi) I
CLEAR(B) holds over Cb2, such that
  Sab *finishes* Cb2
ON(A,B) holds over Oab, such that
  Sab *meets* Oab
CLEAR(A) holds over Ca2, where
  Sab *during* Ca2.

Introducing the action STACK(B,C) over time She for the goal ON(B,C) yields a similar set of constraints, namely

Sbc (> mi) I
Sbc *finishes* Cc2
Sbc *meets* Oab
Sbc *during* Cb3.

The problem solver, besides adding these action descriptions, also adds further constraints based on the structure of the domain. There are two constraints relevant here. The first is a general constraint imposed by the temporal representation, and the second is specific to the blocks world.

When we assert that a proposition *P holds over* a lime interval T, T is assumed to be the largest possible interval over which the proposition holds. This means that two intervals associated with the same proposition cannot meet or overlap, otherwise they would be identical. This results in our first general constraint:

> *The Proposition Constraint:* Two intervals associated with the same proposition are equal, or one is strictly before the other; in other words, if P holds over Tpl and Tp2, then Tpl (< > =) Tp2.

In our present example, there are three intervals associated with the proposition CLEAR(B). Using the above principle, we may add the facts:

Cb1 (< > =) Cb2
Cb2 (< > =) Cb3
Cb1 (< > =) Cb3

The temporal reasoner combines these constraints with those already derived from the action definitions and may derive stronger constraints. In particular, it derives:

(*) Cb1 (< =) Cb2

using the facts that CbI *contains* 1 and Sab (> mi) I to gel Sab (d f > mi oi) CbI, and then combining this with Sab *finishes* Cb2 to derive that CbI cannot be *after* or *met-by* Cb2. The details of how this is accomplished are not important here. Suffice it to say that (*) is simply a logical consequence of the facts added so far.

The other constraint arises from the observation that in the blocks domain a block cannot simultaneously be CLEAR and have another block on it. Thus we have:

> *Domain Constraint 1*: If CLEAR(*x*) holds over Cx, and ON(*y*,*x*) holds over Oyx, then Cx and Oyx cannot overlap in any manner. In particular,
>
> Cx (m mi > <) Oyx.

Using this rule, the problem solver adds a set of constraints between the times for all CLEAR and ON propositions. The one relevant for this example arises from the interaction of CLEAR(B) and ON(A,B):

Cb3 (m mi > <) Oab

which given the existing constraints becomes

(**) Cb3 (< m) Oab.

This was derived since ON(A,B) (over Oab) must hold in the goal state, whereas CLEAR(B) (over Cb3) must hold before the goal state.

Once (**) is derived, an ordering is imposed between the times *of* the two stacking actions. In particular, given (••) and the fact that Cb2 *meets* Oab, it is inferred that

Cb3 (< =) Cb2.

Since we also have from the action definitions that

Sbc *during* Cb3
Sab *finishes* Cb2

we have that Sbc must complete before Sab completes, i.e., Sbc (< o m s d) Sab. If we wish to add a constraint that only one stacking action can be done at a lime, i.e., Sbc (< > m mi) Sab, we then have STACK(B,C) occuring before STACK(A,B), i.e., Sbc (< in) Sab.

In the NOAH system, such orderings between actions was accomplished using a special-purpose program called the *resolve conflicts* critic. This procedure, however, did not guarantee to make the correct ordering decision if the actions were simply required to be disjoint. Tate [1977] extended this approach by introducing backtracking to cover such cases.. In both systems, a conflict had to be resolved by picking a specific ordering. Our system allows one to defer that choice as long as possible as it can simply note that the actions must be disjoint. Thus the same advantages for delaying the binding of variables by posting constraints in Stefik [1981] are achieved in the area of action ordering. In addition, it is satisfying to see this behavior arising simply as a logical consequence of the formalism without resorting to special purpose techniques.

III. Planning Overlapping Actions

We can now outline the problem-solving system in more detail and then present an example that forces us to reason about overlapping actions. The planner uses hierarchical action descriptions mirroring the approach of Sacerdoti [1977]. In the example below, we shall see that when the two overlapping actions are decomposed, their subactions must interleave to produce a solution.

As mentioned in the introduction, we view planning as reasoning about a simulation of the world extending into the future. Goals are simply facts in the simulation that are required to hold but have no causal explanation. The problem solver continually repeats the process of finding causal gaps and eliminating them with new proposed actions. Some causal gaps, however, can be eliminated by making additional assumptions about the simulated world. For instance, assume that proposition P holds over times Tpl and Tp2 such that Tpl (< =) Tp2. Now, if P holding over Tpl has a causal explanation but

P holding over Tp2 does not, we appear to have a causal gap. Hut we can eliminate this gap if we assume that in fact Tpl - Tp2. If we can eliminate all causal gaps by collapsing such intervals, we have a completed plan at the current level of abstraction.

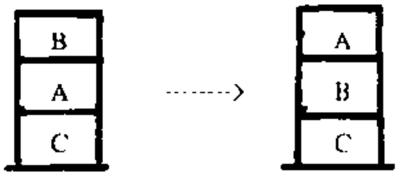
Given the abo\e discussion, we can summarize our problem solver as follows:

Repeat the following two steps until done:

1) Examine the simulation description for causal gaps. If there are none, or they all can be eliminated by collapsing intervals, we are done. Otherwise, the facts without causal explanation become the new set of subgoals.
2) Solve each subgoal independently by introducing a new action. Add any constraints derived from the proposition constraint and the domain constraints.

This algorithm will produce a complete plan at one level of abstraction. The plan can then be refined by adding the decomposition of each action to the simulauon and repeating the process described above.

The following example uses a blocks world in which the table is so small that it can only support one block. The robot has multiple arms and can hold multiple blocks at a time. The problem is to transform a stack with B on A and A on C into a stack with A on H and It on C.



The main action supplied consists of moving a block *x* o another block *t* from a block *f*. This is defined as ollows:

If MOVE(x,t,f) occurs over Mxtf, then
"Preconditions"
 CLEAR(x) holds over Cx1, such that Cx1 *meets* Mxtf,
 ON(x,f) holds over Oxf, such that Oxf *overlaps* Mxtf,
 CLEAR(t) holds over Ct, such that Ct (o s d) Mxtf,
"Effects"
 CLEAR(f) holds over Cf, such that Mxtf (o fi di) Cf,
 CLEAR(x) holds over Cx2, such that Mxtf *meets* Cx2,
 ON(x,t) holds over Oxt, such that Mxtf *overlaps* Oxt,
"Decomposition"
 PICKUP(x,f) occurs over PUxf, such that PUxf *starts* Mxtf,
 HOLDING(x) holds over Hx, such that PUxf *meets* Hx,
 PUTDOWN(x,t) occurs over PDxt, such that Hx *meets* PDxt and PDxt *finishes* Mxtf.

There are a few important things to note about this definition. CLEAR(x) must hold prior to the move action and after the move action, but does not hold while the action is occurring. The CLEAR(t) precondition specifies that t must be clear sometime during the move action, but need not necessarily hold at the start of the MOVE. In other words, this allows the possibility that block I will become clear its a result of some other event while the MOVE is in progress. This condition could be ignored at this level of abstraction, but including it eliminates the possibility of planning a move over a time where the block could not possibly be cleared. The generality of our representation allows us to specify such constraints without having to add the full detail of the decomposition, or alternatively, waiting until the decomposition is done to find the problem, finally, the effect ON(x,t) begins to hold prior to the completion of the move. At that time, the blocks are stacked but the robot is still grasping the top block.

The actions of picking up a block and putting down a block are defined as follows:

If PICKUP(x,y) occurs over PUxy, then
 CLEAR(x) holds over Cx, such that Cx *meets* PUxy,
 ON(x,y) holds over Oxy, such that PUxy *finishes* Oxy,
 CLEAR(y) holds over Cy, such' that PUxy *meets* Cy,
 HOLDING(x) holds over Hx, such that PUxy *meets* Hx.

If PUTDOWN(x,y) occurs over PDxy, then
 HOLDING(x) holds over Hx, such that Hx *meets* PDxy,
 CLEAR(y) holds over Cy, such that Cy *meets* PDxy,
 ON(x,y) holds over Oxy, such that PDxy *starts* Oxy.

In addition to domain constraint I mentioned in the previous example, the following domain constraints are also valid in this blocks world:

A block cannot be held and be clear at the same time:

Domain Constraint 2: For any x, HOLDING(x) is disjoint from CLEAR(x), i.e., Hx (< > m mi) Cx;

A block cannot be held and have another block on it at the same time:

Domain Constraint 3: For any x,y, HOLDING(x) is disjoint from ON(y,x);

A block can only be on one block at a time:

Domain Constraint 4: I or any x,y,z, ON(x,y) is disjoint from ON(x,z), assuming that y is not equal to z.;

A block can only have one block on it at a time:

*Domain Constraint 5: F*or any x,y,/, ON(x,y) is disjoint from ON(z,y), assuming that x is not equal to /.

We can now trace the planning procedure. The initial stale is described by the facts ON(B,A), ON(A,C) and CFFAR(B) holding over the intervals Obc, Oac, and Cbl, respectively. Fach of these intervals contains the initial time I. The final state is described by the fact ON(A,B) and ON(B,C) holding over Gab and Obc. both of which contain G.

In step 1 of the planning algorithm, two causal gaps are discovered, corresponding to the two facts ON(A,B) and ON(B,C). Solving each independently in step 2, two MOVE actions are introduced with their preconditions and effects, namely MOVE (A,B,z) over Mab/ and MOVE(B,C,y) over Mbcy. The *z* and *y* are uninstantiated parameters in the actions to be considered as existentially quantified variables. Once all the domain constraints are added, the temporal reasoner infers that MOVE (B,C,y) must complete before MOVE(A,B,/) completes, i.e., Mbcy (< d m o s) Mabz. This is derived from the fact that ON(A,B), an effect of M0VF(A,B,4 conflicts with CLEAR(B), an effect of MOVE(B,C.y'). Since ON(A,B) holds over the final time G, CFFAR(B) must hold before that.

Step 1 of the planning algorithm is repealed on this new world description. It is noticed that by collapsing Oba with Oby (thereby binding v to block A), and Oac with Oaz (binding z to block C), a plan is produced that is fully causally connected. When the new constraints that Oba *equals* Oby and Oac *equals* Oa/ are added, the temporal reasoner concludes that MOVE (B,C,A) *overlaps* MOVE (A,B,C), i.e., Mbcy *overlaps* Mabz. Thus we can summarize the plan at this level of detail as shown in Figure 2, where time increases from left to right.

This plan can be further elaborated by expanding the overlapping MOVF actions using their decompositions and the definitions of the subactions PICKUP and PUT-DOWN. Once this is done, and the domain constraints have been asserted, the subactions will be constrained to be in the interleaved order shown in Figure 3.

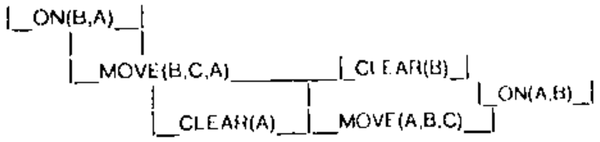


Figure 2.

Most planners' that exploit decomposition as a search technique have depended on the assumption that the actions to be decomposed are disjoint, and therefore their decompositions are independent of each other. Using the temporal world model, this restriction can be relaxed, as the above example demonstrates. We can exploit the efficiency of planning by decomposition without restricting the range of problems that can be solved with the technique.

IV. Current Areas of Research

A. Controlling Temporal Reasoning

The constraint propagation algorithm used in our temporal reasoning system is based on the transitivity of temporal relations. In any realistic application of this reaoning system, however, there will usually be a great many intervals with temporal relations defined between them, and the addition of a single assertion may have extensive effects throughout. Running the second example on our TIMELOGIC system, with 23 intervals defined, 37 task-specific and 52 domain constraints were explicitly asserted. These caused 850 additional constraints to be propagated, only a few of which are "interesting." Although the algorithm only runs while constraints are being further refined and hence activity tends to die down after a while, each newly introduced relation tends to be propagated through the entire system. Once a new interval X is asserted to lie after Y, then it follows immediately that, for every interval Z in the universe such that Y lies after Z, X also lies after Z.

It is clearly desirable to restrict this essentially exponential growth by somehow limiting the scope of constraint propagation. One way we are currently exploring is the use of a hierarchy of reference intervals [Allen, 1981]. We can group various intervals together by asserting their relations to a given reference interval. Then any constraint propagation occurs strictly within the group and its reference interval. Relations between intervals across groups can be deduced from the relation between their respective reference intervals.

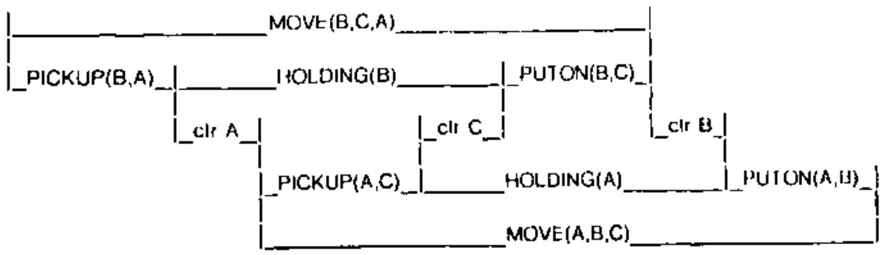


Figure 3.

A number of difficulties have arisen in the use of reference hierarchies, the principal one being the handling of explicitly asserted constraints between intervals with differing reference intervals. Simply asserting the relation does not result in any constraint propagation within either group because the original pair is not from a single group. For instance, in our second example, the interval for the MOVE operation could well serve as a reference interval for the PICKUP, HOLDING, and PUTON intervals. Clearly, any relation between PICKUP and HOI DING is not of interest outside this group. However, the domain constraints often introduce relations across reference groups. We are currently investigating various automatic ways of restructuring the reference hierarchy. Obviously, simply merging two reference groups when their members are found lo interact quickly leads to a total flattening of the hierachy in even as small a problem as our second example.

B. Temporal Durations

One ability that becomes feasible in this new framework concerns reasoning about temporal durations. Since actions take time in this model, we can consider how long they take and use this knowledge in plan construction. This is an important necessary step in designing planners that operate in more realistic domains. The temporal reasoner already can reason about durations. Relative information can be asserted (e.g., interval A takes two to three times as long as interval B), as well as ranges on precise scales (e.g., interval C takes 5 to 10 minutes). This system uses constraint propagation to derive the effects of new duration knowledge. Duration can also affect interval relationships, for example, if the system derives the fact that interval A takes less time than interval B, it adds the constraint that A cannot *contain* B. Similarly, the interval reasoner may constrain the duration reasoner.

Using this work, we hope to build a problem solver in a fairly complex domain, such as cooking or scheduling.

C A General Model of Plan Reasoning

Because we only considered simple problems above, many details of this planning algorithm could be ignored. These need to be addressed to realize the full benefit of the approach. We shall outline two problems that are currently under consideration.

Problems arise when a complete causal explanation cannot be constructed simply by "collapsing" intervals together. In such cases, there may be alternate sets of assumptions, each producing a different set of causal gaps. In these cases, an arbitrary decision must be made by the planner. Once this decision is made, the assumptions motivating it can be removed restoring the world description to its most general case. We currently are examining different strategies for implementing this technique.

The other major problem area involves reasoning about future (or past) events (including actions by other agents). Currently, the existing formalism can express and reason about arbitrary future events, but cannot reason about interfering with them. Thus we can plan to interact with future events, but cannot change them. Reasoning about changing future events (such as preventing an event) requires an ability to change our predictions about the future.

Our approach to this problem is to develop a crude but workable model of hypothetical reasoning, and then use this general mechanism to reason about hypothetical futures. We prefer this approach over that of McDermott [1982], who introduces branching futures into his temporal logic. This is because a hypothetical reasoning ability is required for other purposes besides planning, and once such an ability is present, our simpler temporal model is sufficient. In addition, this framework will allow us to reason about other agents' plans from observing their actions (e.g., [Allen and Perrault, 1980]). Since time only branches into the future in McDermolt's logic, it cannot support such reasoning about the past

V. Summary

We have specified a world model for problem solving using an interval-based temporal logic. The formalism is notable for the following reasons:

-- It allows more general action descriptions than have previously been allowed. In particular, actions may lake time, and their descriptions are not limited to simple precondition and effect descriptions and decomposition.

- It allows more general goal descriptions, and more complicated worlds in which to achieve the goals. In particular, goals are not restricted to a single time, and future events may occur which will affect the planner's behavior.

The action ordering is more general than the nonlinear planners of Sacerdoti and Tate. In particular, actions may overlap, and when conflicts are detected, this method does not have to resort to arbitrary orderings as in Sacerdoti, or backtracking as in fate, as required in certain situations.

REFERENCES

Allen, J.F., "An interval based representation of temporal knowledge," *Proc,* 7th 1JCAI, Vancouver, B.C., August 1981.

Allen, J.F. and O.K. Perrault, "Analyzing intention in utterances," *Artificial Intelligence 15,* 143-178, 1980.

Fikes, R.F. and N.J. Nilsson, "STRIPS: A new approach to the application of theorem proving to problem solving," *Artificial Intelligence* 2, 189 205, 1971.

McCarthy, J., "Programs with common sense," in M. Minsky (ed). *Semantic Information Processing.* Cambridge, MA: The MIT Press, 1968.

McDermott, I)., "Planning and acting," *Cognitive Science 2-2,* 1978.

McDermott, I)., "A temporal logic for reasoning about processes and plans," *Cognitive Science* 6, 101 155, 1982.

Nilsson, N.J. *Principles of Artificial Intelligence.* Tioga Press, 1980.

Sacerdoti, F.D. *A Structure for Plans and Behavior.* New York: Elsevier North-Holland, Inc., 1977.

Stefik, M. ,"Planning with constraints (MOLGEN: Part 1)," *Artificial Intelligence 16,* 2, May 1981.

Tate, A., "Generating project networks," *Proc,* 5th IJCAI, August 1977.

Vere, S., "Planning in time: Windows and durations for activities and goals," Jet Propulsion Laboratory, California Institute of Technology, November 1981.