TEMPORAL SCOPE OF ASSERTIONS AND WINDOW CUTOFF*

Steven Vere
Information Systems Division
Jet Propulsion Laboratory
Pasadena, California 91109

ABSTRACT

In a temporal planning and reasoning system, many logical assertions will have a limited life span: they are "terminated" by later, contradictory assertions. By observing assertion terminators, the lifetime of an assertion can be bounded within an interval called the scope. Assertion scopes can greatly reduce the number of relevant matches returned from an assertion database. Scopes and terminators also permit the more efficient determination of contemporaneous assertions for forward chaining of event rules. Window cutoff is an execution time accelerator for determining if two activities are ordered in a plan, a high frequency, utility operation in plan synthesis. The acceleration is accomplished by doing an early cutoff of search paths to the "lower" activity based on time constraints. These mechanisms reduced the execution time of a temporal planner called DEVISER by four orders of magnitude for 70 goals, allowing very long planning problems to be solved. The mechanisms described employ time constraints to accelerate planning, and are not applicable to precedence planners. DEVISER is a performance planner which has been applied experimentally in planning activities for the Voyager spacecraft in its encounter with Uranus in 1986.

I INTRODUCTION

In September 1983 I translated my temporal planner DEVISER I [Vere, 1983] from INTERLISP on the DEC-10 to ZETALISP on the Symbolics 3600 computer. The ZETALISP version executed about ten times faster and no longer ran out of memory on larger problems. These new circumstances enabled me to perform some experiments to see how execution time Increased as a function of the number of goals. The experiments were for a Voyager knowledge base, with a set of imaging goals which required no backtracking. The results are given in Figure 1, a semilog graph. DEVISER I showed a steep exponential growth which soon overwhelmed the Taster machine. Beyond 15 goals the graph is an extrapolation. Dissatisfaction with these results led me to consider time constraints as a means for improving performance. Since then a new version of the planner, DEVISER II, has been implemented which exhibits a much shallower performance curve. This improvement amounts to an estimated four orders of magnitude for 70 goals, and is due to the mechanisms which will be described. These mechanisms apply time constraints to Improve the efficiency of several important low level operations of plan synthesis. About three fourths of the slope improvement in Figure 1 is due to temporal scoping, and the remaining one fourth is due to window cutoff.

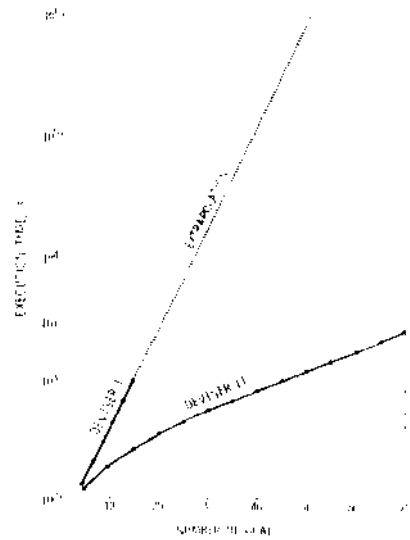

Figure 1. DEVISER Performance Curves

In the past, temporal planners have been investigated because many applications require time as an explicit parameter. The results of this investigation indicate that time can also serve as an important source of additional constraints which can accelerate planning.

DEVISER II lies along the evolutionary path of NOAH [Sacerdoti, 1977] and NONLIN [Tate, 1977]. These earlier planners did not model time. Such a

•This paper presents the results of one phase of research carried out at the Jet Propulsion Laboratory, California Institute of Technology, and sponsored by the National Aeronautics and Space Administration under Contract No. NAS-918.

program will be called a precedence planner, since it merely establishes the precedence of activities in a plan. Other approaches to incorporating time into a planner are to be found in [Allen and Koomen, 1983; Cheeseman, 1983; McDermott, 1982; and Salter, 1983]. These papers, with one exception, do not address the question of the efficiency or performance curves of the resulting system. Allen confessed that his system was rather inefficient even on blocksworld problems. None have attempted to take advantage of time constraints to improve the efficiency of the planner.

II REVIEW OF DEVISER WINDOW LOGIC

In a temporal logic system, the truth of an assertion (literal) must in effect be a function of time. (Throughout this paper the terms assertion and literal may be used interchangeably). In the "window logic" system of DEVISER, every assertion is associated with an activity, and every activity has associated with it a start time interval ("window") and a duration, which may be a function of the activity parameters.

In a temporal blocksworld where a PICKUP action requires 1 second, the action description is

```
(PICKUP ACTION
   ((CLEAR x)
    (ONTABLE x))
   --->
   ((HOLDING x)
    (NOT (ONTABLE x))
    (NOT (CLEAR x)))
(DURATION 1 SECOND))
```

Figure 2 shows the temporal relation between preconditions and postconditions. The extent of the preconditions and postconditions are indicated by timelines. Preconditions must become true on or before the start of the activity, and persist through the duration of that activity. Preconditions not contradicted by postconditions continue to hold. Postconditions (assertions) of an activity become true when the activity finishes and continue to hold indefinitely thereafter until they are explicitly contradicted by assertions of a later activity. If certain preconditions need only hold at the start of an activity (but not through the duration), that activity may be decomposed into subactivities whose preconditions behave as in Figure 2. This is explained in [Vere, 1983].

The figure shows the start time of the activity as a point in time. This is generally only true for activities which have been scheduled. During plan generation, the start time is typically an interval. This should be contrasted with the conventions of other temporal logic systems (e.g. [Allen and Koomen, 1983]) in which an interval specifies the lifetime of an assertion, and the start time is constrained to a point.

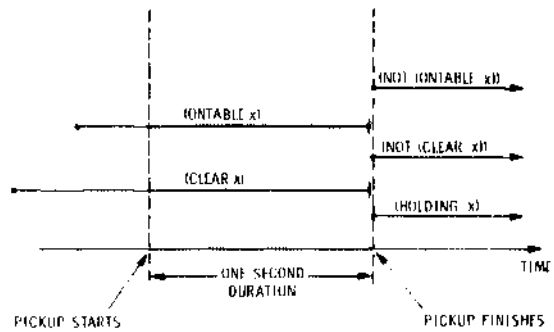

Figure 2. Timeline Diagram for the Pickup Action

Figure 3 shows a small blocksworld plan in which a deadline is imposed on the goal state. The plan start time is 0. The three actions in the plan have only interval constraints on their start times. The dashed lines may be ignored for the moment. Note that each activity except for the start and stop nodes has a start time window, duration, and set of assertions.

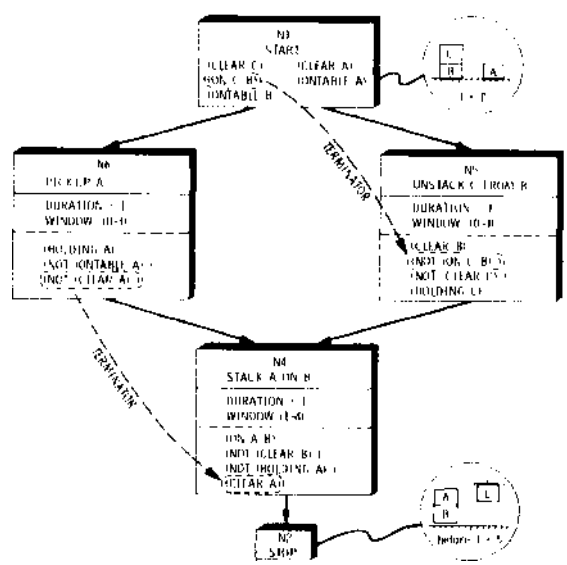

Figure 3. Terminator Relations in a Blocksworld Plan

Strictly speaking, it is activity nodes which are ordered and have start times, not assertions. However, it will be convenient to speak of assertions being ordered or having start times when in fact these relations apply to the nodes to which those assertions belong.

III ASSERTION TERMINATORS AND TEMPORAL SCOPES

In a temporal logic, assertions may in the course of time cease to be true. However, they do not die naturally; instead, they are terminated by later contradictory asertions.

Assertion T is defined to be a terminator for assertion A iff: A and T are contradictory; T follows A in time; and no assertion T' exists satisfying the first two conditions such that T follows T^1 in time.

Usually an assertion has a single terminator, or none at all. However, because activities in a plan are only partially ordered, it is possible for an assertion to have several terminators which are mutually unordered. Also, as planning progresses and new activities are introduced and ordered, the terminator(s) of an assertion may change.

Referring back to Figure 3, the dashed lines connect selected assertions and their terminators. Thus (CLEAR A) of N4 is the terminator of (NOT (CLEAR A)) of N6. In the case where an assertion has no terminator, it holds forever. (ON A B) of N4 is one example of an unterminated assertion.

The temporal scope of an assertion A is defined as an interval $(t_1\ t_2)$ where t1 is the earliest finish time of the assertion's activity node and t2 is the minimum of the latest finish times of the terminators of A, or infinity if the assertion is unterminated. Thus the scope bounds the lifetime of the assertion. Since activity start times and terminators change as planning proceeds, so do temporal scopes. Each time the latest start time or duration of an activity is revised, a check is made of all assertions of that activity. If an assertion serves as a terminator of an earlier assertion, the scope of that earlier assertion may have to be revised too.

Again referring to Figure 3, we can deduce that the temporal scope of (NOT (CLEAR A)) in N6 is (1 5) since the earliest finish time of N6 is 1 and the latest finish time of N4, which asserts the terminator (CLEAR A), is 5. Figure 4 illustrates the notion of temporal scopes in the abstract, and shows their relation to earliest and latest finish times for assertions and their terminators.

Maintaining temporal scopes for each assertion during plan synthesis dramatically improves the overall efficiency of the planner for large goal sets. In a planning engine such as DEVISER, processing of assertion retrievals is a fundamental contributor to total execution time. Screening out irrelevant retrievals with the temporal scope meohanism vastly shortens the number of assertion candidates which must be sorted and processed by fundamental, high frequency planning operations.

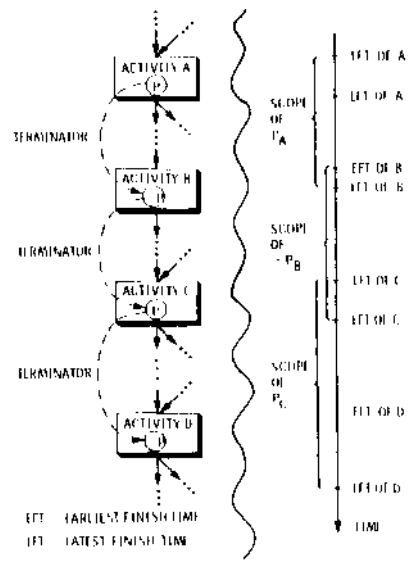

Figure 4. Terminators and Scopes in the Abstract

IV TIME BOUNDED ASSERTION RETRIEVALS

The primary application of terminators and scopes is in screening retrievals from the distributed assertion database. Without scopes, the retrieval mechanism must return a list of all literals matching a given pattern. The subroutine requesting the retrieval must then process this list to determine the "relevant" matches, and this is time consuming. The function BOUNDEDMATCHES is the low level function which uses scopes in accessing this database. BOUNDEDMATCHES takes as arguments an assertion and a time interval. It returns a list of all assertions in the database which match the given assertion and whose scopes intersect the given interval. Thus we are able to filter out most irrelevant retrievals with a quick numerical test. Of course this trick is going to work only in a planner which models time.

BOUNDEDMATCHES is called with high frequency in the planner in the following operations:

- looking for possible "tie-ins'' to allow phantom node creation;
- finding conflicts after an expansion of a node or instantiation of a literal;
- checking phantom node violation after a literal instantiation;
- checking for infinite loops in the backtracking process;
- checking contemporaneous literals for forward chaining.

To illustrate, we will examine the first operation in more detail. If the assertion of a goal node matches an assertion true in the plan "above" the goal node (i.e. earlier), that goal node can be converted to a phantom node, which indicates that no action is required to satisfy the precondition. The situation is shown in Figure 5. B is a goal node for activity C, and B's assertion is (P). We want to know if there is some activity node A above which asserts (P) and if (P) still holds down at C. A necessary condition is that the scope of a matching literal, such as (P) of node A, have a scope intersecting the critical interval bounded by the earliest finish time and latest finish time of C. The result is that BOUNDEDMATCHES is able to return just the few matches overlapping the critical interval, rather than the hundred or more that might exist in a long plan. Of course, these bounded matches must still be further screened to select those literals which satisfy all requirements for a tie-in.

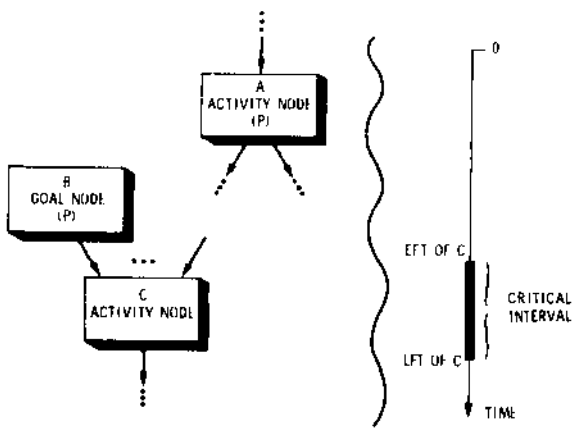

Figure 5. Application of Scopes to Phantom Node Creation

V FINDING CONTEMPORANEOUS LITERALS

Literals are contemporaneous if they are simultaneously true at some instant of time. In a dynamic temporal logic system, to determine if a forward-chaining event rule should fire, it is necessary to determine if there are contemporaneous literals which match the rule antecedent. The situation is more complex than in conventional forward chaining systems such as OPS5 [Forgy, 1982], in which there is a single set of assertions all simultaneously true. In a parallel plan, the assertions are sprinkled throughout the plan and it oan be an expensive computational process to find contemporaneous matches for the antecedent. Here too the concepts of scope and terminators can help to accelerate the computation.

To determine if a set of candidate literals is contemporaneous, their scopes may be intersected, a computationally cheap operation. If this intersection is null, the literals cannot be contemporaneous. However, a non-null intersection unfortunately is not sufficient. Figure 6 shows a segment of a plan diagram which illustrates this. Clearly assertions (A) and (B) are not contemporaneous, since the terminator of (A) is a predecessor of (B). Yet their scopes intersect: (A), (2 11); (B), (7 INFINITY).

This example illustrates the need for a second condition: for a set of literals to be contemporaneous, no literal in the set may follow the terminator of another literal in the set.

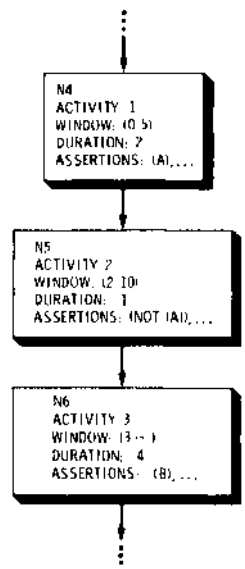

Figure 6. Non-null Scope Intersection Counterexample

VI WINDOW CUTOFF

Window cutoff is a mechanism for accelerating the determination of whether one activity "follows" another in a plan , i.e. whether the activities are ordered. This is a fundamental, high-frequency utility operation in plan synthesis which contributes heavily to the total execution time of the planner. The ordering relation is the transitive closure of the successor relation, which is denoted by the solid directed arcs in a plan diagram such as Figure 3. Successors of an activity are stored explicitly in DEVISER II, but the ordering relation must be computed. (To store the ordering relation explicitly would be unthinkable since successors are constantly changing as the plan develops). For a precedence planner, determining if one activity follows another is straightforward but expensive. You Just begin at one node and do a blind search down

through the plan diagram looking for the other node. In a temporal planner, window cutoff provides a mechanism for early termination of search paths based on time constraints which must hold between ordered nodes.

Suppose we are trying to determine if activity node NB follows activity node NA in a plan. We begin by examining the successors of NA to see if NB is among them. If not, we must search among the successors of the successors . . ., keeping a record of nodes visited, until NB is found or all nodes reachable from NA have been visited. Suppose that we arrive at node NC somewhere below NA, that NC has not already been visited, and that NC is not NB. In a precedence planner we must now search the plan below NC. However, in a temporal planner we may be able to cut off the search at NC if the windows of NC and NB are such that NB could not possibly lie below NC.

Following is a tutorial version of the function ORDERED? which illustrates window cutoff:

```
;-----------------------------------------------
(DEFUN ORDERED? (NODE.A NODE.B VCODE)
   (PROG (A.DURATION RESULT)

   (IF (NULL VCODE) THEN
   !  (COUNTER = COUNTER + 1)
   !  (VCODE = COUNTER)
   )
   (A.DURATION = (DURATION.OF NODE.A))
   (IF (AND (<= ((EST NODE.A) + A.DURATION)
   !            (EST NODE.B))
   !        (<= ((LST NODE.A) + A.DURATION)
   !            (LST NODE.B))) THEN
   !  (FOREACH NODE.C IN (SUCCESSORS NODE.A)
   !  !  UNTIL RESULT DO
   !  !  (IF (NOT (VISITED? NODE.C VCODE)) THEN
   !  !  !  (RECORD.VISITATION NODE.C VCODE)
   !  !  !  (IF (EQ NODE.C NODE.B) THEN
   !  !  !  !  (RESULT = T)
   !  !  !  ELSE
   !  !  !  !  (RESULT = (ORDERED? NODE.C NODE.B
   !  !  !  !      VCODE))
   !  ) ) ) )
   (RETURN RESULT)
))
;-----------------------------------------------
```

VII CONCLUSION

The notions of temporal scopes and terminators contribute to the screening of assertion retrievals, and to the determination of contemporaneous assertions for forward chaining in a temporal planner. Window cutoff oan accelerate the determination of the ordering relation on activities. These features dramatically improved the execution time performance of DEVISER, enabling it to solve in a few hours large planning problems for the Voyager spacecraft consisting of over one hundred goals. This is at least an order of magnitude faster than human performance. The mechanisms for this performance improvement apply only to temporal planners, since they are based on time constraints.

REFERENCES

[1] Allen, J. F. and J. A. Koomen, "Planning Using a Temporal World Model." In Proc. IJCAI-83, 741-747.

[2] Cheeseman, P. "A Representation of Time for Planning," Tech. Note 278, AI Center, SRI International, Menlo Park, California, Feb. 1983.

[3] Forgy, C. L. "Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem." Artificial Intelligence, Vol. 19, No. 1, September 1982, 17-37.

[4] McDermott, D. "A Temporal Logic for Reasoning About Processes and Plans." Cognitive science, Vol. 6, 1982, 101-155.

[5] Sacerdoti, E. D. A Structure for Plans and Behavior. Elsevier, New York. 1977.

[6] Salter, R. M. "Planning in a Continuous Domain—An Introduction." Robotlca, Vol. 1, 1983, 85-93.

[7] Tate, A. "Generating Project Networks." In Proc. IJCAI-77, 888-893.

[8] Vere, S. A. "Planning in Time: Windows and Durations for Activities and Goals." IEEE trans. on Pattern Analysis and Machine intelligence, vol. PAMI-5, No. 3, May 1983, 246-267.