Representing Actions with an Assumption-Based Truth Maintenance System

Paul H. Morris and Robert A. Nado

IntelliCorp
1975 El Camino Real West
Mountain View, California 94040

ABSTRACT

The Assumption-based Truth Maintenance System, introduced by de Kleer, is a powerful new tool for organizing a search through a space of alternatives. However, the ATMS is oriented towards inferential problem solving, and provides no special mechanisms for modeling actions or state changes. We describe an approach to applying the ATMS to the task of representing the effects of actions. The approach extends traditional tree-structured context mechanisms to allow context merges. It also takes advantage of the underlying ATMS to detect inconsistent contexts and to maintain derived results. Some results are presented concerning possible approaches to the treatment of merges in questionable circumstances. Finally, the analysis of actions in terms of a truth maintenance system suggests the need for a more elaborate treatment of contradiction in such systems than exists at present.

1. Introduction

The Assumption-Based Truth Maintenance System (ATMS), introduced by de Kleer [2], is a powerful new tool for organizing an efficient search through a space of alternatives. By explicitly recording the dependence of the reasoning steps on individual choices, a truth maintenance system is able to share partial results across different branches of the search space. In effect, knowledge gleaned in one context is automatically transfered to other contexts where it is relevant. The ATMS permits simultaneous reasoning about multiple, possibly conflicting contexts, avoiding the cost of context switching.

The ATMS as presently constituted views problem solving as purely inferential. This is an appropriate stance for a broad class of constraint satisfaction problems. However, problems involving temporal changes or actions require some additional mechanism. As de Kleer [5] points out, "... problem solvers [may] act, changing the world, and this cannot be modeled in a pure ATMS in which there is no way to prevent the inheritance of a fact into a daughter context." In this paper we explore one approach to using the ATMS to support the modeling of actions. The basic idea is to extend a traditional tree-structured context mechanism (as in CONNIVER and QA4 [1]), taking advantage of an underlying ATMS to allow context merges, to detect inconsistent contexts, and to maintain derived results. This approach has been implemented in the KEEworlds^{TM} facility of the KEE^{TM} (Knowledge Engineering Environment^{TM}) system.*

*KEEworlds, KEE and Knowledge Engineering Environment are trademarks of IntelliCorp.

This research was supported in part by the Defense Advanced Research Projects Agency (DARPA) under contract No. F30602 85 C 0065. The views and conclusions reported here are those of the authors and should not be construed as representing the official position or policy of DARPA or the U.S. government.

In the following sections, we give a functional overview of the KEEworlds facility. We then describe the underlying representation in terms of the ATMS. Special attention is given to the situation where a world has multiple parents. This is followed by a discussion of non-monotonic reasoning about actions in a more general TMS setting, suggested by the worlds mechanism. We close with some remarks about related systems.

2. Worlds

The basic structure provided for modeling actions is a directed acyclic graph of *worlds*. Each world may be regarded as representing an individual, fully specified action or state change. A world together with its ancestors in the graph represents a partially ordered network of actions. Each successor of a world in the graph then represents a hypothetical extension of the world's associated action network to include a new subsequent action. The world graph as a whole may thus be regarded as representing multiple, possibly conflicting, action networks. Each partially ordered action network resembles a procedural net of NOAH [9], or NONLIN [10], where the actions are fully specified. We assume that the effects of a fully specified action can be represented by additions and deletions of base facts, so each world has a set of additions and deletions associated with it that represent the actual primitive changes determined by the action. Since an action corresponds to an *application* of an operator, not an operator itself, this assumption is somewhat less restrictive than that of STRIPS [8], which requires *operators* to have fixed add and delete lists.

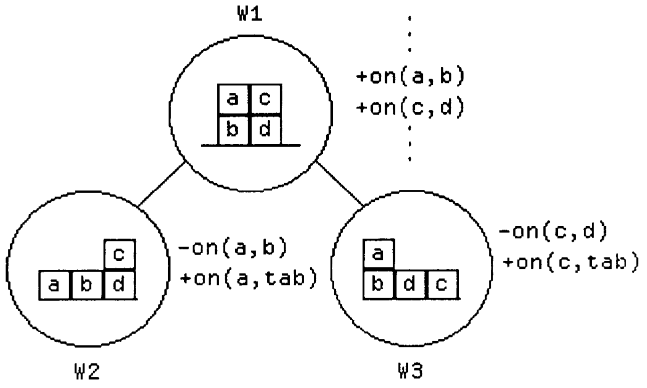

Figure 2-1: A Worlds Graph from Blocks World

Figure 2-1 shows an example worlds graph from the blocks world. The additions at W1 produce the initial state, while the deletion and addition at W2 represent the effect of moving block *a* to the table. Those at W3 correspond to an alternative action of moving block *c* to the table. Notice that preconditions are not represented. These are assumed to have been tested in the parent world before W2 and W3 were constructed. Only the *effects* of the actions are recorded in the worlds.

To simplify the discussion we will assume for the moment that the graph is a tree, as in figure 2-1, i.e., each world has at most one parent and a branch of the tree corresponds to a linear sequence of actions. Later, we will consider the consequences of multiple parents.

Observe that we may associate each world with the state that results from applying the changes encoded by the world and all of its ancestors (in figure 2-1 the states are depicted inside the worlds). Hence, a world plays a double role, representing both a state change and a state. The facts in the state will in general be augmented with deductions using general knowledge of the domain. Thus, the facts that are true at a world fall into the following three categories:

1. facts inherited from ancestor worlds

2. direct additions at this world

3. deductions from facts in 1 and 2

In keeping with the view that additions and deletions represent actual changes, they are only recorded where they are *effective*, that is, an addition only occurs where the fact did not previously hold, and a deletion where it did hold.

The inherited facts follow a principle of inertia (essentially the STRIPS assumption [11]): a fact that is added at a world continues to be true in succeeding worlds, up until (but not including) a world where it is deleted.

The deduced facts may include the distinguished fact FALSE, representing a contradiction. A world where FALSE can be deduced is marked as inconsistent. The system generally avoids further reasoning in such worlds (however, it is possible and sometimes useful to do meta-level reasoning about inconsistent worlds).

3. Worlds in ATMS

Before discussing how the worlds graph is implemented in terms of the underlying ATMS, we give a brief sketch of the ATMS mechanisms that are used, primarily to establish terminology. The reader is urged to consult de Kleer [3, 4, 5] for a full description of the ATMS.

The basic elements of the ATMS are *assumptions* and *nodes*. An assumption in the ATMS corresponds to a decision or choice, and is used as an elementary context descriptor. Nodes correspond to propositional facts or data, which may be justified in terms of other nodes, or assumptions. By tracing back through the justification structure, it is possible to determine the ultimate support for a derivation of a node as a set of assumptions. Such a set is called an *environment* for the node. Since a node may have multiple derivations, it may also have multiple environments. The set of (minimal) environments for a node is called its *label*. Computing the labels of nodes is one of the major activities of the ATMS.

The primary transaction that the ATMS supports is adding a justification. This causes the labels of affected nodes to be recomputed. There is a special element called FALSE, denoting contradiction, which is similar to a node, and may have justifications. The environments that would be in its label are called *nogoods* and constitute minimal inconsistent environments. Environments that are discovered to be inconsistent, i.e., that are supersets of nogoods, are removed from the labels of nodes so that they are not used for further reasoning.

Each world has two ATMS entities associated with it, reflecting its double role: a *world assumption* and a *world environment*. The world assumption corresponds to the action encoded by the world, and may also be thought of as the choice or decision that led to the action. The world environment, on the other hand, corresponds to the state, and actually consists of the set of world assumptions from the given world and all of its ancestors. It is convenient to use the ATMS itself to compute the world environment. This is accomplished by having a special *world node* associated with each world. This node may be thought of as representing the statement that the world's action occurs.

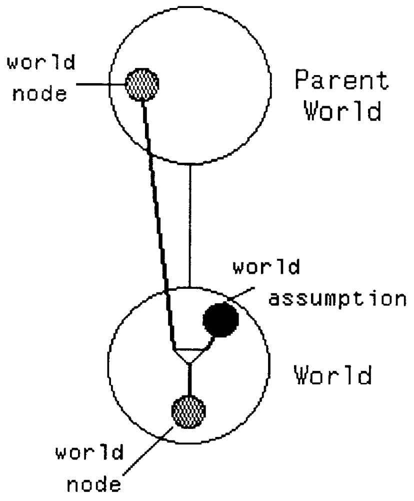

Figure 3-1: Justification for World Node

The world node, N_W, is given a single justification

$$N_{WP} \wedge A_W \rightarrow N_W$$

where N_{WP} is the world node of the parent, and A_W is the world assumption of the given world. This is depicted graphically in figure 3-1. It is not difficult to see that this results in all world nodes having a single environment, of the form described.

Directly adding a fact F at a world can now be accomplished by supplying a justification in terms of the world node. However, to allow for the possibility of later deletion, a *nondeletion* assumption is included. Thus, the justification has the form

$$N_W \wedge A_{W,F} \rightarrow F$$

where $A_{W,F}$ is the nondeletion assumption. A distinct nondeletion assumption is required for each separate addition of a fact at a world (to allow independent deletion). If F is deleted at a subsequent world W1, the justification

$$A_{W1} \wedge A_{W,F} \rightarrow FALSE$$

is supplied to the ATMS, where A_{W1} is the world assumption for W1. We will call nogoods resulting from justifications of this form *deletion* nogoods. This scheme for addition and deletion is shown in figure 3-2.

Apart from the justifications supplied by the system to represent additions and deletions, and justifications for world nodes, there will be justifications installed by the user to represent deductions from the primitive facts. These deductions need be performed only once as the presence of the justifications in the ATMS allows the efficient determination, via label propagation, of which derived facts hold in which worlds.

Derivations of FALSE arising from user justifications are used to determine inconsistent worlds, representing dead ends in the search.

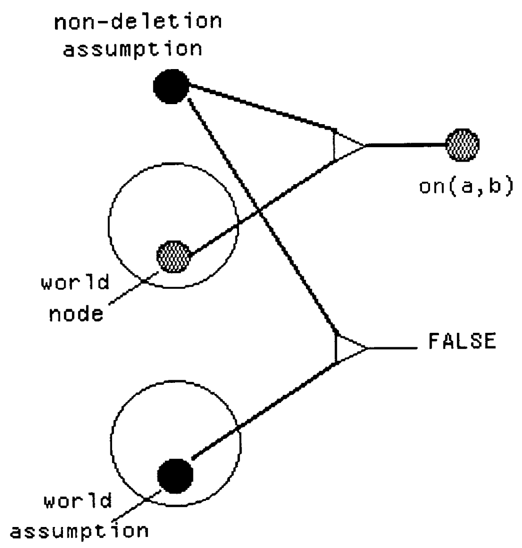

Figure 3-2: Addition and Deletion Justifications

The nogoods determined by the ATMS may, however, contain nondeletion assumptions in addition to the world assumptions. However, only the latter represent choices in the search, and we wish these to take all the "blame" for dead ends (we discuss this further in section 5). Thus, the multiple worlds system incorporates a feedback loop that installs in the ATMS reduced nogoods with the nondeletion assumptions removed. These nogoods are subsets of the original ones, and so, in accordance with the minimality requirement, the latter are removed. Deletion nogoods are, of course, exempt from this process; the feedback procedure ensures that the deletion nogoods are the only ones left that contain nondeletion assumptions.

To test whether a fact holds in a world, we can compare each environment in the node label with the world environment. The comparison is done as follows (in principle; the actual algorithm is equivalent, but more efficient). The world environment is extended with as many nondeletion assumptions as are consistent with it (the extension is necessarily unique since each nogood contains at most one nondeletion assumption). The extended world environment is then checked to see if it is a superset of the fact environment. If so, the fact is regarded as true in the world. For example, in figure 2-1, the world environment at W2 includes the world assumption for W1. When extended, the environment also contains the nondeletion assumption for the addition of on(c,d) at W1. Thus, on(c,d) is true at W2. Note, however, that the nondeletion assumption for the addition of on(a,b) at W1 cannot be consistently added to the environment at W2 because it shares a deletion nogood with the world assumption for W2. Hence on(a,b) fails to be true at W2.

4. Merges

We now consider the more complex situation where a world has multiple parents: we call such a world a *merge*. The ability to perform merges allows a problem to be decomposed into nearly independent components, which can be worked on separately and later recombined. As before, the changes represented by the ancestor worlds are combined. In figure 4-1, the world W4 is a merge. Thus, the state corresponding to W4 will have both blocks moved to the table. We wish to stress that a merge is not the same as a simple union of the facts in the parent worlds, but rather

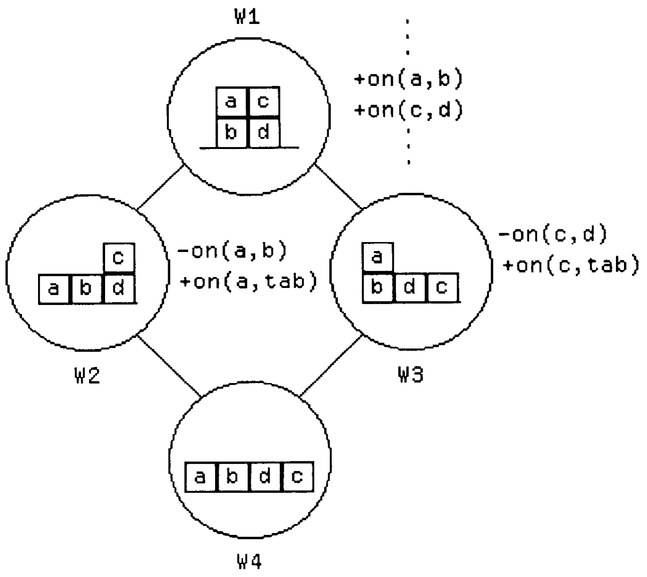

Figure 4-1: A Merge

produces the state resulting from a union of the *changes* from all the ancestor worlds.

In the example of figure 4-1, the changes along the two branches are independent. More generally, a difficulty arises in that the effect of changes may depend on the order in which they are applied, resulting in an ambiguous merge. In figure 4-2, we show two examples of such merges. In both cases, the state at W5

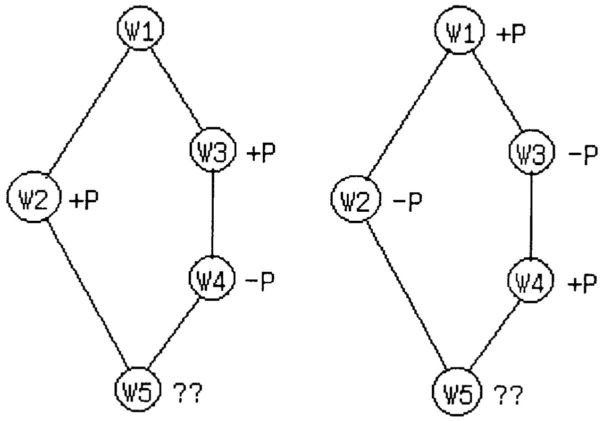

Figure 4-2: Ambiguous Merges

depends on the order of the preceding changes.

There are basically two ways of dealing with this difficulty. One is to forbid the merge in ambiguous cases. The other is to refine the definition of the merge so that the ambiguity is removed. It is also possible to adopt an intermediate position, forbidding some merges and further specifying others. We now consider examples of each approach.

We have already introduced the requirement that additions and deletions at worlds be effective with respect to the state resulting from actions in ancestor worlds. However, from a strict standpoint of fully specified actions, the additions and deletions could be required to be effective even with respect to possible states resulting from actions in sibling or cousin worlds. Thus, one might forbid a merge if the ancestor subgraph of the proposed merge possesses any linearization in which an addition or deletion is ineffective. One can then prove the following.

> **Theorem 1:** A merge that is not forbidden by the above criterion is unambiguous.

The next result assists in the identification of such forbidden merges (remember we are assuming that additions are always effective with respect to ancestor worlds).

> **Theorem 2:** A graph of worlds admits a linearization in which an addition is ineffective if and only if there are at least two worlds in which the addition occurs such that neither is an ancestor of the other.

A similar result holds for deletions. With this approach, the merges in figure 4-2 would be disallowed.

It is of interest that the above restriction resembles that required for conflict-free procedural nets [10] where actions that violate each others' preconditions may not be unordered relative to each other. Indeed, additions and deletions that are mandatory are, in effect, preconditions. From this perspective, the separate branches of the networks of figure 4-2 are in conflict because each branch deletes a precondition of the other.

In a sense, the above restriction is excessive: it forbids many merges that are unambiguous. If one does not require that additions and deletions be effective with respect to non-ancestor actions, a somewhat weaker sufficient condition (but still not necessary) that guarantees unambiguous merges can be obtained, as follows.

> **Theorem 3:** A sufficient condition for a merge to be unambiguous is that the ancestor subgraph may not contain two worlds, one of which deletes a fact and the other of which adds it, such that neither is an ancestor of the other.

This criterion also prohibits the examples of figure 4-2.

We now follow the other approach to removing the ambiguity, and adopt additional criteria for defining the merge. In the *pessimistic merge*, an individual fact belongs to the merge if it survives in *every* linearization of the actions. The rationale is that we may then be assured the fact holds, irrespective of the order in which the actions are performed. Otherwise, we are ignorant of the fact, and the absence of the fact from the merge simply denotes such ignorance, not falsity. Notice that this definition is consistent with the original when the merge is unambiguous. With the pessimistic merge, the fact P is absent at W5 in both examples of figure 4-2. A dual to the pessimistic merge is the *optimistic merge* where a fact is true in the merge if it is true in *some* linearization. Again, this is consistent with the original for the unambiguous case. With the optimistic merge, P is present at W5 in both examples.

We now discuss the effect of merges on the ATMS representation. When a world has multiple parents, the justification for the world node must be expanded to include each of the parent world nodes among the justifiers. The justification scheme for additions and deletions is unchanged. The different kinds of merge are obtained by different selections of which additions the deletions affect, i.e., which justifications for FALSE are entered. For the pessimistic merge, the deletions are effective with respect to all except descendant additions. For the optimistic case, the deletions are effective with respect to ancestor additions only (the optimistic merge tends to be easier to implement efficiently, although less defensible on semantic grounds).

One might imagine a wide variety of possible merge algorithms. There are two overriding constraints that led to the schemes described here. One is the necessity of quickly determining whether a potential merge would produce a consistent world, since that is expected to be a high frequency operation. The schemes described allow the merge to be computed as a simple union of ATMS environments. The other constraint is the existence of a large core of unambiguous cases where there is only one reasonable value for the merge.

A further merge type that has some intuitive appeal, but does not appear to admit an efficient implementation, arises as follows. Let us call a linearization of the ancestor subgraph of a proposed merge *valid* if it results in every addition and deletion being effective. It is possible to show that every valid linearization gives the same result for the merge. Thus, one might define the merge to be this common value (if there is any such linearization; the merge could be forbidden otherwise). In figure 4-2, this would lead to P holding at W5 in the left example, but not in the right.

5. Actions and NonMonotonicity

It is instructive to consider how actions might be represented in a more general TMS setting, as suggested by the worlds system. For definiteness, and for contrast, this will be cast in terms of a Doyle-style truth maintenance system [6]. The general approach we follow is to use a form of nonmonotonic inference to reason about the effects of actions. However, the behavior we require in response to contradiction is somewhat different from the standard approach in truth maintenance systems.

We will regard a context, or current state of the system, as describing the evolution of a situation to a particular point in time. Besides containing assertions about facts in the "present" such as "block *a* is on block *b*," the context records past actions like A3: "block *a* was placed on block *b*". Note that there may be several occurrences of individual actions with the same description; we distinguish between the occurrences by giving them unique identifiers such as A3. The numbering of the identifiers is not intended to imply temporal order. Thus - so far - the relative timing of past actions has not been represented.

The positive effects of an action can be represented by justifications linking the occurrence of past actions to present facts. For example,

$A3 \wedge P5 \rightarrow$ block *a* is on block *b*.

P5 is a preservation condition of the form "block *a* was not moved off block *b* after A3." In order to allow deletion, we justify P5 as an assumption by giving it a nonmonotonic justification of the form

$(D5) \rightarrow P5$

Here, "(D5)" indicates that D5 is an OUT-justifier, where D5 is the statement that "some action after A3 moves block *a* off block *b*". If a subsequent action, say A4, moves the block off, we supply a justification

$A4 \rightarrow D5$

causing the OUT-justifier to come IN, thereby undercutting the derivation of "block *a* is on block *b*." Note that the information about the relative timing of actions is now implicitly represented by these justifications.

A difficulty with this representation arises when the problem solving process generates contradictions that represent dead ends in the search space. We do not wish the preservation assumptions to be implicated in these; rather, we wish the assumptions representing choices of actions to be the ones considered for revision. Choosing a preservation assumption as culprit during backtracking would amount to postulating the existence of an unknown action that deletes one of the facts leading to the contradiction. However, if we make the separation between problem solving and truth maintenance suggested by de Kleer, then from the point of view of the TMS, the only actions that exist are those that the problem solver has informed it about. Some new mechanism is required to ensure that the TMS handles this correctly. One possibility is to have something like a "sheltered" assumption, which could be refuted directly, but not indirectly in response to a contradiction.

Incidentally, the need for a more discriminating process of culprit identification is not confined to the difficulty with preservation assumptions. As another example, consider a situation where a burglar is planning to break into a house late at night. To accomplish his purpose, he must choose some method of entry. One method is to break in a window. However, this may have the consequence of waking the occupants, if they are home, which would defeat his purpose. Let us suppose the burglar makes the default assumption that the occupants are home. The difficulty is that a standard truth maintenance system, in attempting to resolve the "contradiction" of waking the occupants, might elect to revise the assumption that the occupants are home, even though that is not subject to the burglar's control, instead of the real culprit, breaking the window. The system would in effect regard the undesired consequence of waking the occupants as evidence for their absence. However, it is only when there is independent evidence for the occupants being absent that this possibility is worth considering. This example of "wishful thinking" suggests that truth maintenance systems in general need a more refined treatment of contradiction handling.

Although the approach outlined here could be adapted to using the ATMS more directly for modeling actions, it would be cumbersome for a user to have to input the justifications representing additions and deletions by hand. The worlds facility described earlier provides a framework that presents a more convenient interface to an action modeling system.

6. Closing Remarks

The worlds considered here resemble the data pools of McDermott [7]. However, the result of a merge in the data pool approach is determined by the arbitrary chronological order in which items are recorded in data pools. This means that two graphs with the same apparent external structure may have different results for a merge. Another difference is that data pools apparently have no notion of contradiction (at least none is mentioned by McDermott in the paper). One attractive aspect of McDermott's approach is that justifications may have OUT-justifiers. However, this requires that labels be computed by solving Boolean equations, rather than the simple propagation procedure of the ATMS.

The Viewpoints^{TM} facility of Inference Corporation's ART^{TM} system appears quite similar in behavior to the worlds facility described here.* However, it is difficult to make detailed comparisons since little information has been made available about the underlying mechanisms of ART.

We have described an approach to constructing a context mechanism that represents a partially ordered network of actions or state changes. A realization of the mechanism has been described in terms of an underlying Assumption Based Truth Maintenance System. An examination of a similar representation in a classical TMS system suggests a shortcoming in the way existing truth maintenance schemes handle contradictions.

The approach described has been implemented as part of the KEEworlds facility of KEE and appears to provide a useful and efficient tool for reasoning about multiple situations. The KEEworlds facility integrates the multiple worlds system with an existing frame-based representation system, provides a graphical browser for manual exploration of worlds and allows rule-based generation of worlds during either forward or backward chaining. An application-oriented discussion of the KEEworlds facility, together with an example of its use, may be found in [8].

* Viewpoints and ART are trademarks of Inference Corporation.

References

[1] Bobrow, G. and B. Raphael.
New Programming Languages for Artificial Intelligence Research.
Computer Surveys 6(3):153-174, 1974.

[2] de Kleer, J.
Choices Without Backtracking.
In *Proceedings, AAAI-84*. Austin, Texas, 1984.

[3] de Kleer, J.
An Assumption-Based Truth Maintenance System.
Artificial Intelligence 28(1), 1986.

[4] de Kleer, J.
Extending the ATMS.
Artificial Intelligence 28(1), 1986.

[5] de Kleer, J.
Problem Solving with the ATMS.
Artificial Intelligence 28(1), 1986.

[6] Doyle, J.
A Truth Maintenance System.
Artificial Intelligence 12(3), 1979.

[7] McDermott, D.
Contexts and Data Dependencies: A Synthesis.
IEEE Transactions on Pattern Analysis and Machine Intelligence 5(3):237-246, May, 1983.

[8] Nardi, B. and A. Paulson.
Multiple Worlds With Truth Maintenance In AI Applications.
In *Proc. ECAI-86*. Brighton, England, 1986.

[9] Nilsson, N.J.
Principles of Artificial Intelligence.
Tioga Publishing Company, Palo Alto, Ca., 1980.

[10] Sacerdoti, E.D.
A Structure for Plans and Behavior.
Elsevier North-Holland, 1977.

[11] Tate, A.
Generating Project Networks.
In *IJCAI-77*, pages 888-893. Cambridge, Massachusetts, 1977.

[12] Waldinger, R.J.
Achieving Several Goals Simultaneously.
In Elcock, E. and Michie, D. (editor), *Machine Intelligence 8*, pages 94-136. Ellis Horwood, Chichester, 1977.