Goal Ordering in Partially Ordered Plans

Mark Drummond
Sterling Federal Systems
NASA Ames Research Center
Mail Stop: 244-17
Moffett Field, CA 94035

Ken Currie*
AI Applications Institute
University of Edinburgh
80 South Bridge
Edinburgh, UK EH1 1HN

Abstract

Partially ordered plans have not solved the goal ordering problem. Consider: a goal in a partially ordered plan is an operator precondition that is not yet achieved; operators, orderings and variable bindings are introduced to achieve such goals. While the planning community has known how to achieve *individual* goals for some time, there has been little work on the problem of *which* one of the many possible goals the planner should achieve next. This paper argues that partially ordered plans do not usefully address the goal-ordering problem and then presents a heuristic called *temporal coherence* which does. Temporal coherence is an admissible heuristic which provides goal-ordering guidance. Temporal coherence is admissible in the sense that if a solution exists in the planner's search space, then there will be a series of goal achievements permitted by the heuristic which can produce this solution.

1 Introduction

Most planners using partially-ordered plans operate by repeatedly transforming a plan until it meets certain requirements. An important and typical requirement is that the given plan have no operators with false preconditions. Search must continue until all operator preconditions are true according to the plan's operators, orderings and bindings. NonLin [Tate, 1977] was the first planner to demonstrate the ability to transform plans in this manner through search. Of course, NOAH [Sacerdoti, 1977] first introduced the basic idea, but it was unable to backtrack over incorrect decisions. NonLin was more general in the sense that it was able to reconsider previous choices; that is, it could *search* for a successful plan.

Chapman [1987] has given us the Modal Truth Criterion (the MTC) as a statement of the conditions under which a precondition will be true at a point in a partially ordered plan. The MTC is intended to characterize NonLin's [Tate, 1977] goal achievement procedure. The MTC is simply a characterization of the conditions under which an operator precondition will be true at a point in a partially ordered plan. To actually build a planner, one must make this characterization *effective* by transforming it into an algorithm for achieving outstanding preconditions. Such an algorithm will be a key component of any planner. We will refer to any precondition-achievement algorithm based on the MTC as a *goal achievement procedure.*

*This work has been partially supported by the Science and Engineering Research Council under grants GR/E/05421 and GR/D/58987, and by the Air Force Office of Scientific Research, Artificial Intelligence Research Program.

Chapman's planner, TWEAK, explores its space breadth-first. Practical planners cannot afford this luxury [Wilkins, 1984; Currie and Tate, 1985]. Heuristics for selecting among the plan modification operations sanctioned by a planner's goal achievement procedure are required if plans are to be produced in acceptable time.

Goal ordering is a problem, even if a planner uses partially ordered plans. Suppose that a typical plan for the blocks world has on average 4 outstanding goals. Suppose as well that there are on average 3 ways to achieve each of these goals. This gives us, on average, 12 ways to change an arbitrary blocks world plan into another one. Each change is designed to achieve a single precondition. For a typical blocks world problem, suppose that 7 plan modifications are required to change an initially provided plan into one which has no outstanding goals. Breadth-first search must therefore explore (at worst) 12^7 partial plans. And the blocks world is *easy* compared to real domains. In part, this explosion results from the different choices of the *goal to work on next.* If this choice can be effectively managed then the search can be made more efficient.

A planner's goal achievement procedure says nothing about an order in which to pursue goals. The heuristic of temporal coherence addresses this problem. It avoids working on plans whose bulk preconditions are not consistent. The bulk preconditions for a plan are the overall conditions on which the plan depends for its successful execution; these preconditions are consistent if they describe a physically realisable domain state. According to this heuristic, if a plan's bulk preconditions are not consistent, then the plan has internal inconsistencies and is best avoided.

This paper is about the goal ordering problem in partially ordered plans, and is organized as follows. Section 2 explains why partially ordered plans do not provide a solution to the goal ordering problem. Section 3 presents a heuristic which does.

2 Orderings of Goals and Operators

2.1 Partially Ordered Plans: A Class of Data Structures

A partially ordered plan is typically defined to be a set of operators and a strict partial ordering over that set. Such plans were initially called *nonlinear,* in the sense that planned operators were not necessarily totally ordered in time [Sacerdoti, 1977]. The original intuition behind this was to postpone decision making as long as possible in the hope of more efficient plan construction. The actual complexity of reasoning about partially ordered plans is made clear by Dean and Boddy [1988]. In this paper, we concentrate on the control problem of *goal ordering.*

2.2 The Linearity Assumption: An Approach to Control

Planners must search to construct plans. After first selecting a plan in the search space to work on, a planner must next select *one* of the outstanding goals in that plan to achieve. This goal selection is an issue only for the planner's *control* mechanism. The choice has nothing to do with the particular plan representation used. Planners using totally ordered plans have the same decision to make. *The plan representation impacts only the way in which a goal can be achieved* for partially ordered plans, a goal achievement procedure based on the MTC is appropriate. The problem of goal achievement ordering is unaddressed until something is said about the planner's control structure. The problem of deciding which goal to work on is called the *goal ordering problem.*

Sussman [1973] presented an approach to the goal ordering problem. His approach was based on the "linear assumption"; namely, that ^{u}subgoals are independent and thus can be sequentially achieved in an arbitrary order." [Sussman, 1973, p.58]. We refer to this as the *linearity assumption.*

Sussman's planner, Hacker, could not solve the blocksworld "Sussman anomaly" problem. This was because Hacker's search space was narrowed by the linearity assumption so as to preclude finding a solution to this particular problem. Other planners, which built totally ordered plans, *could* solve the Sussman anomaly. For instance, Interplan [Tate, 1974] and Warplan [Warren, 1974] could both solve it. It appears that NOAH [Sacerdoti, 1977], the first partially ordered planner, solved the Sussman anomaly through its judicious selection of goals, *not* through its use of partially ordered plans. The Sussman anomaly highlights the goal ordering problem. NOAH's success with the anomaly is a direct result of its approach to goal ordering, but its results don't appear to be general: successful goal ordering strategies seem to have been built in to NOAH's control structure.

2.3 Goal Interactions

Partially ordered plans are often advertized as a "solution" to the linearity assumption [Stefik, 1981, p.134; Barr and Feigenbaum, 1982, p.520]\ This is incorrect. It is simple to show that the logical extreme of partially ordered planning, the totally unordered plan, will only result when the linearity assumption would work anyway. To show this we must make a few harmless assumptions.

We assume that the initial plan provided to the planner has only two operators, *S* and *F* (for Start and Finish). *S* is ordered before F, *S* asserts all of the problem's initial conditions and has no preconditions itself (it defines the initial situation), and *F* has as its preconditions the user-supplied goals of the problem. We will call such a plan ***a typical initial plan.*** **An** ***unordered plan*** is defined to be a partially ordered plan which has one least and one greatest operator under the plan's partial order: all other operators are unordered with respect to one another. A *fault free* plan is defined to be one in which each and every operator precondition is true by the planner's goal achievement procedure. Such a fault free plan is assumed to exist in the space of partial plans defined by the goal achievement procedure. For the remainder of this paper, we assume that our goal achievement procedure is based on the Modal Truth Criterion.

Theorem 1 *// an unordered fault free plan exists then a planner with a control structure that makes the linearity assumption can generate it.*

Proof. Suppose the planner generates an unordered fault free plan. This plan starts with 5 and finishes with *F.* Let the preconditions of *F* be pi,p2, • • *,Pn-* Each of the added operators achieves *at least one* of the preconditions of F, or it would not have been added. Assume for now that each operator has been introduced to achieve exactly one *pi.* Label the operator which achieves p, as A_i. The plan will be composed of 5, ordered before each of the A, (the added operators), each of which is in turn ordered before *F.* Operators in a plan are ordered by the planner's goal achievement procedure based on the truth of their preconditions. Since the A, are unordered, their preconditions must be true from the initial operator, 5, and remain true through the occurrence of all other *Aj, j ^ i.* Based on this lack of ordering on operators, a goal achievement ordering over the pj, starting with the typical initial plan, can be induced as follows. Achieve an arbitrary unachieved p, by introducing A, and ordering A, —► *F.* Let the preconditions of A, be *qi,q2,* • • •, *Qm-* Any precondition of A,, say <jfc, may be immediately achieved from 5. None of the other p_7, *j # i* need be considered for achievement. Thus all of the preconditions of A, may be achieved before *all* other preconditions of *F.* Repeat the process; *i.e.* achieve another arbitrary unachieved precondition of *F.* This process finds an ordering on the complete recursive achievement of the pi, and this is exactly what the linearity assumption requires. We started off assuming that each A, achieved exactly one precondition of *F.* Suppose that this is not so: the introduction of one A may achieve more than one of F's preconditions. This doesn't matter: an operator Ai could be introduced to

achieve any subset of F's preconditions. There is still no need to achieve any of the other unachieved preconditions of *F* before finishing with all the preconditions of *Ai.* Therefore, if an unordered fault free plan exists then a planner with a control structure that makes the linearity assumption can generate it. D

In fact, the unordered plan specifies that *all* orderings of goal achievement will work; this is of course a stronger requirement than is issued by the linearity assumption, which requires only that *one* ordering work.

Theorem 2 *If a planner with a control structure that makes the linearity assumption fails to produce a fault free plan then there does not exist an unordered fault free plan.*

Proof If the user-supplied goals cannot be sequentially achieved in an arbitrary order, then there are no operators which achieve these goals which are free from interference according to the planner's goal achievement procedure. This means that the selected operators will not be left unordered by the goal achievement procedure in the plan. Once a goal achievement procedure introduces an order to deal with the goal interaction(s), there is no chance of finding an unordered fault free plan. Orderings are never removed by a goal achievement procedure (as based on the MTC), only added. Therefore if a planner with a control structure that makes the linearity assumption fails to produce a fault free plan then there does not exist an unordered fault free plan. □

This is not a dramatic result. We have not shown that there is no fault free plan, only that there is no fault free plan that is also *unordered.* Many interesting cases lie between totally ordered and unordered plans. The point is simply that partially ordered plans, just like the linearity assumption, require a certain degree of "freedom from interference" among operators. As a result, such plans can hardly be viewed as a means for escaping from the control structure confines of the linearity assumption.

3 Temporal Coherence

Temporal Coherence (TC) is a heuristic which provides goal ordering guidance. It obviates the need for exhaustive breadth-first search, is admissible, and allows a planner to produce a solution in reasonable time. This section motivates, defines and explains the use of TC. A sketch of the proof of TC's admissibility is also given.

3.1 Motivation and Definition

The basic principle of TC is this: *do not work on partial plans which have inconsistent bulk preconditions.* Consider: at any point in its search a planner will have a partially completed plan. The search begins with an initial plan, and each partial plan in the search is produced by the addition of some operator schemas, variable bindings and operator orderings. Added operators often have new preconditions. The search continues until all operator preconditions are true as judged by the planner's goal achievement procedure.

Each precondition which *is* true will either be true by some added operator, or true as a postcondition of the initial operator. Consider those preconditions which must be made true if the partial plan developed so far is to be "executable"; such preconditions are the "bulk" preconditions for the developed plan. They are preconditions in the plan which the planner's goal achievement procedure labels as outstanding goals, together with those which are not goals only because they are true as postconditions of the initial operator.

TC suggests working on those plans whose bulk preconditions describe a physically possible state of the planning domain. By "possible" we mean consistent with certain given physical laws. Suppose that a plan's bulk preconditions do not describe a possible domain state. This would happen only if the operators in the plan were not independent of each other, and required further sequencing to form a valid plan. Plan modifications sanctioned by the planner's goal achievement procedure might well introduce the required orderings. But when a planner has the choice between a plan that *already* contains all required orderings and a plan that must have the orderings added, it makes sense to choose the former. This avoidance of temporary impossibilities is a good search heuristic. In our experience it can lead to significant time savings in plan construction.

The above discussion can be made more precise. Take the planner's goal achievement procedure and a partially ordered plan. Remove the start node and its postconditions, and call the plan which results the *modified plan.* Let *P* be the set of all assertions which occur as pre- or post-conditions in the modified plan. The *primary cut* of the original plan is defined to be a set of assertions C, such that *C* C P, and Vc G C, *c* is not true by the goal achievement procedure in the modified plan. The primary cut of the plan is thus the set of preconditions which are not necessarily true by the goal achievement procedure in the modified plan. This set gives us the bulk preconditions required by the plan for its execution.

We would like a plan's primary cut to be logically consistent. Unfortunately, full consistency is only semi-decidable, meaning that a procedure that tests for consistency may or may not terminate. It doesn't make sense to base a heuristic on such a test, since a possibly non-terminating heuristic is of seriously limited utility. However it is possible to make do with a more limited notion of consistency, a notion that we call *coherence* [Drummond and Currie, 1988]. Specifically, we exploit the fact that assertions in a plan are typically non-negated literals. This means that each assertion in a plan is of the form

$$relation(argi, \quad arg2,..., \quad arg_n)$$

where negation, conjunction and disjunction are not allowed.

We define coherence in terms of domain-specific constraints that give the inviolate laws of the application domain. For the blocks world, five constraints are necessary: 1) Blocks cannot both be clear and under some other block; 2) Blocks cannot be on two different objects; 3) Blocks cannot be under two different objects; 4) Objects are not both blocks and tables; 5) Different objects cannot be on each other.

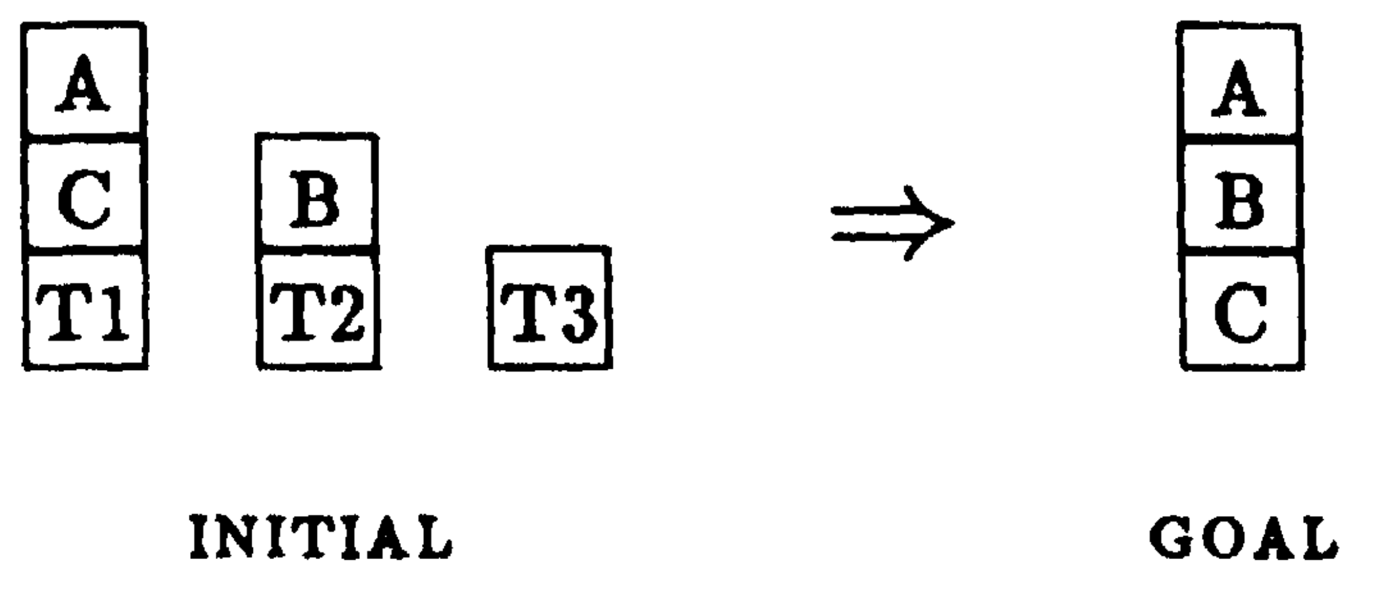

Figure 1: A block stacking problem.

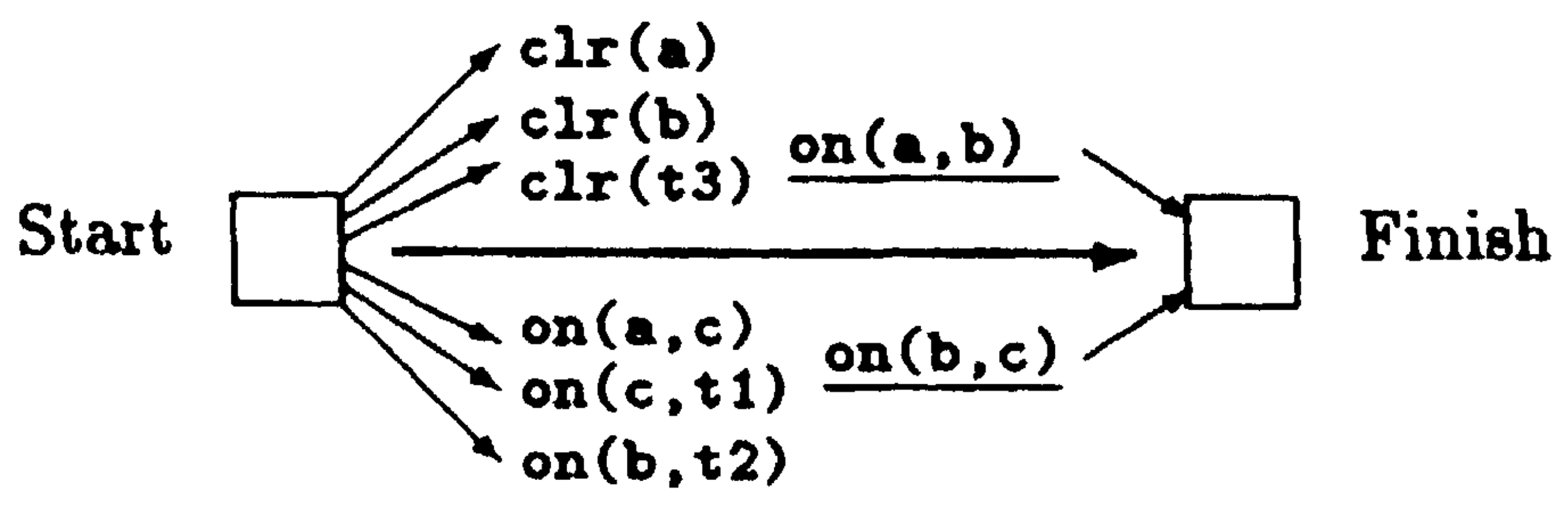

Figure 2: The first partial plan.

Domain constraints must be given to the planner in an appropriate language. We use constraints of the form:

$$\neg(relation_1(\ldots) \wedge \cdots \wedge relation_n(\ldots)).$$

Such a negated conjunction is equivalent to the statement

$$\neg relation_1(\ldots) \vee \cdots \vee \neg relation_n(\ldots).$$

Either form of the constraint indicates that at least one of the specified relations must not hold: if all given relations in a constraint are true the constraint is violated.

Coherence is weaker than consistency, but it works for planners that use only non-negated literals, provided also that the planner does not allow for inference among the assertions scattered throughout a plan. This is typically a safe assumption, since the goal achievement procedures of almost all planners ignore full inference.

Of course, the actual efficiency of the coherence check is determined by the number and size of the domain constraints. For the blocks world example considered there are five constraints, and most constraints contain three relations to check. In many domains most constraints turn out to be binary partitions on alternatives, such as ojff and on, *open* and *closed.* As a result, computing coherence is often simple and cheap in practice.

3.2 A Brief Example

A block stacking problem is given in figure 1. This version of the blocks world only has *blocks* in it: there are no infinite capacity tables. The effectiveness of TC does not depend on this particular version of the blocks world.

We use this problem to briefly illustrate the use of TC. The plan of figure 2 is used to encode the problem. We consider the goal-ordering alternatives open to a hypothetical planner that starts its search at this initial plan. The "Start" operator in the plan is used to assert the problem's initial situation. The "Finish* operator is used to present the problem's conjunctive goal. This plan is contained in the root node of the search space that our hypothetical planner must explore.

First, some simple graphical conventions. Operators are drawn as boxes; operator preconditions are indicated by drawing an arc from the precondition to the operator; preconditions which are also deleted are indicated by scoring across this precondition relation; added conditions are indicated by drawing an arc from an operator to the condition. Preconditions that are outstanding goals (as judged by the planner's goal achievement procedure) are indicated by underlining.

We assume the global availability of an operator schema *mcn>e(X, Z,Y)* for moving a block *X* from some initial location *Z* to some final location *Y.* As suggested by this schema, variables are denoted by upper case letters and constants are denoted by lower case letters. Please note that TC *does not depend* on this somewhat trivial formulation of operators and their effects. This formulation is used only to facilitate concise explanation. Since all the objects we consider are blocks we will not always write the predicate *block(X),* but rather assume it for all objects *X* that we deal with.

The first plan has the goals $on(a,b)$ and *on(b,c).* There is no other way to achieve these goals except through the introduction of a new operator, derived from the general schema. Suppose we work on the goal on(6, c) first. Binding an instance of the schema and inserting it gives us the plan in figure 3. (To be compact, we have not drawn the Start and Finish operators in this plan, but of course in reality, they would be there.) Now apply TC to this plan. The primary cut of this plan is $\{on(a,b), on(b,Z1), clr(b), clr(c)\}$. This cut is not coherent since it violates the domain constraint which says that blocks cannot be both clear and support some other block. The plan is not temporally coherent, so it is terminated, and the search continues from the last choice point. This was the choice of the achievement of *on(b, c)* over $on(a,b)$.

How are we to interpret this goal ordering advice? Temporal coherence discarded the plan of figure 3, where the assertions oni(a,b) and *clr(b)* were both in the primary cut. This plan did not order the two assertions in time; because of this, it is possible that $on(a,b)$ could be achieved temporally before the action which achieves on(6, *c).* But since *on(a. b)* is a final goal, it must be true at the end of the plan. If the action *move(b, Zl,c)* occurs after whatever action is selected to achieve $on(a,b)$, then the truth of $on(a,b)$ is in jeopardy. Thus, the goals have been attacked in the wrong order. The last action in the plan *must* be the one which achieves $on(a,b)$, and not *on(b,c).*

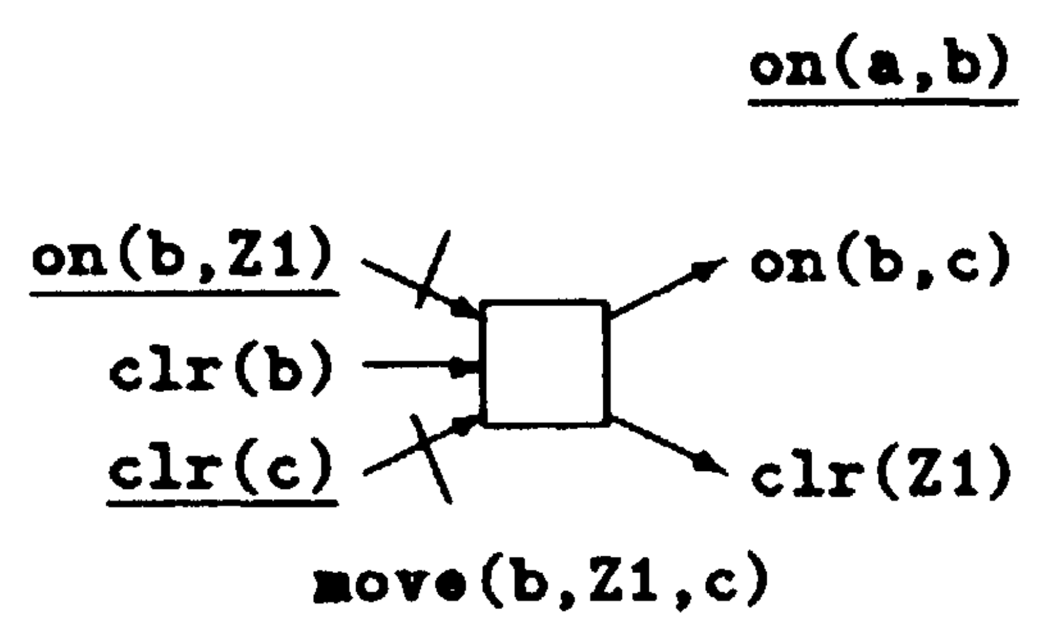

Figure 3: Achieving on(b,c).

3.3 When Will Temporal Coherence Work?

Each plan suggested by a planner's goal achievement procedure is examined by TC: if the plan's primary cut is not coherent, we terminate the search at the state containing the plan. If the primary cut *is* coherent, the plan state is retained. The idea is to ignore parts of the partial plan search space defined by the planner's goal achievement procedure. We cannot claim that the parts of the space so ignored *do not* contain plans which are solutions. All we do claim is that the subspace which remains after the application of TC *does* contain a solution, provided that certain restrictions are in force. Necessary restrictions apply to the planner's operator schemas and initial plan. When these restrictions are enforced, if a solution plan exists in the space defined by the planner's goal achievement procedure then a temporally coherent path to it also exists.

As per section 2.3 we assume that the search begins with a typical initial plan (such as the plan of figure 2). We insist that the set of assertions describing the initial situation given by *S* be coherent with respect to the given set of domain constraints. Likewise, the goals imposed by *F* must be coherent with respect to the given constraints. (The specific constraints will be implicit in the rest of the discussion.) We assume that a fault free plan exists in the space of partial plans. A *temporally coherent path* to this plan is a sequence of plan transformations permitted by the planner's goal achievement procedure, such that each partial plan derived under the transforms in the sequence is temporally coherent.

We require our operator schemata to be *sound* as defined by Lifschitz [1986, definition C]. This ensures that if a state description makes sense, i.e. is coherent, then the application of an operator preserves this coherence.

Theorem 3 *If a fault free plan exists in the search space of partial plans defined by the planner's goal achievement procedure then at least one temporally coherent path to it also exists.*

Proof Sketch (see Drumrnond and Currie [1988] for details). First, we need to translate the fault free plan into a state-space structure called a *projection.* We can construct a projection by starting with the "empty state", containing no assertions, and apply each operator in the plan in an order consistent with that required by the plan's operator ordering. We can apply an operator if and only if all of the operators which immediately precede it have already been applied. *S* is applicable immediately, since it has no predecessors. Typically, applying an operator means acting on the assertions specified in its delete- and add-lists. To derive a successor state, we first delete all the formulae in the operator's delete-list, and then add all the formulae in its add-list. This generates a successor state in the projection which describes the state of the environment following successful execution of the action denoted by the operator.

The initial state, that state produced by the application of 5, is coherent by assumption. By the operator soundness requirement, each successor state in the projection will also be coherent. We can use this projection as a demonstration of the existence of (at least one) temporally coherent path for plan construction. The final successful plan must be built up step by step, with the addition and ordering of new operators. We can add operators in any order licensed by the order in which the operator names appear in reverse paths through the projection; each partial plan we produce in this way will be temporally coherent. Consider: starting with the typical initial plan, we can add any *one* of the final operators which is indicated by the corresponding instance in the projection. We can do this recursively until all operators are added to the initial plan (i.e. recurse until we arrive at the application of the initial operator, 5, in the projection).

The initiating state for an operator in the projection (the state on which the operator depends for the truth of its preconditions) is coherent by the above argument. The primary cut of the plan which has been constructed by the addition of operators up to and including the operator will be a subset of this state. It must be, or the remainder of the plan would not be applicable, and the projection could not have been constructed: recall that a plan's primary cut is its set of bulk preconditions. Since subsets of coherent sets are coherent, the primary cut of each partial plan will also be coherent.

This means that *at least* one temporally coherent path of construction exists to the final fault free plan. More than one path may exist. The number of paths is determined by the number of different reverse routes through the plan's projection. The number of paths increases with the factorial of the number of unordered operators in the fault free plan. Therefore if a fault free plan exists in the search space of partial plans defined by the planner's goal achievement procedure then at least one temporally coherent path to it also exists. □

3.4 The History of the Idea

Warren [1974] was the first to suggest the use of domain constraint information, although he only applied it to totally ordered plan structures, and used it to guide the insertion of operators into developing plans. The idea of applying a slice-wise consistency analysis to partially ordered plan structures was first presented by Allen and Koomen [1983]. In their formulation however, violated domain constraints were used to suggest alternative orderings of already planned operators. But this is not the way that NonLin derivative planning systems work [Tate, 1977]. Chapman's [1987] formalization of NonLin is the base that we take for the application of domain constraints. TC is the application of such constraints to partial plans; plans which only *might* be on route to a solution are rejected in the interest of efficiency. Other uses of the consistency argument have been advanced; see for instance, Ginsberg and Smith [1987a, 1987b], and Drummond [1986a, 1986b]. TC has been implemented and tested in the O-Plan system [Currie and Tate, 1985]. Initial results in simple domains are promising, and work is underway to apply TC to larger domains where the problem of search control is more acute.

4 Conclusions

Partially ordered plans have not solved the goal ordering problem. The goal ordering problem is tackled by heuristics such as the linearity assumption which prevents recursive subgoals from being interleaved with goals at a higher level. Of course this introduces incompleteness, but it *is* a way of reducing the search. This paper has presented TC, an admissible heuristic for the goal ordering problem. TC works by avoiding plans with internal ordering problems. These internal problems are detected by analyzing the plan's bulk preconditions. Inconsistent bulk preconditions indicate that the plan will require corrective ordering work in the future. We have shown that such work should always, *and can always,* be avoided.

Acknowledgements

Austin Tate, Amy Lansky and John Bresina have influenced the development of the ideas presented here. Peter Friedland, Peter Cheeseman, Hamid Berenji, Kevin Thompson and Lisa Leonard provided useful comments on a draft. To all, our thanks.

References

[Allen and Koomen, 1983] Allen, J. and Koomen, J. Planning Using a Temporal World Model. In *Proceedings of the Eighth Joint Conference on Artificial Intelligence,* pages 741-747, Karlsruhe, West Germany, 1983. International Joint Committee on Artificial Intelligence.

[Barr and Feigenbaum, 1982] Barr, A. and Feigenbaum, E. *The Handbook of Artificial Intelligence,* Volume III. William Kaufmann Inc.

[Chapman, 1985] Chapman, D. Nonlinear Planning: a Rigorous Reconstruction. In *Proceedings of the Ninth Joint Conference on Artificial Intelligence,* pages 1022-1024, Los Angeles, CA, 1985. International Joint Committee on Artificial Intelligence.

[Currie and Tate, 1985] Currie, K. and Tate, A. O-Plan: Control in the Open Planning Architecture. In *Proceedings of the British Computer Society Expert Systems '85,* Warwick, U.K., pages 225-240, 1985. Cambridge University Press.

[Dean and Boddy, 1988] Dean, T., and Boddy, M. Reasoning About Partially Ordered Events. *Artificial Intelligence,* Vol. 36, pages 375-399, 1988.

[Drummond, 1986a] Drummond, M. A Representation of Action and Belief for Automatic Planning Systems. In *Proceedings of Reasoning About Actions and Plans,* Timberline, Oregon, USA. Pages 189-211. 1986. Morgan Kauffman.

[Drummond, 1986b] Drummond, M. Plan Nets: a Formal Representation of Action and Belief for Automatic Planning Systems. Ph.D dissertation, 1986, Dept. of AI, University of Edinburgh.

[Drummond and Currie, 1988] Drummond, M. and Currie, K. Exploiting Temporal Coherence in Nonlinear Plan Construction. *Computational Intelligence,* Vol 4(3), August, 1988.

[Ginsberg and Smith, 1987a] Ginsberg, M. and Smith, D. Reasoning About Action I: A Possible Worlds Approach. Stanford Logic Group Report Logic-87-9, Stanford University, 1987.

[Ginsberg and Smith, 1987b] Ginsberg, M. and Smith, D. Reasoning About Action II: The Qualification Problem. Stanford Logic Group Report Logic-87-10, Stanford University, 1987.

[Lifschitz, 1986] Lifschitz, V. On the Semantics of STRIPS. In *Proceedings of Reasoning About Actions and Plans,* Timberline, Oregon, USA. Pages 1-9. 1986. Morgan Kauffman.

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

[Stefik, 1981] Stefik, M. Planning With Constraints. *Artificial Intelligence,* Vol. 16, pages 111-140, 1981.

[Sussman, 1973] Sussman, G. A Computational Model of Skill Acquisition. MIT AI Technical Report 297, 1973.

[Tate, 1974] Tate, A. Interplan: A Plan Generation System Which Can Deal With Interactions Between Goals. University of Edinburgh, Machine Intelligence Research Unit Memorandum MIP-R-109, 1974.

[Tate, 1977] Tate, A. Generating Project Networks. In *Proceedings of the Fifth Joint Conference on Artificial Intelligence,* pages 888-893, Boston, MA, 1977. International Joint Committee on Artificial Intelligence.

[Warren, 1974] Warren, D. Warplan: a System for Generating Plans. Memo 76, Computational Logic Dept., School of Artificial Intelligence, University of Edinburgh, 1974.

[Wilkins, 1984] Wilkins, D. Domain Independent Planning: Representation and Plan Generation. *Artificial Intelligence,* Vol. 22, pages 269-301, 1984.