Generalizing Nonlinear Planning to Handle Complex Goals and Actions with Context-Dependent Effects

Edwin P.D. Pednault
AT&T Bell Laboratories
Crawfords Corner Road
Holmdel, NJ 07733

Abstract

This paper presents a general, mathematically rigorous approach to nonlinear planning that handles both complex goals and actions with context-dependent effects. A goal can be any arbitrary well-formed formula containing conjunctions, disjunctions, negations, and quantifiers. Actions are likewise not constrained and can have an unrestricted number of complex, situation-dependent effects. The approach presented here can thus be used to solve a wider range of problems than previous approaches to nonlinear planning. The approach is based on previous work by the author on linear planning. The same mathematical framework is used with the results extended to nonlinear plans.

1. Introduction

Efforts have been made over the past several years to place automatic planning on a firm mathematical foundation. These efforts have focused on two types of planning techniques: *linear planning,* in which plans are represented as (linear) sequences of actions [Pednault 1985, 1986, 1987, 1988, 1989], and *nonlinear planning,* in which partially-ordered networks (i.e., directed acyclic graphs) are used to represent plans [Chapman 1985, 1987; Christensen 1990a, 1990b; McAllester and Rosenblitt 1991; Yang and Tenenberg 1990],

Work on the theory of linear planning has resulted in a general planning method capable of handling both arbitrarily complex goals involving conjunctions, disjunctions, negations, quantifiers, etc., as well as arbitrarily complex actions whose effects can change according to the situations in which they are performed [Pednault 1986, 1988]. A planner that incorporates some of these results has recently been implemented by McDermott [McDermott 1991]. McDermott's work provides independent confirmation of the theory and it begins to address some of the implementation issues that the theory entails.

In contrast, work on the theory of nonlinear planning has focused on a simplified version of the STRIPS framework [Fikes and Nilsson 1971] in which goals are constrained to be conjunctions of literals, and actions are constrained to those representable by means of simple add and delete lists. Several planning systems have been implemented that incorporate these results [Chapman 1985, 1987; Christensen 1990b; McAllester and Rosenblitt 1991; Yang and Tenenberg 1990]. However, because of the limited representation employed for goals and actions, the range of problems that these systems can solve is quite restricted. This is particularly true in comparison to the range of problems that can be solved by the linear planning techniques cited above. Clearly, a gap exists in the generality of the formal results obtained to date for linear and nonlinear planning methodologies.

The purpose of this paper is to bridge this gap by presenting a general approach to nonlinear planning capable of handling arbitrarily complex goals and actions. This approach is an extension of the linear planning methods previously developed by the author. The extensions preserve the range of problems that can be solved while taking advantage of the ability of partial orders to represent several possible sequences of actions simultaneously. This ability can potentially make the nonlinear approach more attractive than the linear method upon which it is based.

2. Mathematical Considerations

In all mathematical approaches to planning that have been developed to date, planning algorithms are constructed from theorems that define the kinds of actions a plan must contain in order to achieve one's goals. Chapman and others calls these theorems *truth criteria,* I prefer the term *causality theorems* because these theorems establish causal connections between the goals to be achieved, and the actions and subgoals that must appear in the plan.

General causality theorems exist that can be used as the basis for constructing linear planners [Pednault 1985,1986, 1988]. They will likewise be used in this paper for the purpose of nonlinear planning. The main causality theorem can be stated informally as follows:

Theorem 1 *(Main Causality Theorem):* A condition O will be true at a point *p* during the execution of a plan if and only if one of the following holds:

(1) An action *a* is executed prior to point *p* such that *a* causes O to become true and O remains true thereafter until at least point *p.*

(2) O is true in the initial state and remains true until at least point p.

Formal statements and proofs of this theorem have been previously reported [Pednault 1985, 1986, 1988]. The important thing to note for our current purposes is that Theorem 1 is stated with respect the to sequence in which the actions of a plan are executed. A linear plan defines a single execution sequence; a nonlinear plan defines several possible sequences. Thus, in applying this theorem to nonlinear planning, we will in effect be applying it to each of the execution sequences defined by a nonlinear plan.

To apply Theorem 1, one must have the ability to assert that a particular action causes a particular goal to become true, and to assert that a particular action preserves the truth of a particular goal once it has been achieved. In many planners, such as the one considered by Chapman [Chapman 1985, 1987], these kinds of assertions are quite straightforward to make because the representations employed limit themselves to actions with context-independent effects. Such actions will either cause a particular goal to become true in all circumstances, cause the goal to become false in all circumstances, or leave the truth value of the goal unaltered in all circumstances. To assert that a particular action achieves a particular goal, one need only verify that the action does in fact achieve the goal and eliminate from consideration any action that fails to do so. To assert that a particular goal is to be preserved between two points in a plan, one need only verify that all intervening action either achieve the goal or leave its truth value unaltered, and eliminate from consideration any plan (or execution sequence of a nonlinear plan) that violates this protection constraint Thus, except for keeping track of the intervals during which certain goals are to be preserved, the other assertions that particular actions must achieve particular goals or preserve particular goals need not be represented explicitly in a plan. They need only be embodied implicitly in the procedures by which a plan is constructed. Note that this is only true for actions with context-independent effects.

In the general case, the effect of an action can depend on the situation in which it is performed. Asserting that such an action achieves or preserves a goal then amounts to asserting that the action is performed in a situation in which is has this effect These kinds of assertions cannot be made implicitly; they must be represented explicitly. In the linear planning approach previously developed by the author, these assertions are made by introducing additional preconditions to the actions in a plan. These additional preconditions, which are collectively called *secondary preconditions,* define the conditions under which the actions have their desired effects. Two types of secondary preconditions are used: *causation preconditions,* which define the contexts in which actions achieve desired goals, and *preservation preconditions,* which define the contexts in which actions preserve the truth of goals. Causation and preservation preconditions can be defined in terms of regression operators [Waldinger 1977] in a very general way [Pednault 1986, 1988]. Remarkably, a causality theorem exists that is essentially equivalent to Theorem 1 except that it is expressed in terms of achieving the appropriate secondary preconditions. In the statement of this theorem which follows, Ejj! is used to denote the causation precondition for action *a* to achieve cp>, while Π_φ^a denotes the preservation precondition for action *a* to preserve O.

Theorem *2 (Causality Theorem for Secondary Preconditions):* A condition o will be true at a point *p* during the execution of a plan if and only if one of the following holds:

(1) An action *a* is executed prior to point *p* such that

(a Σ_φ^a s true immediately before executing *a.*

(b) Π_φ^b is true immediately before the execution of each action *b* between *a* and point *p.*

(2) p is true in the initial state and Π_φ^a is true immediately before the execution of each action *a* prior to point *p.*

An example of causation and preservation preconditions are the so-called "codesignation constraints" (i.e., equality constraints) employed in Chapman's TWEAK program. When inserting a new action or using an existing action to achieve a goal, TWEAK introduces codesignation constraints as needed to ensure that the goal unifies with one of the formulas in the add list of the action. These equality constraints satisfy the definition of causation preconditions [Pednault 1986,1988] and are effectively used as such in TWEAK. TWEAK also has the option of introducing noncodesignation constraints (inequalities) to prevent a goal from unifying with one of the formulas in the delete list of a potential "clobbering" action. These inequality constraints satisfy the definition of preservation preconditions and are clearly used as such in TWEAK. Causation and preservation preconditions, however, are not limited to equality formulas. In general, they can be arbitrarily complex formulas whose complexity is a function *of the degree to* which an action's effects are context dependent.

In the author's previous work, Theorem 2 plays the same role in constructing a linear planner that Chapman's "modal truth criterion" [Chapman 1985, 1987] plays in constructing a nonlinear *planner. Theorem* 2, however, applies to all actions that can be represented in terms of state transitions, whereas the modal truth criterion is limited to actions representable in a limited STRIPS framework. In addition, goals can be arbitrarily complex formulas involving conjunctions, disjunctions, negations, and quantifiers—O in Theorems 1 and 2 is not restricted to atomic formulas and their negations as is the case for the modal truth criterion.

To synthesize linear plans, Theorem 2 can be converted into a nondeterministic procedure in much the same way that Horn clauses are converted into procedures in PROLOG. The procedure is nondeterministic in the sense that the clauses of Theorem 2 define alternate ways of achieving a goal, but it is impossible to determine beforehand which will lead to a solution. Search is therefore required to explore the possibilities.

In the first clause of Theorem 2, action *a* might already appear in the plan constructed thus far, or it might have to be added. Hence, this clause defines two alternatives: one in which a new action is inserted into the plan to achieve a goal, the other in which an existing action is used for this purpose. The second clause defines only one alternative, which is to prevent the goal from becoming false if it is true in the initial state. In each case, the appropriate secondary preconditions are introduced as subgoals to actions as per Theorem 2. These three alternative ways of achieving a goal in a linear plan are discussed in detail elsewhere [Pednault 1986, 1988]. We will now consider how these alternatives generalize when constructing nonlinear plans.

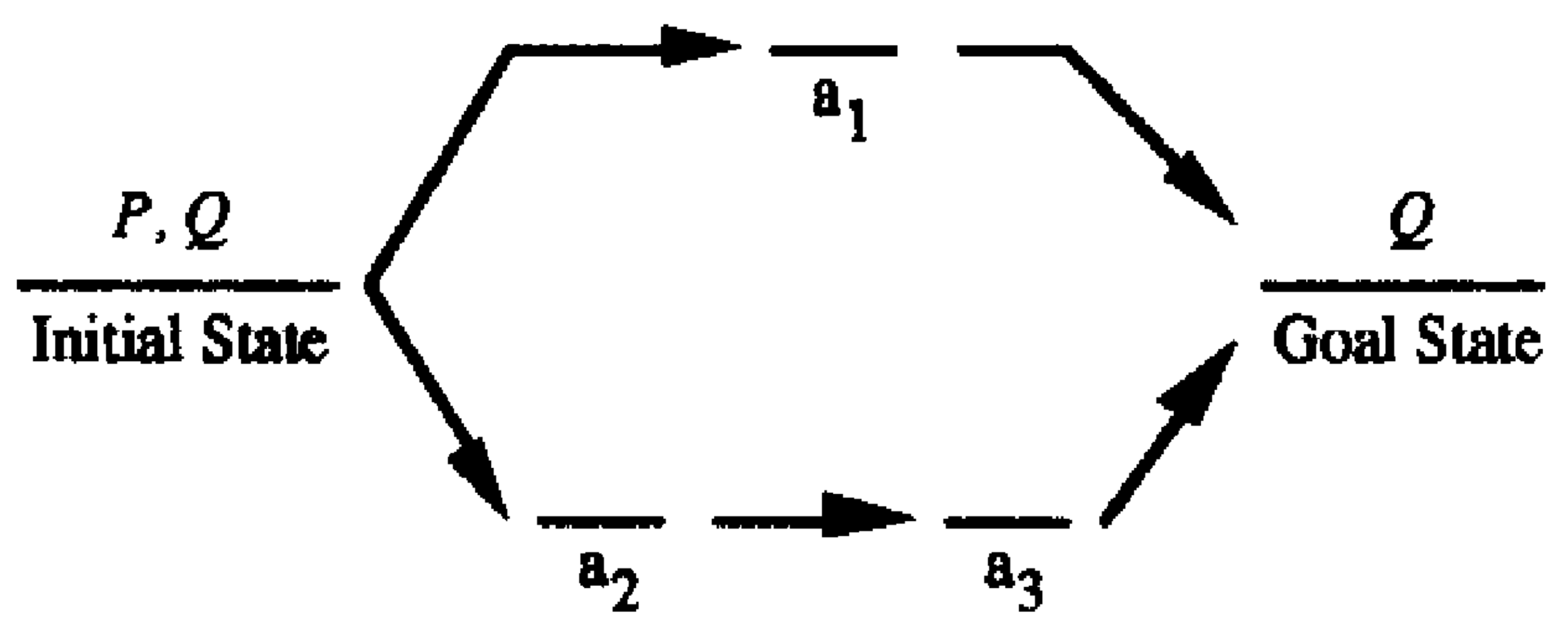

Figure 1: A Sample Nonlinear Plan.

3. Using Theorem 2 in Nonlinear Planning

The first thing to note about Theorem 2 is that it cannot be applied to nonlinear planning in the same manner it is applied to linear planning. The reason is that when dealing with actions Lhat have context-dependent effects, it is possible for a goal to be achieved by a certain action in one execution sequence of a nonlinear plan, a different action in another sequence, and to be preserved by all actions in a third sequence. Therefore, it is impossible in the general case for a single set of secondary preconditions to cover all possible execution sequences of a nonlinear plans. A single set of subgoals, however, is the sort of thing one would want when building a planner.

To illustrate why a single set of secondary preconditions is not sufficient in the general case, consider the three actions, a_1 a_2, and a_3, shown in Figure 1. Action a_1 causes Q to become true if it is not already true, and it causes P to become false if it is not already false. Actions a_2 and a_3 both have the same effect, which is to toggle the truth value of Q whenever P holds. In other words, a_2 and a_3 cause Q to become false if both P and Q are presently true, and they cause Q to become true if P is true and Q is false. The plan in Figure 1 defines three possible orders of execution: $a_1a_2a_3$ $a_2a_1a_3$, and $a_2a_3a_1$ Each each execution sequence results in Q being true in the goal state. However, in the first sequence, Q is true by virtue of the fact that it never becomes false (i.e., all three actions preserve the truth of 0; in the second sequence a_1 causes Q to become true after a_2 makes it false; and in the third execution sequence a_2 is the action that finally achieves Q. Because Q is achieved in different ways in each execution sequence, a different set of secondary preconditions must be satisfied in each case as dictated by Theorem 2- The secondary preconditions for each execution sequence are shown in Figure 2,

Figure 2 also illustrates why Chapman's modal truth criterion does not hold for actions with context-dependent effects. His criterion requires that, for every action in the plan capable of negating a goal before the point at which the goal must be true, there is an intervening action in the plan that re-achieves the goal. However, in the first two orderings of the plan shown in Figure 2 (i.e., $a_1a_2a_3$ and $a_2a_1a_3$), action a_3 is capable of negating Q yet there are no actions that follow a_3 that re-achieve Q. Nevertheless, Q is true in the final state. The reason is that while a_3 is capable of negating Q, it can only do so when P is true at the time the action is performed. Such context-dependent behavior is not accounted for in Chapman's criterion.

We are now faced with a dilemma. On the one hand, when constructing a nonlinear planner, it is desirable to introduce a single set of secondary preconditions that apply to every execution sequence defined by a plan. On the other hand, Theorem 2 tells us that in general a different set of secondary preconditions will be needed for different execution sequences. We could try to maintain these different sets explicitly for each execution sequence; however, we would then be doing a rather convoluted form of linear planning. By representing each execution sequence explicitly, we would defeat the purpose of nonlinear representations altogether, which is to represent multiple execution sequences implicitly. How then can we use Theorem 2 to construct nonlinear plans in the general case?

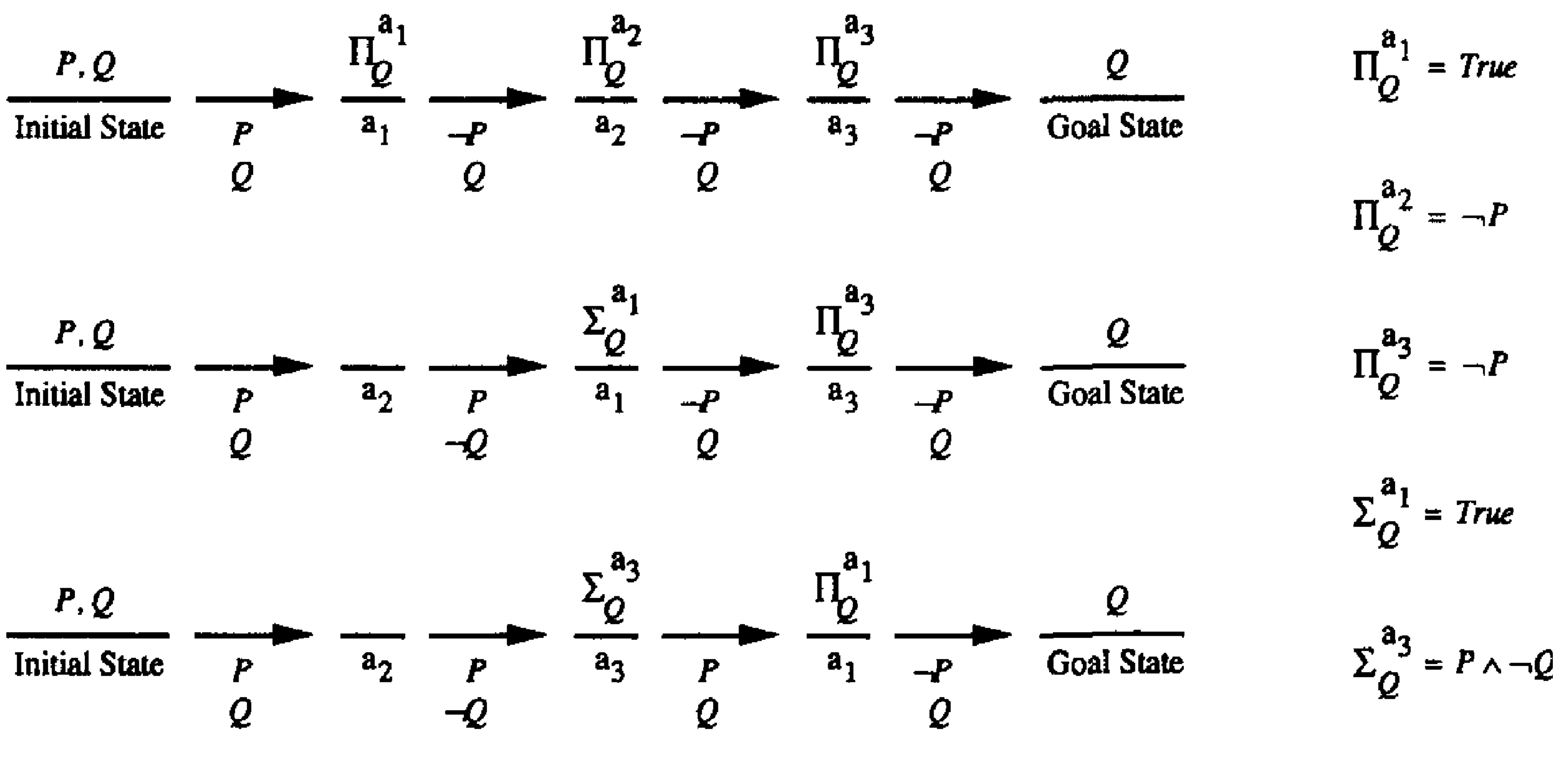

Figure 2: Q is achieved in different ways in different execution sequences

The answer is to recognize that the main role of nonlinear representations is to reduce the size of the search space by allowing several execution sequences to be represented and manipulated simultaneously. In generalizing nonlinear planning, we need only maintain the partial order as long as this is advantageous. At any point, we are at liberty to decompose a nonlinear plan into several alternate plans by introducing additional ordering constraints. Specifically, if actions *a* and *b* are mutually unordered in a nonlinear plan (i.e., if the plan does not specify which action is to be executed first), then we are at liberty to force an ordering. However, when doing so, we must consider all possibilities (in this case, *a* before *b* and *b* before a). If all possibilities are not considered, a solution to the planning problem could inadvertently be eliminated from the search space through this oversight. The decision to force an ordering among two or more actions therefore gives rise to a set of alternate plans that covers the *set* of possible execution sequences defined by the original plan. To maintain completeness, each of these alternate plans must be introduced into the search space as a possible decomposition of the original plan.

This principle of decomposing a plan by forcing an ordering among actions can be used to resolve the dilemma of needing multiple sets of secondary preconditions for a nonlinear plan. If a decomposition is chosen so that for each resulting plan the same secondary preconditions apply to every execution sequence of that plans, then the problem of having multiple sets of secondary preconditions would disappear. Each nonlinear plan in the decomposition would have a different set of secondary preconditions; however, the secondary preconditions of each plan would apply to all the execution sequences of that plan. By taking this approach, Theorem 2 can be applied in much the same manner as for linear planning, except that nonlinear plans might have to be decomposed in the process.

In the following discussion, the necessary plan decompositions are described procedurally. A mathematical analysis of these decompositions and their proofs of correctness are presented in a forthcoming paper [Pednault forthcoming]. They are not present presented here due to space limitations.

Consider what must be done to carry out the fewest number of decompositions necessary to satisfy the constraints described above. Three cases need to be considered, one for each of the three ways of achieving a goal implicit in Theorem 2 (i.e, protecting the goal from the initial state, inserting a new action that achieves it, and using an existing action for this purpose). As will soon become apparent, decompositions will only need to be performed as a result of protecting a goal from one point in a plan to another. Therefore, it is useful to begin the analysis by considering the case of achieving a goal by protecting it from the initial state.

According to Theorem 2, if a goal is protected from the initial state, the appropriate preservation preconditions must hold for every action that is between the initial state and the point at which the goal is to be achieved when the plan is executed. Thus, if we were to achieve a goal by protecting it from the initial state, we would certainly have to introduce preservation preconditions as subgoals for every action necessarily constrained to lie between these two points in the plan constructed thus far. However, what about the actions that lie between the initial state and the point at which the goal is to be achieved in some execution sequences but not in others? Clearly, some of these actions might have no effect on the goal, in which case their preservation preconditions would be the formula *TRUE.* Since *TRUE* is always satisfied, the plan need not be altered on the basis of these actions. Other actions, however, might always have the effect of negating the goal, in which case their preservation preconditions would be the formula *FALSE.* It is inconsistent for such actions to lie between the initial state and the point at which the goal is to be achieved given that we are trying to prevent the goal from becoming false in this interval. Therefore, ordering constraints must be introduced into the plan to assert that actions whose preservation preconditions are *FALSE* necessarily follow the point at which the goal is to be achieved. The introduction of these ordering constraints is essentially a special case of the *linking out* procedure first introduced in Tate's NONLIN program [Tate 197[7]].

The final group of actions that could potentially lie between the initial state and the point at which the goal is to be achieved are those that are capable of preserving the goal, but only in certain circumstances. These circumstances are described by the preservation preconditions for those actions. If the preservation preconditions are not satisfied, the actions could potentially negate the goal. Therefore, in accordance with Theorem 2, each preservation precondition must be achieved as a subgoal to the corresponding action in every execution sequence in which that action lies between the initial state and the point at which the goal is to be achieved. In all other execution sequences, the action follows the point at which the goal is to be achieved. Its preservation precondition is superfluous in this case and need not be achieved. Therefore, to avoid multiple sets of secondary preconditions, the plan constructed thus far must be decomposed into two or more alternate plans that *effectively* group the various execution sequences according to whether or not preservation preconditions must be introduced for the various actions. Note that each alternative then becomes part of the search space, since the alternate plans define different sets of potential solutions.

The decomposition can be accomplished as follows. If only one action is affected, two alternative plans are produced. In one plan, the action is linked out by introducing an ordering constraint to assert that the action necessarily follows the point at which the goal is to be achieved. In the other plan, the appropriate preservation precondition is introduced as a subgoal to the action, and the action is *linked in* [Tate I977] by introducing an ordering constraint to assert that the action necessarily precedes the point at which the goal is to be achieved. The first plan defines all execution sequences of the original plan in which the preservation precondition need not be introduced. The second plan defines the remaining sequences in which the preservation precondition must be introduced. The combined execution sequences of the two plans are thus the same as the execution sequences of the original plan. If several actions potentially require preservation preconditions, this decomposition process must be repeated for each of these actions in turn, with each successive decomposition applied to the results of the preceding one. The net result is several alternate plans that then become part of the search space.

A simitar process of introducing preservation preconditions and ordering constraints must be followed when achieving a goal by introducing a new action, or when using an existing action for this purpose. In general, when protecting a goal from one point in a plan to another, preservation preconditions must be introduced as subgoals to all actions that necessarily lie between these two points. Every action that could possibly lie in this interval whose preservation precondition reduces to the formula *FALSE* must be excluded from the interval by introducing ordering constraints so that the action either necessarily follows the interval or necessarily precedes the interval. This is Tate's general linking out procedure [Tate 1977] . Note that all possible linking-out combinations must be considered for all actions that are affected. In addition, every action that could possibly lie in the interval but whose preservation precondition reduces to neither *TRUE* nor *FALSE* must either be excluded from the interval in the manner just described, or be constrained to lie within the interval and have the appropriate preservation precondition introduced as a subgoal. As before, actions whose preservation preconditions reduce to the the formula *TRUE* need not be considered in this process.

A second way in which a goal can be achieved is by introducing a new action into the plan that makes the goal true and then protecting the goal up to the point at which it is to be achieved. The insertion process is straightforward enough: Ordering constraints are introduced to assert that the new action necessarily follows the initial state and necessarily precedes the point at which the goal is to be achieved. The appropriate causation precondition is then introduced as a subgoal to the new action (as per Theorem 2), together with the preconditions that are normally introduced by planning programs to ensure the action will be executable. Once the action has been inserted, the goal must be protected from the new action up to the point at which it is to be achieved. This is done as described in the preceding paragraph. Finally, since the action could potentially interfere with existing goals that have previously been protected in the plan, the new action must be evaluated with respect to the existing protections, with further plan decompositions performed to either exclude the new action from a protected interval, or constrain it to lie within the protected interval with the appropriate preservation precondition added as a subgoal.

The final way in which a goal can be achieved is by using an existing action. If the existing action is not already constrained to precede the point at which the goal is to be achieved, then an ordering constraint must be added to the plan to establish this relationship (linking in). Once the ordering constraint has been established, the appropriate causation precondition is added as a subgoal to the existing action and the goal is protected from the existing action to the point it is to be achieved in the manner described previously. Unlike the case in which a new action is introduced, there is no need to compare the existing action to goals already protected in the plan, since this analysis would have already been performed at an earlier stage in the planning process. The preconditions for the execution of the existing action would likewise have been introduced at a earlier stage.

The three ways of modifying a nonlinear plan to achieve a goal that have just been described can be embodied in a plan transformation rule much as was done in my previous work on linear planning [Pednault 1986]. The resulting rule generalizes the corresponding rule for linear plans and is used to generate a search tree of possible solutions. When implementing a planning system, the nonlinear rule rule can be combined with the other rules discussed in the author's dissertation to take advantage of formal objects and to decompose complex goals into simpler ones. This is discussed in detail in a forthcoming paper [Pednault forthcoming]

4. The Homeowner's Problem

The following example illustrates the approach to nonlinear planning described above. The example is presented informally. The problem, however, can be formalized by representing the actions in my ADL language [Pednault 1985, 1986, 1989]. The appropriate secondary preconditions can men be derived from this representation. This is discussed in detail in [Pednault forthcoming], but not here because of space limitations.

Suppose you have just closed on a house that you agreed to buy "as is." You arrive at your new "pride and joy" to find that the water has been turned off and a number of sizable holes have been punched into the walls. This situation is not to your liking, so you decide to rectify the problem by turning the water on and fixing the walls (Figure 3a). After experimenting with the water main a bit (Figure 3b), you discover that the reason there are holes in the walls is that the plumbing behind the walls leaks profusely. Fixing the walls can be accomplished only if the plumbing is fixed or the water main is kept off. This condition is therefore introduced as a causation precondition to fixing the walls (Figure 3c). Turning the water on after fixing the walls will create new holes unless the plumbing is fixed. This condition is therefore introduced as a preservation precondition to the parallel action of turning the water on (Figure 3c). This is done so as not to negate the protected goal of having the walls fixed.

Realizing that there is no way to avoid fixing the plumbing, you decide to incorporate this step into your plan (Figure 3d). Fixing the plumbing after patching the walls would merely require that new holes be torn open to gain access. A preservation precondition cannot be introduced to avoid this interaction, since in this case the precondition simplifies to the formula *FALSE* (i.e., one cannot avoid having holes in the wall when fixing the plumbing). The only option is to fix the plumbing before patching the walls (Figure 3d). At this point, either of the two execution sequences defined by the plan will achieve the original goals of turning the water on and fixing the walls.

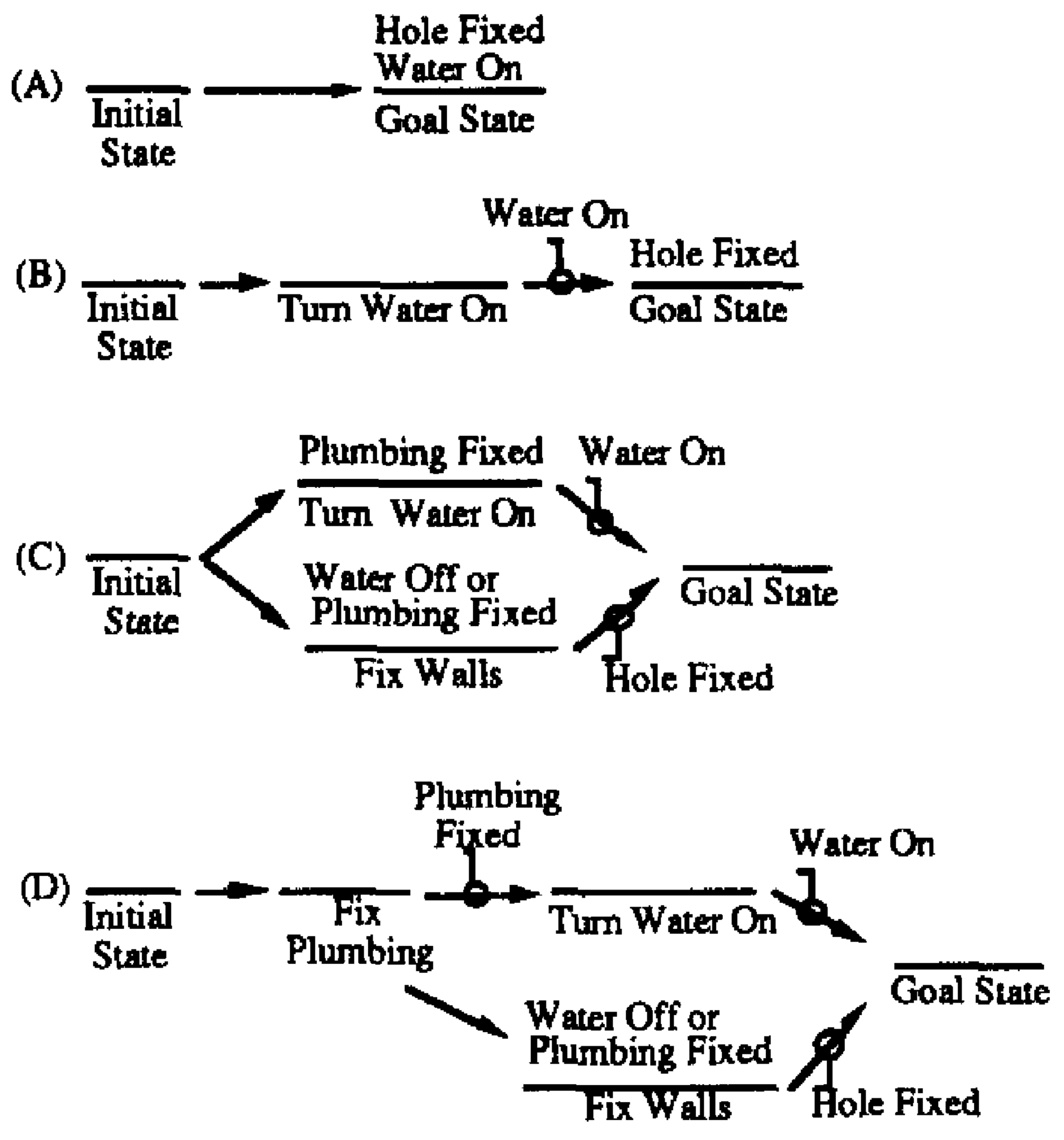

Figure 3: Steps in the Solution of the Homeowner's Problem

This example illustrates the two ways of preventing protected goals from being negated by parallel actions. In Figure 3c, the preservation precondition of fixing the plumbing was introduced as a subgoat to turning the water on so as not to negate the protected goal of having the walls fixed. This method could not be used with regard to the action of fixing the plumbing in Figure 3d, so this action was excluded from the protected interval by introducing an ordering constraint

5. Discussion and Conclusions

In comparing the nonlinear planning approach presented in this paper to the linear planning approaches previously presented by the author, several things can be observed. The first is that, when viewed at the level of the execution sequences, there is essentially no difference in the way in which the plans are being constructed. Both approaches rely on Theorem 2 in the same way. Both generate the same space of potential solutions at the execution level. The only real difference is that partial orders enable several execution sequences to be represented simultaneously in the nonlinear planning approach. This can potentially have the effect of cutting down the branching factor and, hence, the size of the search space. This last statement may at first seem paradoxical given the kind of plan decompositions that are performed when goals are protected. However, keep in mind that these alternatives also appear in the search space of the linear planner. Thus, in the worst case, the nonlinear planning approach will have no worst a search space than the linear approach.

In the best case, the nonlinear approach should be far superior. In cases where the actions unconditionally preserve the goals on parallel branches of a plan, the preservation preconditions for those action simplify to the formula *TRUE,* Thus, the plan need not be decomposed, and preservation preconditions need not be introduced on parallel branches. The result is a tremendous reduction in the size of the search space. This case typically arises when the actions on parallel branches affect disjoint parts of the world. Nonlinear planning achieves its highest level of efficiency for problems in which this special case occurs.

References

Chapman, D. (1985). *Planning for Conjunctive Goats.* Technical Report 802, M.I.T. AI Lab.

Chapman, D. (1987). "Planning for Conjunctive Goals." *Artificial Intelligence.* 32: 333-377.

Christensen, J. (1990a). *Automatic Abstraction in Planning.* Ph.D. Thesis, Computer Science Department, Stanford University, Stanford, California.

Christensen, J. (1990b). "A Heirarchical Planner that Generates its Own Hierarchies." *Proc. AAAI-90,* Boston, Massachusetts, pp 1004-1009.

Fikes, R. E. and N. J. Nilsson. (1971). "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving." *Artificial Intelligence,* 2: 189-208.

McAllester, D. and D, Rosenblitt. (1991), "Systematic Nonlinear Planning." *Proc. AAAI-93 (submitted),*

McDermott, D. (1991). "Regression Planning." *International Journal of Intelligent Systems.*: (in press).

Pednault, E. P. D, (1985). *Preliminary Report on A Theory of Plan Synthesis.* Technical Report 358, AI center, SRI International, Menlo Park, California.

Pednault, E. P. D. (1986). *Toward a Mathematical Theory of Plan Synthesis,* PHD Thesis, Dept. of Electrical Engineering, Stanford University, Stanford Ca.

Pednault, E. P. D. (1987), "Formulating Multiagent, Dynamic-World Problems in the Classical Planning Framework." *InReasoning About Actions and Plans: Proceedings of the 1986 Workshop,* M. P. Georgeff and A, L. Lansky (ed). Los Altos, Ca., Morgan Kaufmann.

Pednault, E. P. D. (1988). "Synthesizing Plans that Contain Actions with Context-Dependent Effects.' *Computational Intelligence.* 4(4): 356-372.

Pednault, E. P. D, (1989). "ADL: Exploring the Middle Ground Between STRIPS and the Situation Calculus." *Proc. KR'89,* Toronto, Canada, pp 324.

Pednault, E. P. D. (forthcoming). "Generalizing Nonlinear Planning/*

Tate, A, (1977). "Generating Project Networks/[1] *Proc, IJCAI-77,* Massachusetts Institute of Technology, Cambridge, Mass, pp 888-893.

Waldinger, R. (1977). "Achieving Several Goals Simultaneously." *InMachine Intelligence* 5, E. Elcock and D. Michie (ed). Edinburgh, Scotland, Ellis Horwood.

Yang, Q. and J. D. Tenenberg. (1990). "ABTWEAK: Abstracting a Nonlinear, Least Commitment Planner." *Proc. AAAI-90,* Boston, Massachusetts, pp 204-209.