 Rev. October, 1993

Nonlinear STRIPS Planning

The basic planning procedure introduced in the previous section is called a *linear* planner because it builds a linear sequences of operations, i.e., a plan, by sequentially adding steps. In a backward chaining planning the steps are sequentially added to the beginning of a partial plan. In a forward chaning planner the steps are sequentially added to the end of a partial plan. A nonlinear planner builds a partially ordered set of steps rather than a linear sequence. In a nonlinear plan a partial plan be refined by either adding new plan steps or adding new ordering constraints on the steps already in the plan.

Recall that a STRIPS operator specification consists of an operation symbol, a prerequisite list, and add list, and a delete list each of which is a set of proposition symbols. Actually, there is no problem in considering arbitrary proposition expressions provided that we consider only *closed* expressions, i.e., expressions that do not contain variables. For example, in blocks world planning the operator with name `MOVE(A, B, C)`, which should be read as "move block A from block B to block C", has prerequisites `CLEAR(A)`, `ON(A, B)`, and `CLEAR(C)`, delete list `ON(A, B)` and `CLEAR(C)`, and add list `CLEAR(B)` and `ON(A, C)`. Also recall that the free model of set of STRIPS operator specifications is a graph whose nodes are sets of propositions (intuitively, the propositions true at that node). The operators are viewed as mapping between these nodes as defined in the previous chapter. In a STRIPS planning problem we are given a set of operator specifications, an initial set Σ of known propositions, and a set Ω of goal propositions. The objective is to find a plan (a sequence of operations) that leads from the set Σ to a set containing Ω.

Nonlinear planners can also be viewed as exploiting a form of quotienting. Each linear plan (sequence of operations) can be abstracted to a unique nonlinear plan (partial order of operations). This abstraction determines an equivalence relation on linear plans — two linear plans are equivalent if they have the same nonlinear abstraction. A nonlinear planner searches the space of equivalence classes of plans. Since a given equivalence class can contain many linear plans, the space of equivalence classes is smaller than the space of plans.

Nonlinear planners are also sometimes called "least commitment planners". The informal principle of least commitment states that one should make low commitment choices before making high commitment choices. In a nonlinear planer we can avoid making the choice that the first (or last) step in a plan is the particular operation o_i and instead make the "weaker" commitment that o_i appears somewhere in the plan without specifying exactly where.

1 The Nonlinear Abstraction of a Linear Plan

Intuitively, two plans are considered to be equivalent if one can be derived from the other by reordering non-interacting steps. For example, consider a robot that must perform a bunch of tasks in room `A` and a bunch of tasks in room `B`. Each task is formally associated with a certain goal proposition. We want to achieve propositions $P_1, \cdots, P_n, Q_1, \cdots, Q_m$. We are given operators $A_1, \cdots, A_n, B_1, \cdots, B_m$ where A_i achieves P_i but must be done in room `A` and B_i achieves Q_i but must be done in room `B`. More formally, each A_i has the single prerequisite `IN(A)`, adds the proposition P_i, and does not delete any propositions. Each B_i is defined similarly with the prerequisite `IN(B)`. We also have an operator `GO(A)` which has no prerequisites, adds the proposition `IN(A)`, and deletes the proposition `IN(B)`. We also have an analogous operator `GO(B)`. The goal set $\{P_1, \cdots, P_n, Q_1, \cdots, Q_m\}$ can be achieved (without any initial assumptions) by the plan `GO(A)`; $A_1; \cdots; A_n$;`GO(B)`; $B_1; \cdots; B_m$. Clearly, the order of the A_i steps and the order of the B_i steps does not matter as long as all the A_i steps are done in room `A` and all the B_i steps are done in room `B`. This plan should be considered to be equivalent to the plan `GO(A)`; $A_n; \cdots; A_1$;`GO(B)`; $B_m; \cdots; B_1$ which performs the A_i steps and B_i steps in the opposite order. Every (linear) plan that is a solution to a given planning problem can be abstracted to a "nonlinear plan" where the nonlinear plan contains only a partial order on the plan steps. Two linear plans are considered to be equivalent if they are different representations of the same nonlinear plan.

To define the nonlinear plan associated with a given linear plan we must first overcome a minor technical difficulty. In a linear plan we can name the individual plan steps by referring to the first step, the second step, and so on. In a nonlinear plan, however, there may not be any well defined second step. We can not name a step by giving the operator used at that step because several steps may involve the same operator. To provide names for the steps in a nonlinear plan we assume that each step is associated with a distinct symbol called the name of that step.

> **Definition:** A *symbol table* is a mapping from a finite set of step names to operators. Every symbol table is required to contain two distinguished step names called `START` and `FINISH`. `START` is mapped to an operator that has no prerequisites and no delete list but which adds a set of "initial propositions". `FINISH` is mapped to an operator that has a set of prerequisites called "goal formulas" but has an empty add list and delete list.

Note that a symbol table does not impose ordering constraints on the step names. Also note that step names are different from operator names. Two step names, say `STEP-37` and `STEP-52`, may both map to the operator named `MOVE(A, B, C)`. Also note that `STEP-37` does not necessarily precede `STEP-52` — the step names have no significance other than as place holders for steps in a plan.

Consider a (linear) solution to a STRIPS planning problem. Without loss of generality we can assume that every prerequisite of every step in the solution plan is true when that step is executed. If w is a step name, and P is a prerequisite of w, then there must be some "source" for P at w — although the source may be the virtual step `START`. If every prerequisite is satisfied for every operation in the plan, then every prerequisite P of every step has a unique source — the last preceding step that adds P. This includes the prerequisites of the virtual step `FINISH`, i.e., the goal formulas. The notion of source motivates the definition of a causal link.[1]

[1]Tate uses the term "range" rather than causal link.

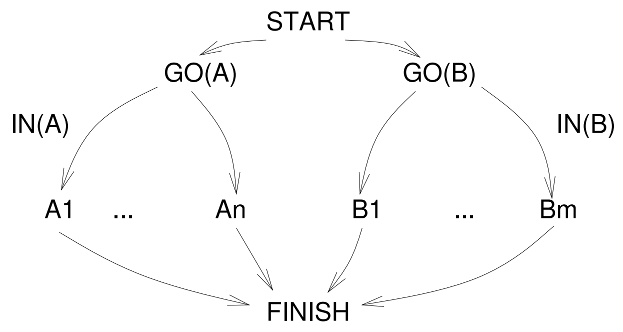

Figure 1: The causal links in $\mathtt{GO(A)}; A_1; \cdots; A_n; \mathtt{GO(B)}; B_1; \cdots; B_m$

.

> **Definition:** A *causal link* is a triple $<s, P, w>$ where P is a proposition symbol, w is a step name that has P as a prerequisite, and s is a step name that has P in its add list. Causal links are written as $s \xrightarrow{P} w$

Causal links indicate the dependencies among steps in a plan. A causal link should be viewed as a constraint. The causal link $s \xrightarrow{P} w$ requires that step s precede step w and that *no step between s and w either adds or deletes P*. Given any linear plan, one can extract the causal links of that plan. For example, consider the plan $\mathtt{GO(A)}; A_1; \cdots; A_n; \mathtt{GO(B)}; B_1; \cdots; B_m$ discussed earlier. This plan has the set of causal links shown in figure 1. Unfortunately, the causal links do not contain all of the relevant ordering constraints in this plan. The partial order information implicit in the casual links does not require each A_i to be done before $\mathtt{GO(B)}$. This example shows the need for ordering information other than causal links.

> **Definition:** A step name v is called a *threat* to a causal link $s \xrightarrow{P} w$ if v is a step name, other than s and w, that either adds or deletes P.
>
> **Definition:** A *safety condition* is an ordering $s < w$ or $w > s$ where s and w are step names.

In the plan shown in figure 1, the step GO(A) is a threat to each casual link of the form $\texttt{GO(B)} \overset{\texttt{IN(B)}}{\rightarrow} B_i$ (since GO(A) deletes the proposition IN(B)), leading to the safety condition $\texttt{GO(A)} < \texttt{GO(B)}$. Similarly, the step GO(B) is a threat to each causal link of the form $\texttt{GO(A)} \overset{\texttt{IN(A)}}{\rightarrow} A_i$, leading to safety conditions of the form $A_i < \texttt{GO(B)}$. These safety conditions, along with the orderings implicit in the causal links, allow the steps to be executed in any order in which GO(A) precedes every A_i, every A_i precedes GO(B), and GO(B) precedes every B_i.

Definition: A *nonlinear plan* consists of a symbol table, a set of casual links, and a set of safety conditions.

Definition: A nonlinear plan is called *complete* if the following conditions hold.

- Every step name appearing in the causal links and safety conditions has an entry in the symbol table.
- If w is a step name in the symbol table, and w has prerequisite P, then the plan contains some causal link of the form $s \overset{P}{\rightarrow} w$.
- If the plan contains a causal link $s \overset{P}{\rightarrow} w$, and the symbol table contains a step name v that is a threat to the causal link $s \overset{P}{\rightarrow} w$, then the plan contains either the safety condition $v < s$ or the safety condition $v > w$.

Consider a (linear) solution to a STRIPS planning problem. This solution can be converted to a nonlinear plan as follows. First, one adds the steps START and FINISH to the two ends of the linear plan where START has no prerequisites and adds given additional set Σ, and FINISH has the given goal set Ω as prerequisites. Second, one constructs a symbol table with one entry for each step of resulting linear plan. Next one adds a safety condition $s \overset{P}{\rightarrow} w$ for each s, P, and w in the linear plan such that s adds P, P is a prerequisite of w, and no step between s and w (in the linear plan) either adds or deletes P. Finally, for each causal link $s \overset{P}{\rightarrow} w$ derived from the linear plan, and each plan step v that is a threat to this link, one adds either the safety conditions $v < s$ or the safety condition $w > v$, depending on which condition holds in the given linear plan. This nonlinear abstraction of a given linear plan is the least complete nonlinear plan corresponding to the given linear plan and is complete up the arbitrary names of the steps used in the symbol table.

The nonlinear abstraction of linear plans determines an equivalence relation on linear plans — two linear plans are considered to be equivalent if they have the same nonlinear abstraction.

2 Topological Sorts and Consistent Nonlinear Plans

The nonlinear planner given in the next section incrementally refines incomplete nonlinear plans. During this refinement process it checks the consistency of the incomplete plan — when an incomplete plan can be shown to be inconsistent no refinement of that plan need be considered. To define consistency we first define the notion of a topological sort of a nonlinear plan.

> **Definition:** A *topological sort* of a nonlinear plan is a linear sequence of all the step names in the symbol table such that the following conditions hold.
>
> - The first step in the sequence is `START`.
> - The last step in the sequence is `FINISH`.
> - For each causal link $s \xrightarrow{P} w$ in the plan, the step s precedes the step w.
> - For each safety condition $u < v$ (or $v > u$) in the plan, the step u precedes the step v.

It is not too difficult to show that any topological sort of a complete nonlinear plan gives a solution to the STRIPS planning problem defined by the `START` and `FINISH` steps. However, if the order constraints implicit in the causal links and safety conditions are inconsistent, i.e., if they contain a loop, then there is no topological sort. This leads to the notion of a consistent nonlinear plan.

> **Definition:** A nonlinear plan (not necessarily complete) is called *order inconsistent* if it has no topological sort.

A transitive closure algorithm can be used to determine if a given nonlinear plan is order inconsistent. A nonlinear plan is order inconsistent if and only if the causal links and safety conditions of the plan define a cycle in the plan steps.

3 A Nonlinear Planning Algorithm

Nonlinear planning was introduced by Sacerdoti [Sacerdoti, 1975]. The procedure presented here is a simplified version of a procedure due to Tate [Tate, 1977], although the formal properties of the procedure, such as soundness completeness and systematicity, were never discussed by Tate and not recognized in the planning literature until recently [McAllester and Rosenblitt, 1991].

Our search procedure is a bounded depth first procedure that can be used with iterative deepening. The procedure takes a nonlinear plan and a cost bound and searches for a completion of the given plan such that the total cost of the steps in the completion is not greater than the given bound. Initially the procedure is called on a nonlinear plan that contains only the `START` and `FINISH` steps corresponding to a given STRIPS planning problem.

The Procedure `FIND-COMPLETION(`β c`)`

1. If the nonlinear plan β is order inconsistent, or the total cost of the steps in β is greater than c, then fail.

2. If the nonlinear plan β is complete then return β.

3. If there is a causal link $s \xrightarrow{P} w$ in β and a threat v to this link in the symbol table such that β does not contain either $v < s$ or $v > w$, then nondeterministically return one of the following.

   (a) FIND-COMPLETION($\beta + (v < s), c$)

   (b) FIND-COMPLETION($\beta + (v > w), c$)

4. There must now exist some step w in the symbol table and some prerequisite P of w such that there is no causal link of the form $s \xrightarrow{P} w$. In this case nondeterministically do one of the following.

   (a) Let r be (nondeterministically) some step name in the symbol table that adds P and return the plan

   FIND-COMPLETION($\beta + r \xrightarrow{P} w, c$).

   (b) Select (nondeterministically) an operator o_i from the allowed set of operations such that o_i adds P. Create a new entry in the symbol table that maps a new step name r to the operator o_i. Then return the plan

   FIND-COMPLETION($\beta + r \xrightarrow{P} w, c$).

We can show that this procedure is sound, complete, and systematic. Soundness in this case means that every value returned by the procedure is a solution to the given planning problem. Soundness follows from the fact that any topological sort of a complete nonlinear plan is a solution. Completeness is the property that the procedure finds a solution whenever a solution exists. Systematicity is the property that a given solution is generated down at most one branch in the search tree (the tree of possible executions of the procedure). systematicity guarantees that one avoids redundant search.

To prove completeness and systematicity we first note that the causal links and safety conditions of the nonlinear plans can be viewed as constraints on linear plans. The procedure maintains the invariant that every step other than the START step can be reached by tracing causal links back from the FINISH step. This means that each step can be named in a way that is independent of the arbitrary names in the symbol table. For example, if the nonlinear plan contains the casual links $s \xrightarrow{P} w$ and $w \xrightarrow{Q}$ FINISH, then the step w can be named as "the casual source of the goal Q" and the step s can be named as "the causal source of the precondition P of the step that is the causal source of the goal Q". The symbol table can be viewed as making assertions about the operations used as casual sources. For example, if the nonlinear plan contains the casual link $w \xrightarrow{Q}$ FINISH and symbol table states that the operation of step w is o_i, these two facts together can be viewed as the statement that casual source of the goal Q must be an instance of the operator o_i. In summary, a nonlinear plan in which each step other than the START step is casually connected to the FINISH step can be viewed as a well defined set of constraints on linear plans. Any topological sort of a complete nonlinear plan is a solution to the constraints implicit in that plan.

Now consider a call to the above planning procedure on a given nonlinear plan β and let α be a linear plan that is consistent with the constraints implicit in β. To show completeness and systematicity it suffices to check that the plan α satisfies exactly one of the constraint sets generated by the nondeterministic branching of the procedure. For example, if the procedure executes step 4 then α must satisfy exactly one of the constraint sets generated in 4a and 4b.

4 Prioritized Prerequisites

One of the earlier ideas in planning is to divide the prerequisites into "high level" and "low level" prerequisites. Intuitively, a high level prerequisite is a high level goal which one should solve before solving low level details. In the AI literature this is sometimes known as planning in a hierarchy of abstraction spaces [Sacerdoti, 1974]. Although so-called hierarchical planning was initially viewed as independent (or orthogonal to) nonlinear planning, it now seems that the two ideas are closely related. One of the main advantages of nonlinear planning is that allows the planner to focus on any prerequisite of any plan step in the partially constructed plan. This means that the planner can focus exclusively on "high level" prerequisites — providing all such prerequisites with causal sources — before considering low level prerequisites. One simple scheme for this is to assign each proposition symbol a numerical priority and when a prerequisite is (deterministically) selected in step 4 of the procedure one selects a prerequisite of maximum priority.

5 Safety and The Modal Truth Condition

The lemma in the previous section states that every complete nonlinear plan is safe in the sense that every topological sort of the plan is a solution. However, the converse of this lemma does not hold — there exist safe partial orders on plan steps which do not correspond to any complete nonlinear plan. For example, consider the partial order on plan steps shown in figure 2.[2] In this plan `FINISH` has prerequisites P, Q and R. The step w_1 adds P and Q while the step w_2 adds P and R. Step w_1 has prerequisite $W1$, which is added by s_1, and w_2 has prerequisite $W2$, which is added by s_2. Unfortunately, both s_1 and s_2 delete P. The partial order on plan steps shown in figure 2 is safe.[3] However, any complete plan (under the definition given here) must specify which of w_1 or w_2 is the casual source of the prerequisite P of the `FINISH` step. This will enforce an explicit ordering of w_1 and w_2.

A necessary and sufficient condition for the safety of a partial order on plan steps was devised by Chapman [Chapman, 1987]. This condition is called "the modal truth condition" and can be stated as follows. A partial order on plan steps is safe if and only if for each step w and prerequisite P of w there exists a step v that necessarily precedes w and adds P, and so that for any step z that can appear between v and w such that z deletes P, there exists a "white night" step z' that is necessarily between z and w such that z' adds P. The above example of a safe partial order on plan steps demonstrates the need for the white night condition in the modal truth criterion.

[2]This example is due to Subbarao Kambhampati.

[3]This example demonstrates the necessity of the white night condition in checking the safety of ground nonlinear plans.

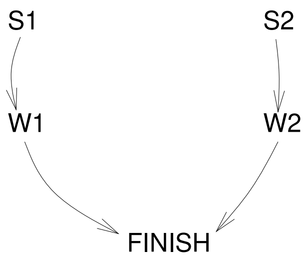

Figure 2: A safe but incomplete nonlinear plan. See the text for an explanation.

It is possible to systematically search all partial orders on step names and find the most abstract (least committed) partial order that is safe. Unfortunately, it seems that any such search process fails to be systematic. The basic problem is that the steps can not be named in a manner that is independent of the arbitrary names in the symbol table. The "different" plans $u; v; w$ and $w; v; u$ may both correspond to the same sequence of operations in the case where u and w represent different instances the same operation.

6 Problems

1. The Sussman anomaly is a blocks world planning where the initial state is given by the propositions

`CLEAR(C),   ON(C,A),  ON(A,PLACE1)`

`CLEAR(B),  ON(B,PLACE2),  CLEAR(PLACE3)`

and the goal consists of the two propositions `ON(A,B)`, `ON(B,C)`. Consider applying the non-linear planning procedure in the notes to the Sussman anomaly. Draw a seach tree for solving the Sussman anomaly where each node in the tree is labeled with a non-linear plan H. The root of your tree should be the initial plan added to `FRINGE` by step 1 of the procedure. A node labeled with plan H should have a child node for each plan H' added to `FRINGE` when H is processed. Exactly what set of children of a given node should have depends on the choice of the unsafe causal link in step 4 or the open prerequisite in step 5. You should make one such choice and only generate the children that correspond to that choice. Show all non-linear plans generated by the procedure that have three or fewer internal nodes.

2. Suppose that we change the definition of a threat to be the following.

> **Definition:** A step name v is called a *threat* to a causal link $s \xrightarrow{P} w$ if v is a step name, other than s and w, that deletes P.

Note that this is different from the definition in the text because, unlike the definition in the text, a step that adds the proposition' of the causal link is not considered to be a threat.

Given an example of a blocks world planning problem and a linear plan that solves it where the linear plan is the topological sort of two different complete nonlinear plans each of which is a possible output of the linear planner. We might say that the nonlinear planner is generating "redundant" plans.

References

[Chapman, 1987] David Chapman. Planning for conjunctive goals. *Artificial Intelligence*, 32:333–377, 1987.

[McAllester and Rosenblitt, 1991] David McAllester and David Rosenblitt. Systematic nonlinear planning. In *AAAI-91*, pages 634–639. Morgan Kaufmann Publishers, July 1991. internet file ftp.ai.mit.edu:/pub/dam/aaai91c.ps.

[Sacerdoti, 1974] Earl D. Sacerdoti. Planning in a hierarchy of abstraction spaces. *Artificial Intelligence*, 5:115–135, 1974.

[Sacerdoti, 1975] Earl D. Sacerdoti. The nonlinear nature of plans. In *IJCAI75*, pages 206–214, 1975.

[Tate, 1977] Austin Tate. Generating project networks. In *IJCAI77*, pages 888–893, 1977.