University of Malta
BSc(Hons.) IT Year II

Problem Solving Techniques in Artificial Intelligence

Lecture 8

Planning

- Partial Order Planning

Department of Computer Science and AI

Situation Space v. Plan Space

Planning = Search. But search through what?

- **Situation-Space Planners**
  - Search through a space of **situations** that are reached by applying **actions**.
  - The nodes of the search tree are situations, and paths through the search tree are plans.
  - STRIPS searches this space backward from the goal situation toward the current situation, but other strategies are possible.

- **Plan-Space Planners**
  - An alternative approach is to search through the space of **plans**.

Situation Space v. Plan Space

Situation Based	**Plan Based**
$\boxed{\text{Initial Situation}}$	$\boxed{\text{Initial Plan}}$
$\Downarrow$	$\Downarrow$
$O_1 \ldots O_n$	$O_1 \ldots O_n$
$\Downarrow$	$\Downarrow$
$\boxed{\text{Goal Situation}}$	$\boxed{\text{Final Plan}}$

- **Situation-Space Planning:** the operators transform one situation into another.

- **Plan-Space Planning:**, the operators transform one **partial plan** into another.

- **Partial Plans:** the representation used for incompletely specified plans.

How Partial Plans are Partial

A partial plan may be incomplete in several ways.

1. Incomplete steps.
2. Partial Ordering between steps.
3. Presence of **unbound variables**.

A partial plan can be regarded as a representation for a **set** of complete, fully specified plans. As more detail is added, members are removed from the set.

Reasoning with partial plans amounts to reasoning about classes of individual plans.

Principle of least commitment:

$\boxed{\text{Only commit when you have to}}$

Data Structure for Plans

Basic idea is that a plan is a **set** of entities called *plan steps*. over which certain constraints hold

- **Step Ordering Constraints** of the form $S_i \prec S_j$. This means that S_i must precede S_j (but not necessarily *immediately* precede).
- **Variable Binding Constraints** of the form $v = x$ where x is either a constant or another variable.
- **Causal Links** ($S_i \xrightarrow{c} S_j$). S_i achieves c for S_j. Causal links serve to record the purpose of plan steps (here S_i achieves the preconditions of S_j

Solutions

Definition: a **solution** is a plan that an agent can execute and that gurantees achievement of the goal. N.B. A solution does not have to be fully instantiated nor fully ordered.

A **complete plan** is one in which every precondition of every step is achieved by some other step. A step S_i achieves a precondition C of step $S_j$if C $\in$ effects(S_i) and $S_i \prec S_j$ and no other step can possibly cancel out the condition, i.e. there is no other step S_k such that -C $\in$ effects(S_k) and $S_i \prec S_k \prec S_k$

A **consistent plan** is one in which there are no contradictions in either the ordering constraints ($S_i \prec S_j \& S_j \prec S_i$ for some i,j) or the binding constraints (v = a & v=b for two different constants a and b).

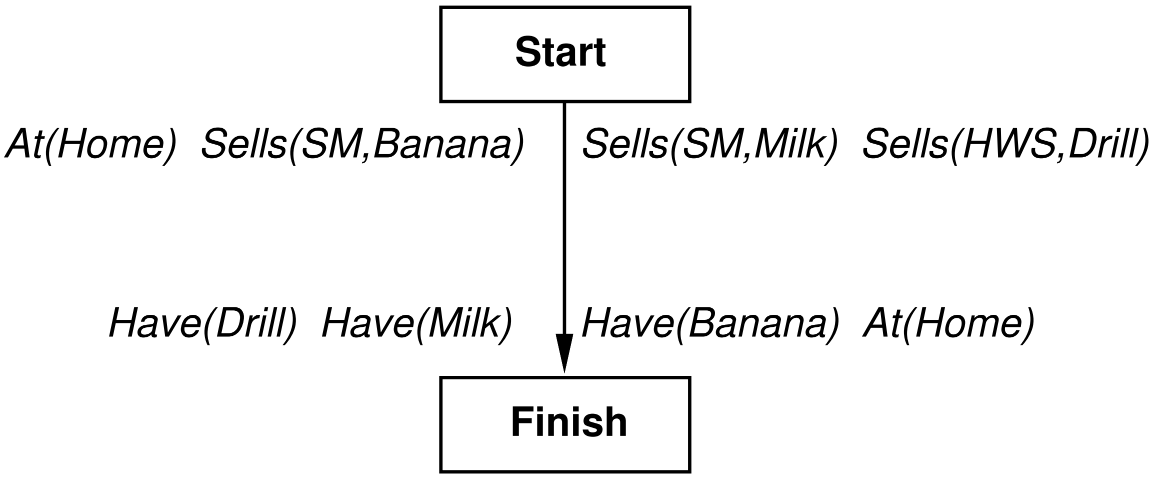
Example

Actions

at(Here)
go(There)
at(There) & -at(Here)

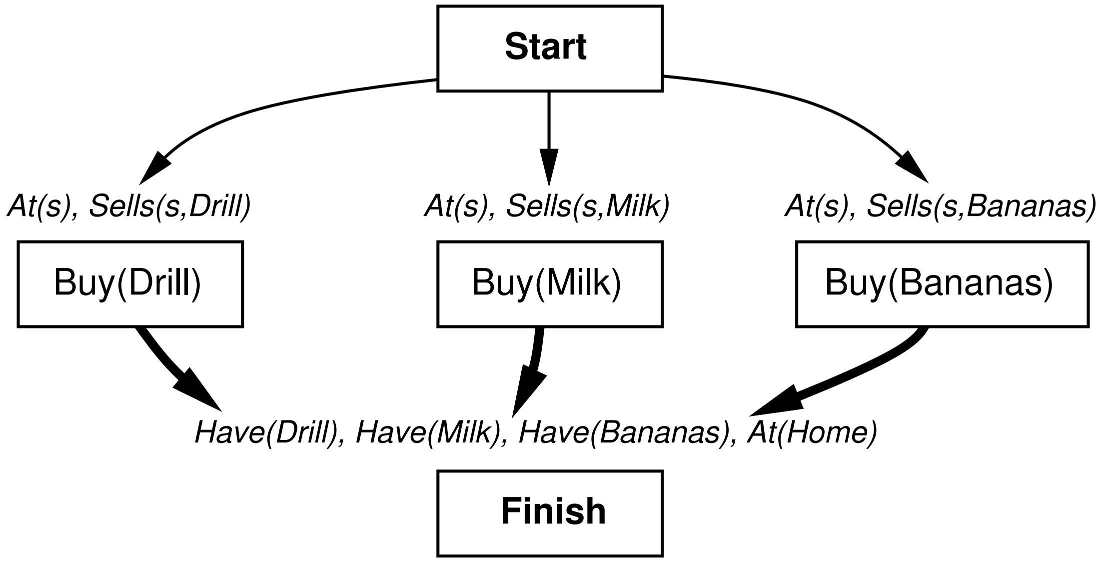
at(S) & sells(S,X)
buy(X)
have(X)

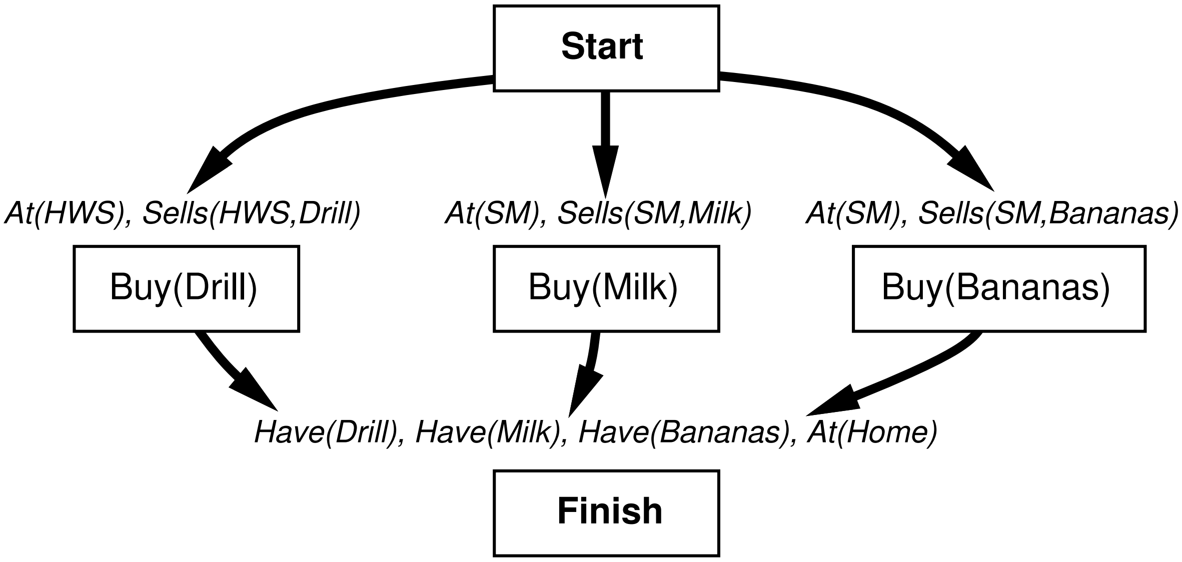
Initial Partial Plan



- **Start:** Special initial step whose addlist contains initial situation
- **Finish:** Special final action whose preconditions are the goal.

Achieving Preconditions of Finish





Remarks

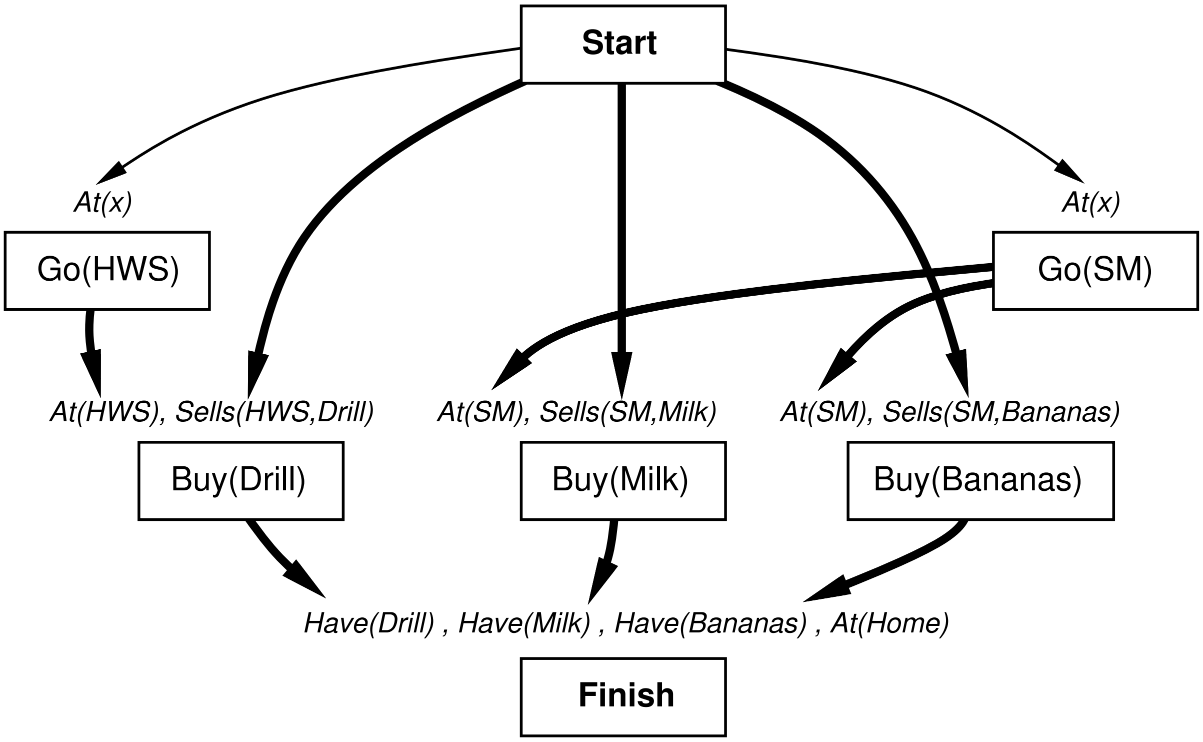
- Bold arrows in the figure are **causal links** between actions and planned effects, e.g. buy(milk) causes the achievement of have(milk), one of the preconditions for for Finish. Such links are protected, so that if another step threatens have(milk), it will not be interposed between the buy step and the Finish.

- Light arrows are ordering constraints. By definition, all steps come after Start. Hence the light arrows to the buy actions.

- Upper figure: achieve three “have” preconditions for the Finish action. This is accomplished by choosing actions whose addlist includes “have”.

- Lower figure: achieve the sells preconditions for the three buying actions. Note that this is accomplished by binding variables.

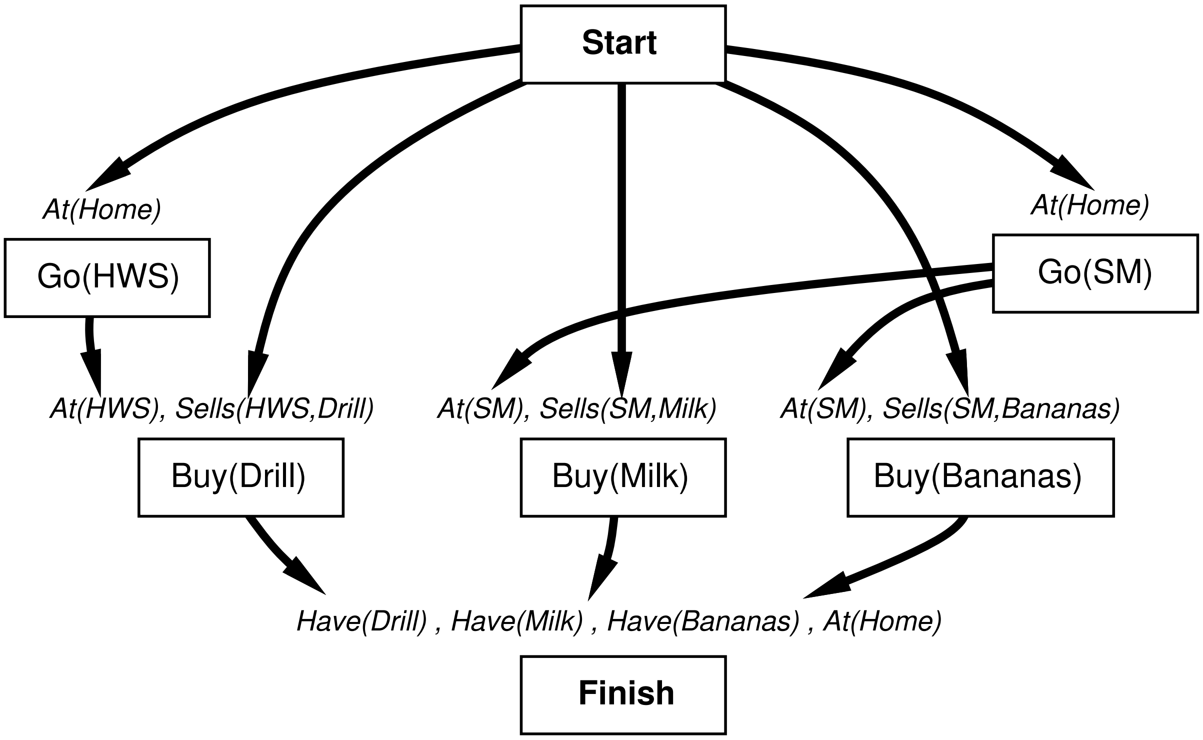
Step 3



- Achieve the "at" preconditions for the three "buy" actions. Only two "go" actions are required.

- Achieve the "at" preconditions for the "go" actions by linking the bindings from the initial situation.

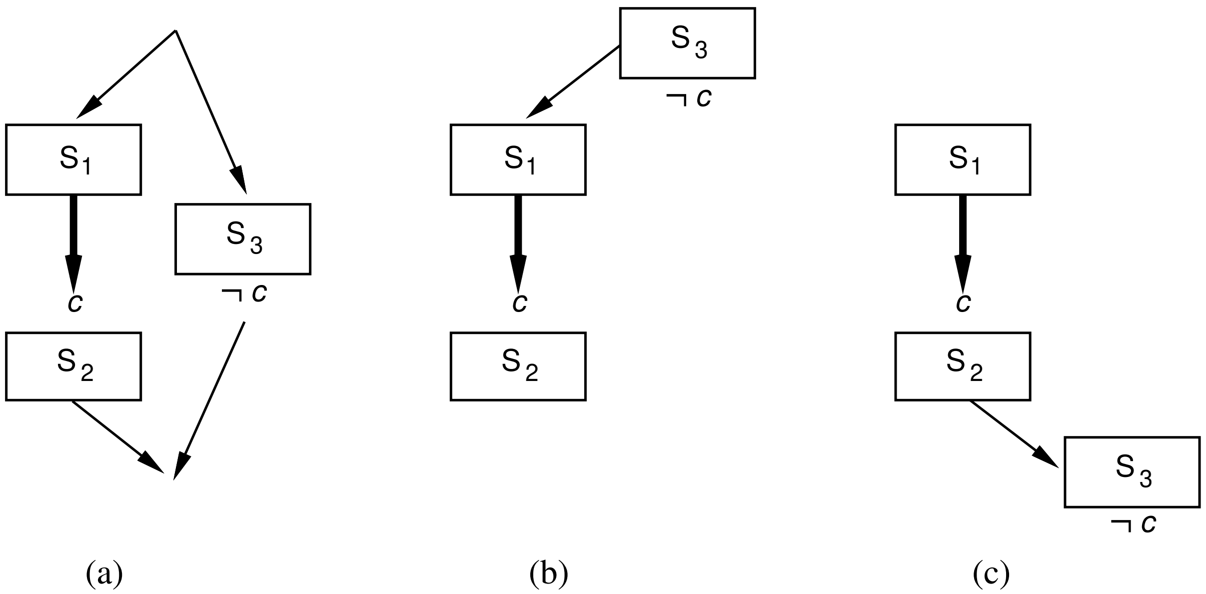
Step 4



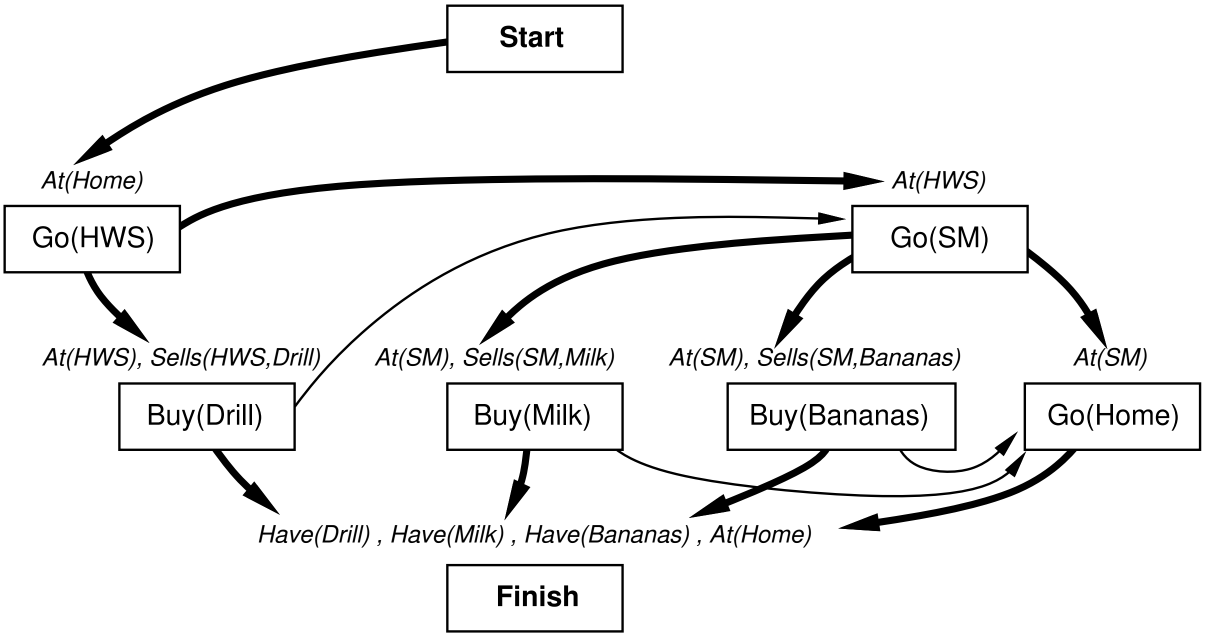
- This plan is flawed.

- go(hws) deletes at(home), thereby threatening the precondition for go(sm). Similarly, if go(sm) is chosen first, it clobbers the precondition for go(hws).

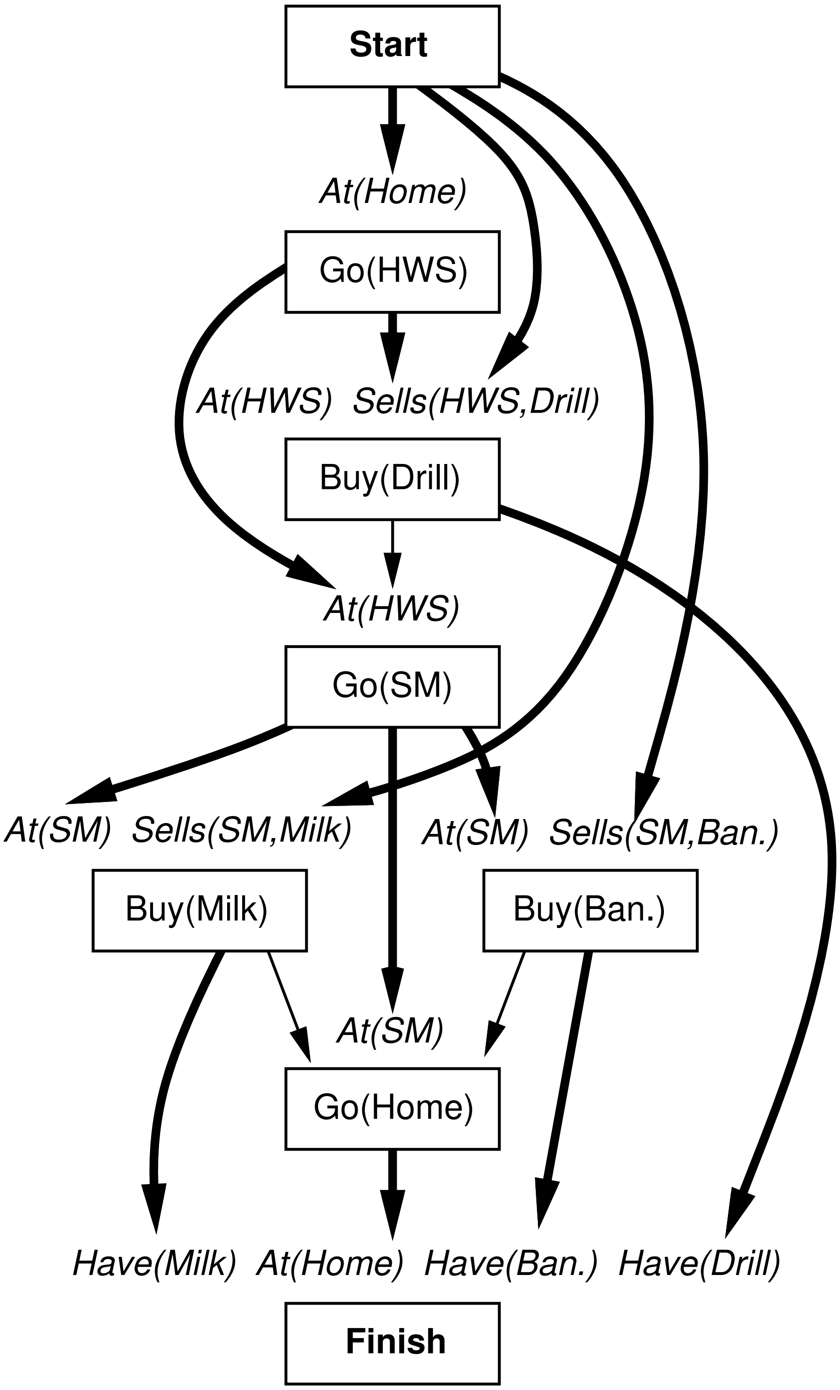
Threat, Promotion, Demotion



Causal Link Protection



Finished Plan



POP Planning Algorithm

```
function POP(initial, goal, operators) returns plan

  plan ← MAKE-MINIMAL-PLAN(initial, goal)
  loop do
     if SOLUTION?(plan) then return plan
     S_need, c ← SELECT-SUBGOAL(plan)
     CHOOSE-OPERATOR(plan, operators, S_need, c)
     RESOLVE-THREATS(plan)
  end
```

```
function SELECT-SUBGOAL(plan) returns S_need, c

  pick a plan step S_need from STEPS(plan)
     with a precondition c that has not been achieved
  return S_need, c
```

```
procedure CHOOSE-OPERATOR(plan, operators, S_need, c)

  choose a step S_add from operators or STEPS(plan) that has c as an effect
  if there is no such step then fail
  add the causal link S_add --c--> S_need to LINKS(plan)
  add the ordering constraint S_add ≺ S_need to ORDERINGS(plan)
  if S_add is a newly added step from operators then
     add S_add to STEPS(plan)
     add Start ≺ S_add ≺ Finish to ORDERINGS(plan)
```

```
procedure RESOLVE-THREATS(plan)

  for each S_threat that threatens a link S_i --c--> S_j in LINKS(plan) do
     choose either
        Promotion: Add S_threat ≺ S_i to ORDERINGS(plan)
        Demotion: Add S_j ≺ S_threat to ORDERINGS(plan)
     if not CONSISTENT(plan) then fail
  end
```

This algorithm is discussed in further detail in Russell and Norvig chapter 11/