

Lecture slides for  
*Automated Planning: Theory and Practice*

# **Chapter 5**

## **Plan-Space Planning**

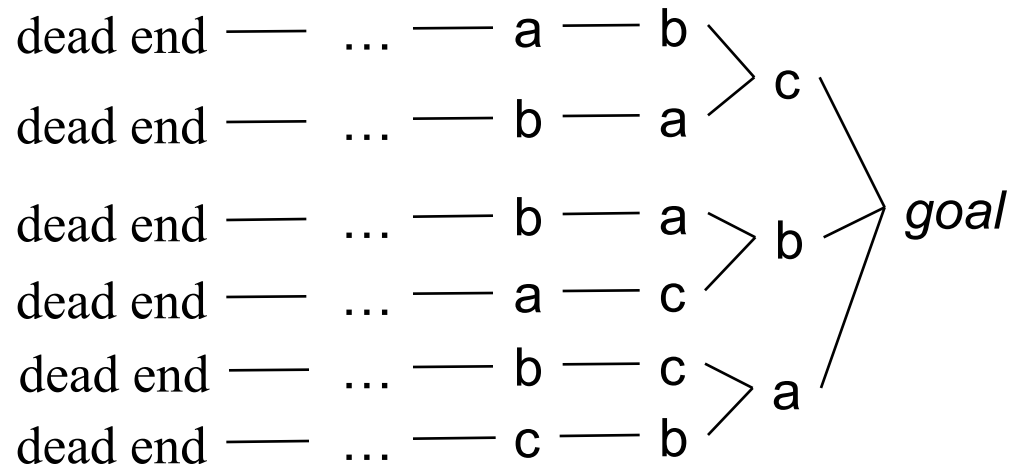
Dana S. Nau

University of Maryland

Fall 2009

# Motivation

- Problem with state-space search
  - ◆ In some cases we may try many different orderings of the same actions before realizing there is no solution



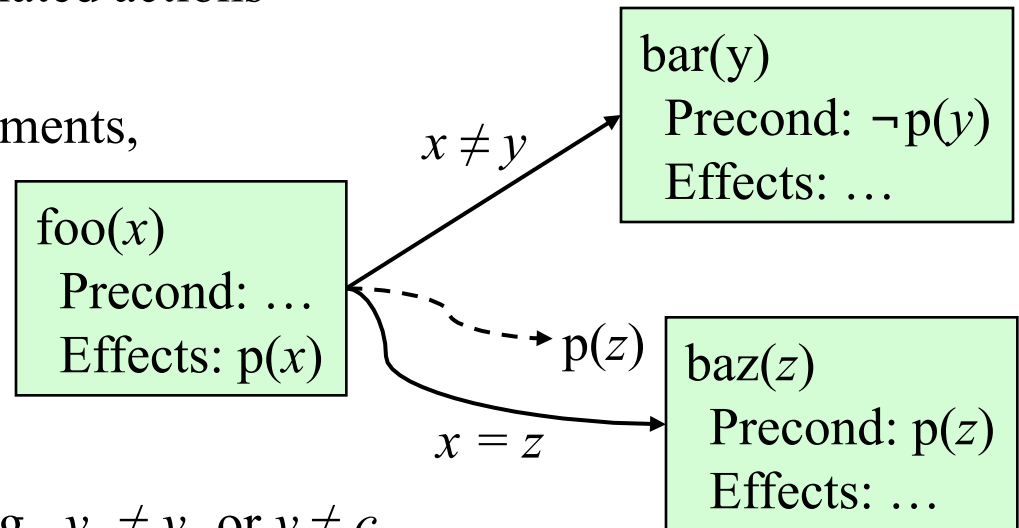
- *Least-commitment strategy*: don't commit to orderings, instantiations, etc., until necessary

# Outline

- Basic idea
- Open goals
- Threats
- The PSP algorithm
- Long example
- Comments

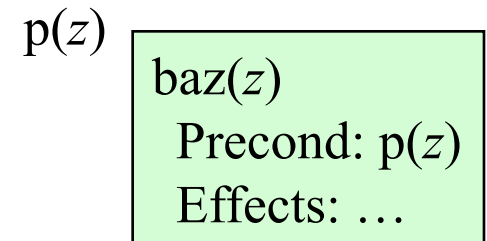
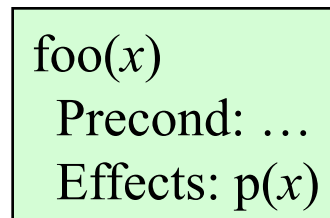
# Plan-Space Planning - Basic Idea

- Backward search from the goal
- Each node of the search space is a *partial plan*
  - » A set of partially-instantiated actions
  - » A set of constraints
  - ◆ Make more and more refinements, until we have a solution
- Types of constraints:
  - ◆ *precedence constraint*:  
 $a$  must precede  $b$
  - ◆ *binding constraints*:
    - » inequality constraints, e.g.,  $v_1 \neq v_2$  or  $v \neq c$
    - » equality constraints (e.g.,  $v_1 = v_2$  or  $v = c$ ) and/or substitutions
  - ◆ *causal link*:
    - » use action  $a$  to establish the precondition  $p$  needed by action  $b$
- How to tell we have a solution: no more *flaws* in the plan
  - ◆ Will discuss flaws and how to resolve them

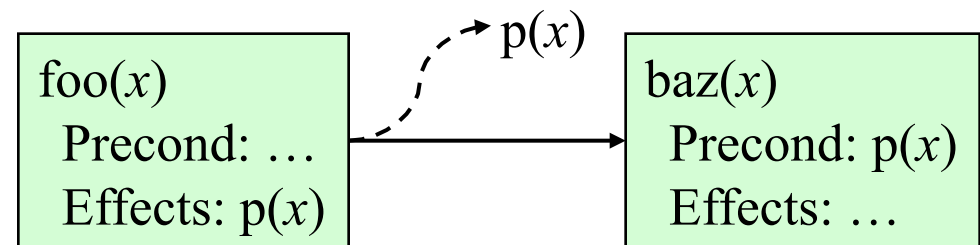


# Flaws: 1. Open Goals

- Open goal:
  - ◆ An action  $a$  has a precondition  $p$  that we haven't decided how to establish

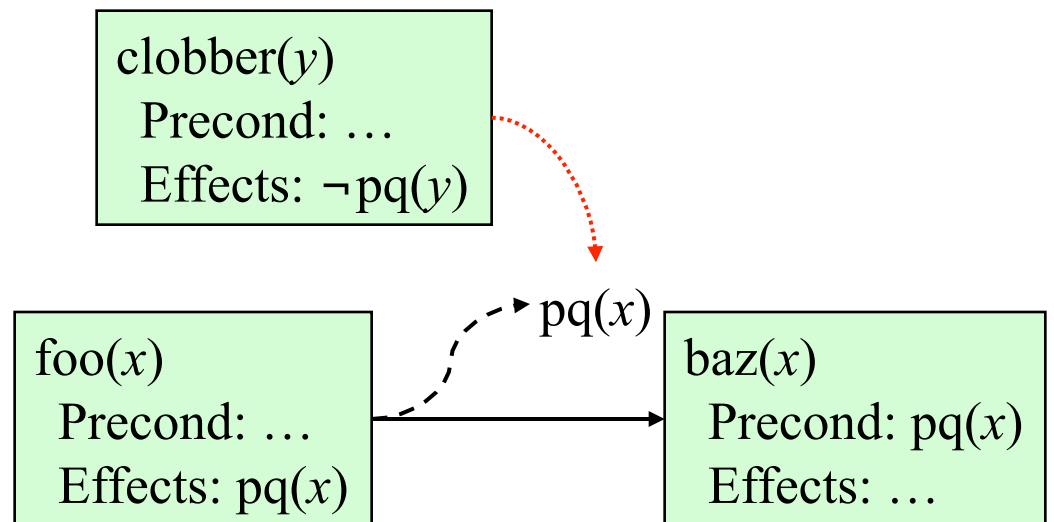


- Resolving the flaw:
  - ◆ Find an action  $b$ 
    - (either already in the plan, or insert it)
  - ◆ that can be used to establish  $p$ 
    - can precede  $a$  and produce  $p$
  - ◆ Instantiate variables and/or constrain variable bindings
  - ◆ Create a causal link



## Flaws: 2. Threats

- Threat: a deleted-condition interaction
  - ◆ Action  $a$  establishes a precondition (e.g.,  $pq(x)$ ) of action  $b$
  - ◆ Another action  $c$  is capable of deleting  $p$
- Resolving the flaw:
  - ◆ impose a constraint to prevent  $c$  from deleting  $p$
- Three possibilities:
  - ◆ Make  $b$  precede  $c$
  - ◆ Make  $c$  precede  $a$
  - ◆ Constrain variable(s) to prevent  $c$  from deleting  $p$



# The PSP Procedure

```
PSP( $\pi$ )
   $flaws \leftarrow \text{OpenGoals}(\pi) \cup \text{Threats}(\pi)$ 
  if  $flaws = \emptyset$  then return( $\pi$ )
  select any flaw  $\phi \in flaws$ 
   $resolvers \leftarrow \text{Resolve}(\phi, \pi)$ 
  if  $resolvers = \emptyset$  then return(failure)
  nondeterministically choose a resolver  $\rho \in resolvers$ 
   $\pi' \leftarrow \text{Refine}(\rho, \pi)$ 
  return(PSP( $\pi'$ ))
end
```

- PSP is both sound and complete
- It returns a partially ordered solution plan
  - ◆ Any total ordering of this plan will achieve the goals
  - ◆ Or could execute actions in parallel if the environment permits it

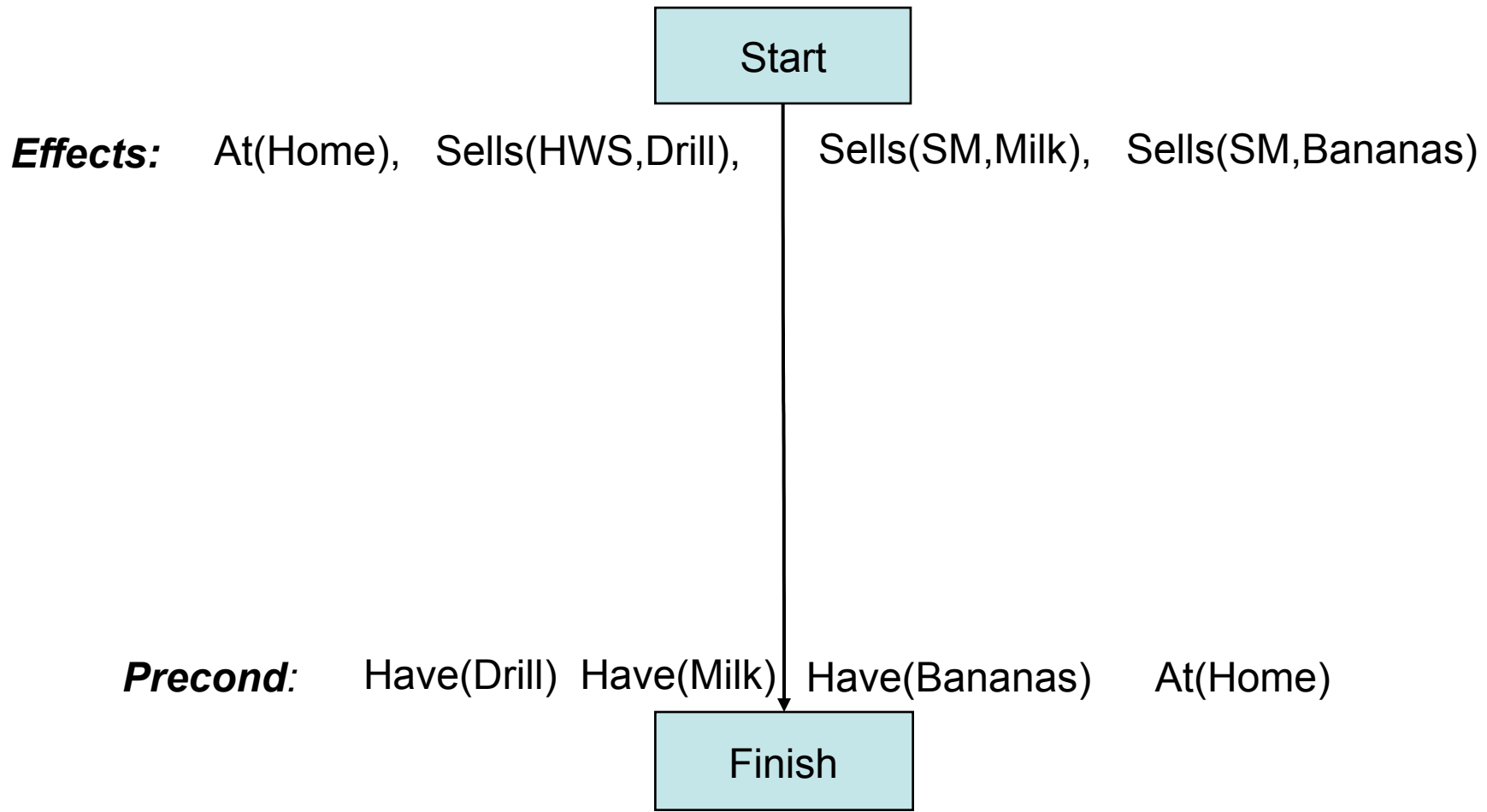
# Example

- Similar (but not identical) to an example in Russell and Norvig's *Artificial Intelligence: A Modern Approach* (1st edition)
- Operators:
  - ◆ **Start**
    - Precond: none
    - Effects: At(Home), sells(HWS,Drill), Sells(SM,Milk), Sells(SM,Banana)
  - ◆ **Finish**
    - Precond: Have(Drill), Have(Milk), Have(Banana), At(Home)
  - ◆ **Go(*l,m*)**
    - Precond: At(*l*)
    - Effects: At(*m*),  $\neg$ At(*l*)
  - ◆ **Buy(*p,s*)**
    - Precond: At(*s*), Sells(*s,p*)
    - Effects: Have(*p*)

**Start** and **Finish** are dummy actions that we'll use instead of the initial state and goal

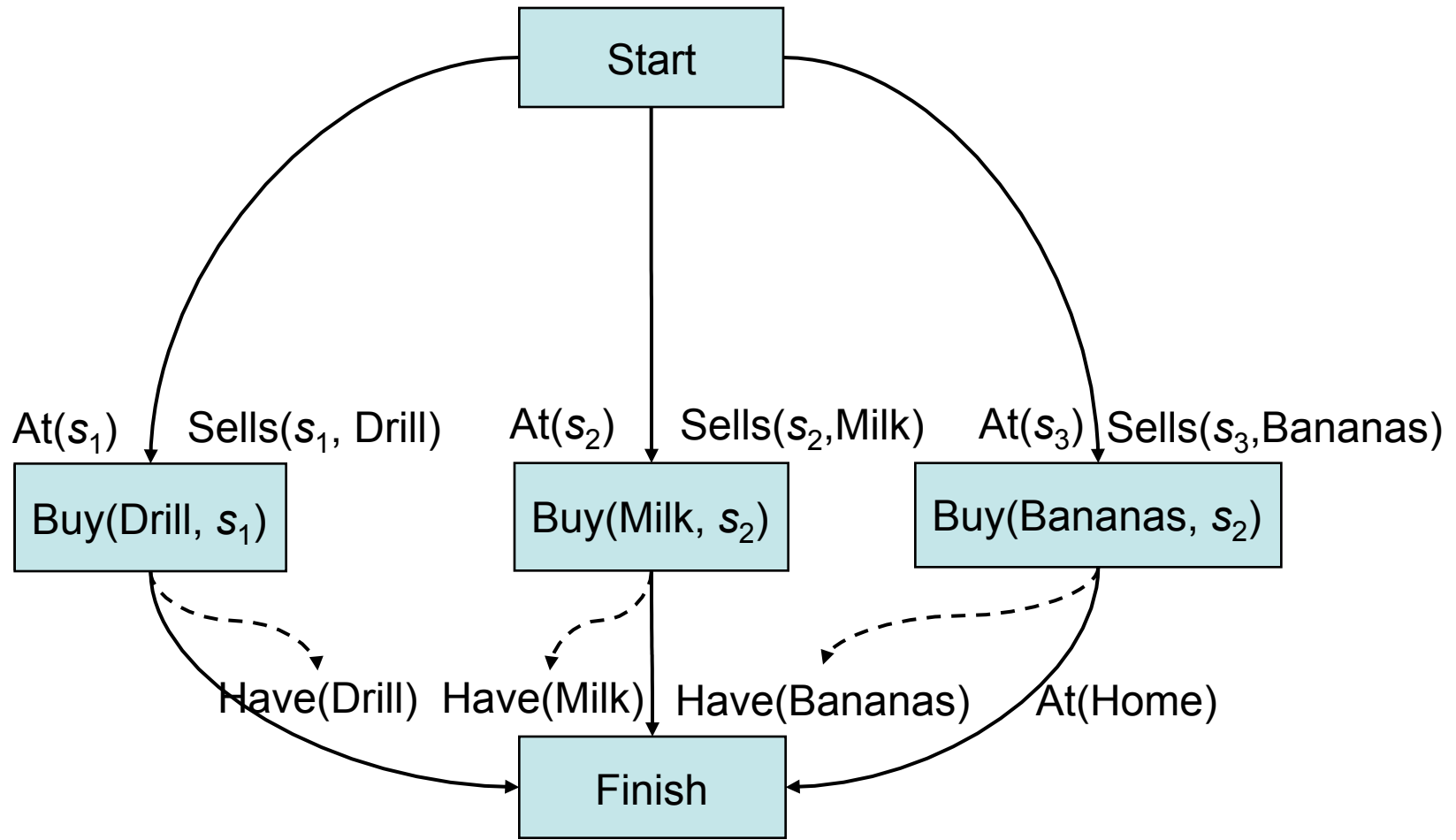
# Example (continued)

- Need to give PSP a plan  $\pi$  as its argument
  - ◆ Initial plan: **Start**, **Finish**, and an ordering constraint



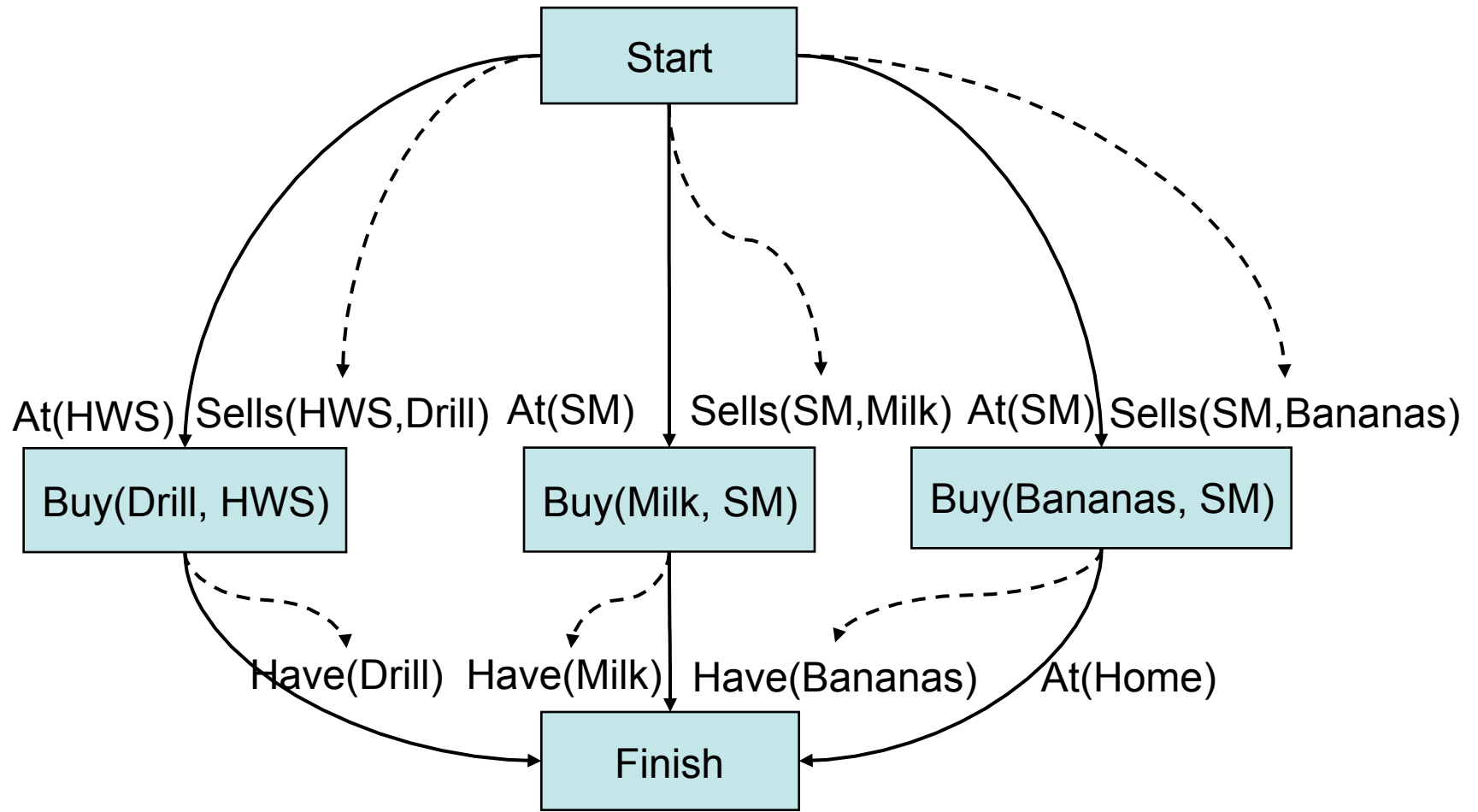
# Example (continued)

- The first three refinement steps
  - ◆ These are the only possible ways to establish the Have preconditions



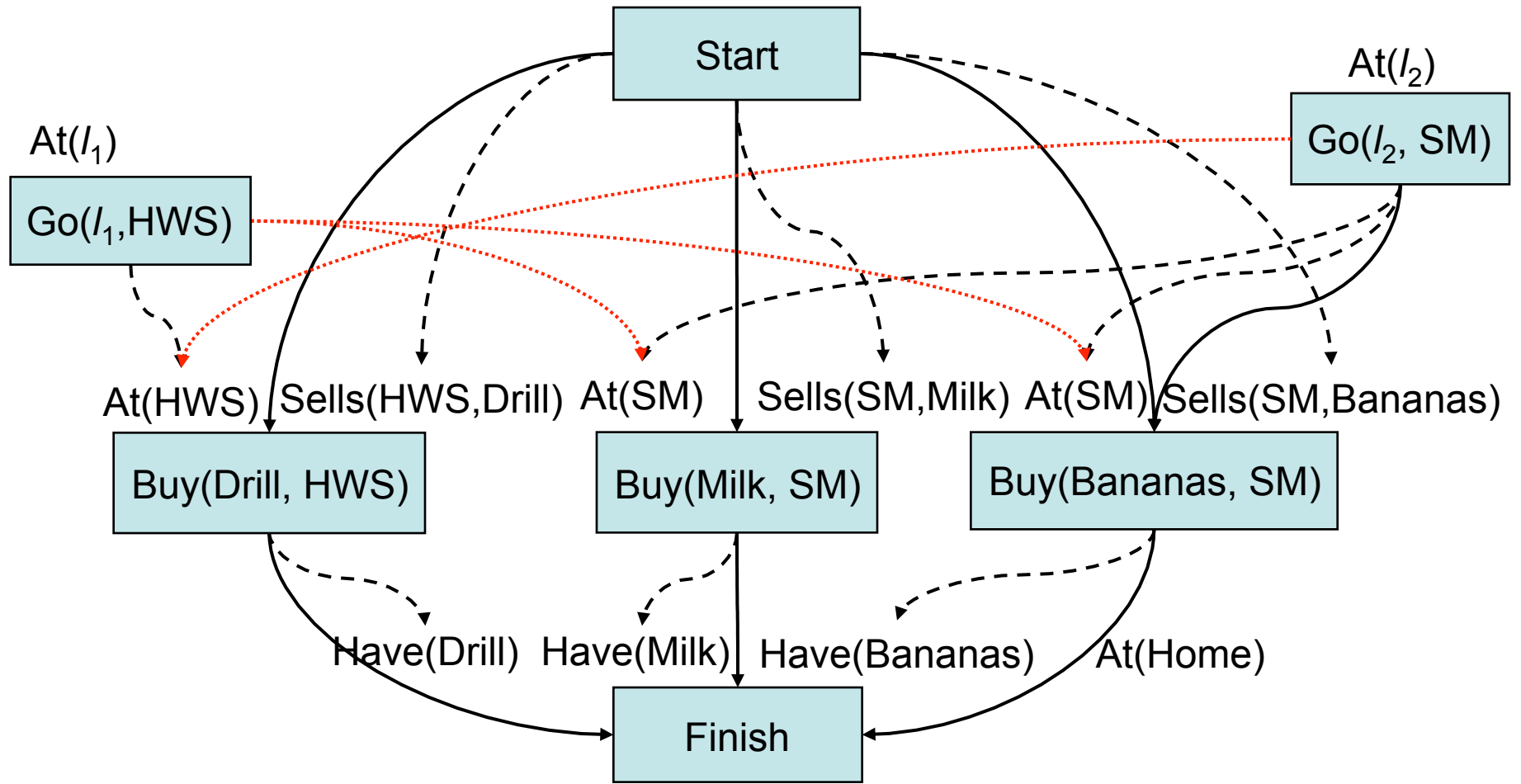
# Example (continued)

- Three more refinement steps
  - ◆ The only possible ways to establish the Sells preconditions



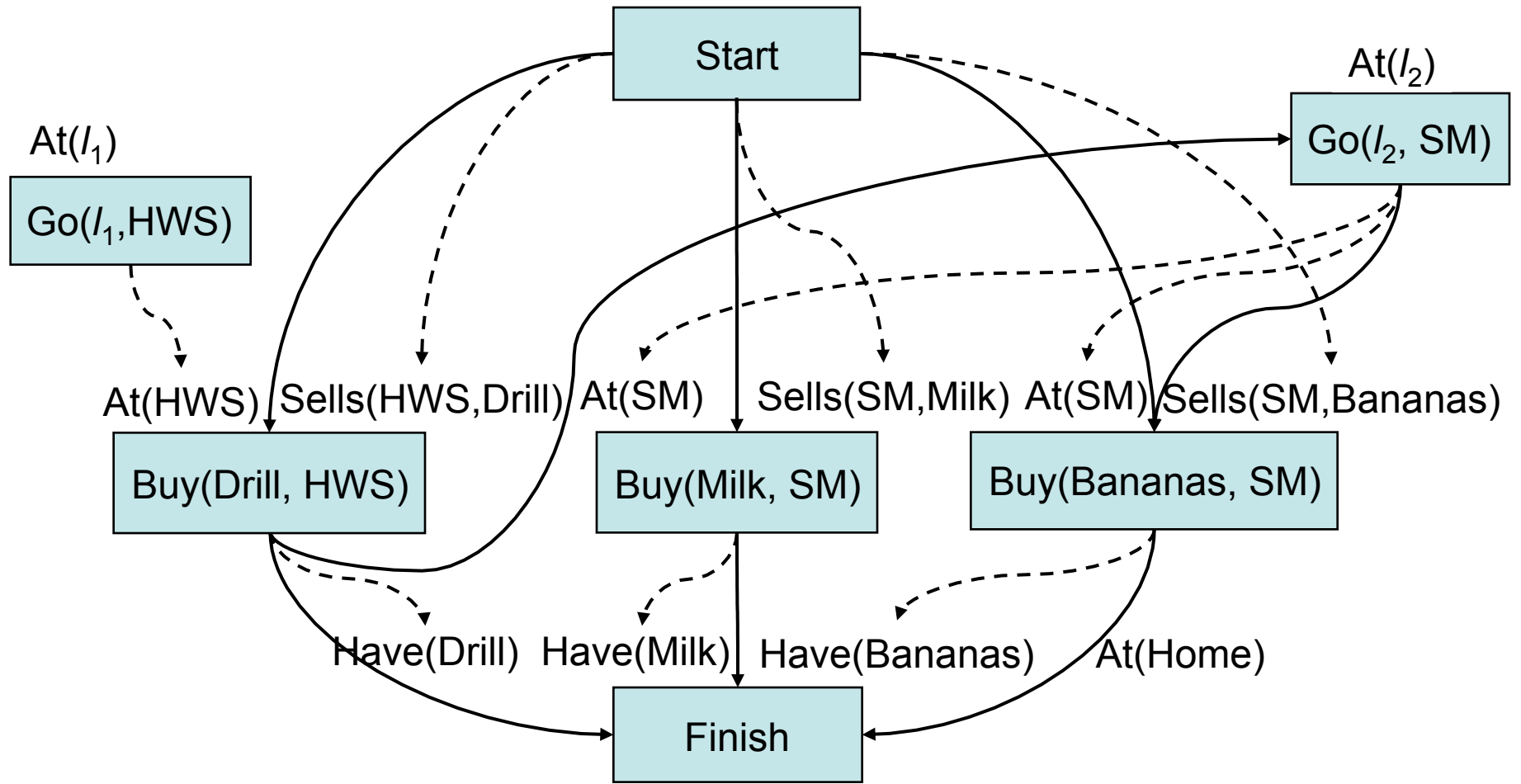
# Example (continued)

- Two more refinements: the only ways to establish  $At(HWS)$  and  $At(SM)$ 
  - ◆ This time, several threats occur



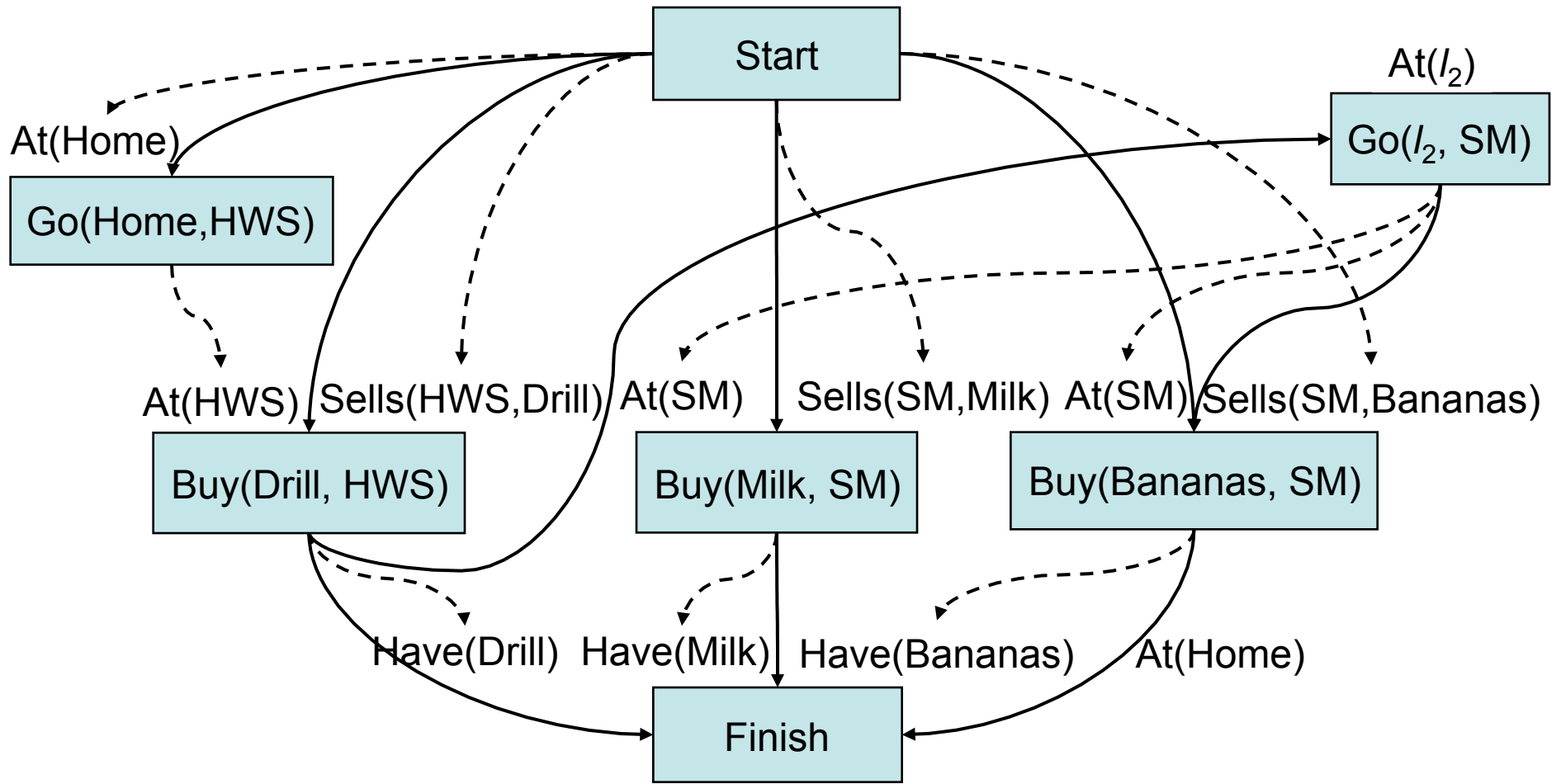
# Example (continued)

- Finally, a nondeterministic choice: how to resolve the threat to  $At(s_1)$ ?
  - ◆ Our choice: make  $Buy(Drill)$  precede  $Go(SM)$
  - ◆ This also resolves the other two threats



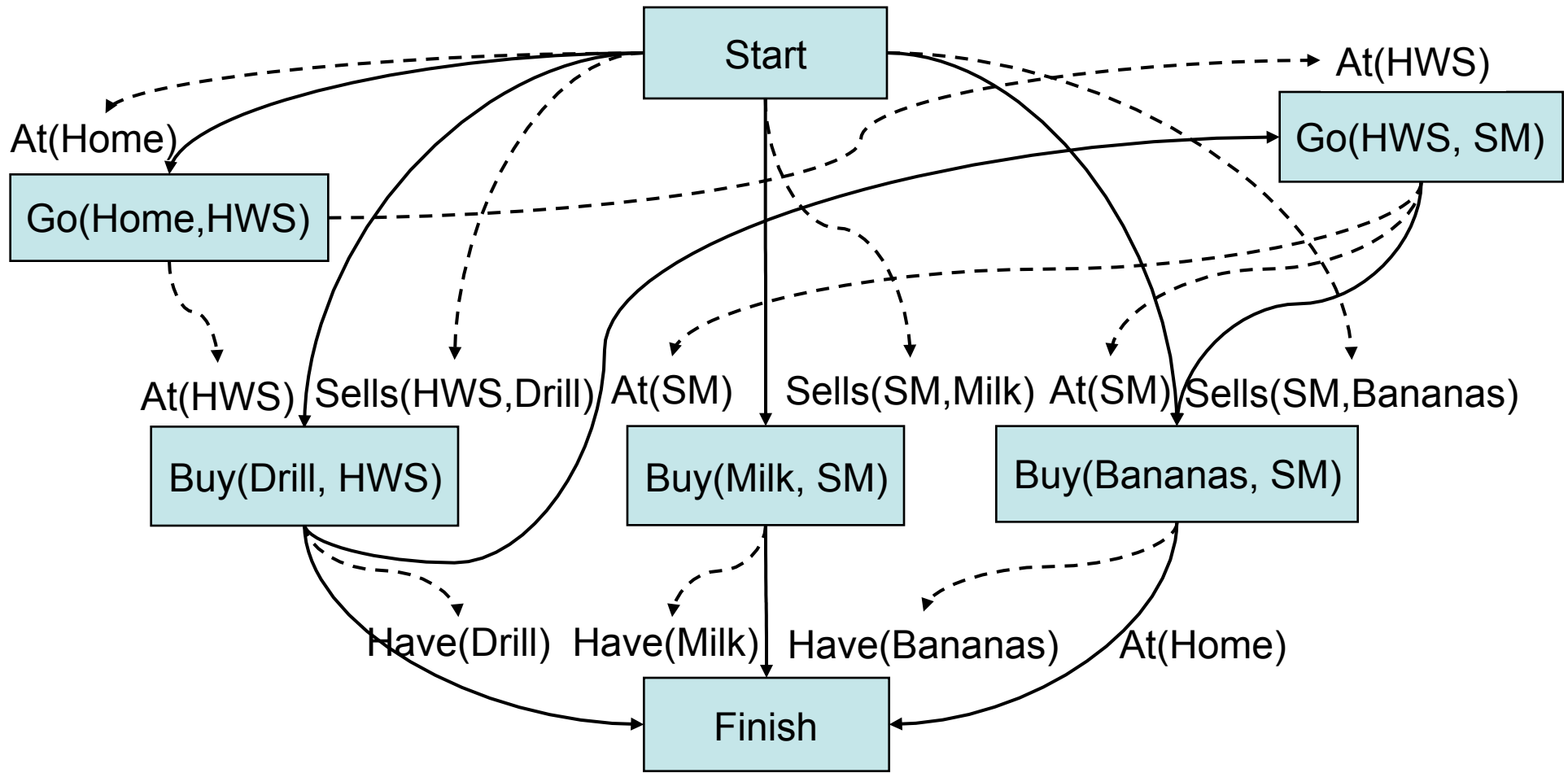
# Example (continued)

- Nondeterministic choice: how to establish  $At(l_1)$ ?
  - ◆ We'll do it from Start, with  $l_1=Home$



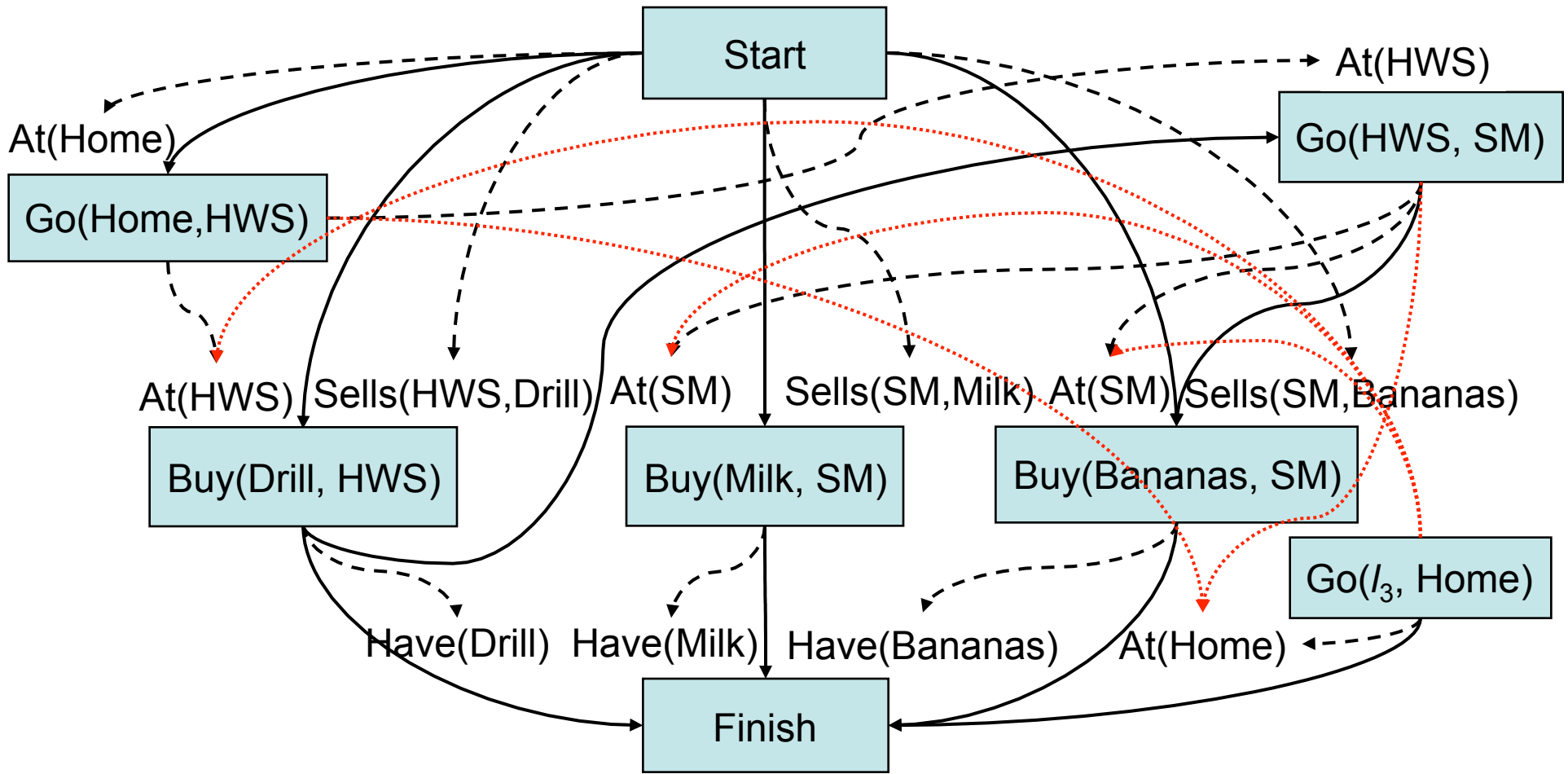
# Example (continued)

- Nondeterministic choice: how to establish  $At(l_2)$ ?
  - ◆ We'll do it from  $Go(Home, HWS)$ , with  $l_2 = HWS$



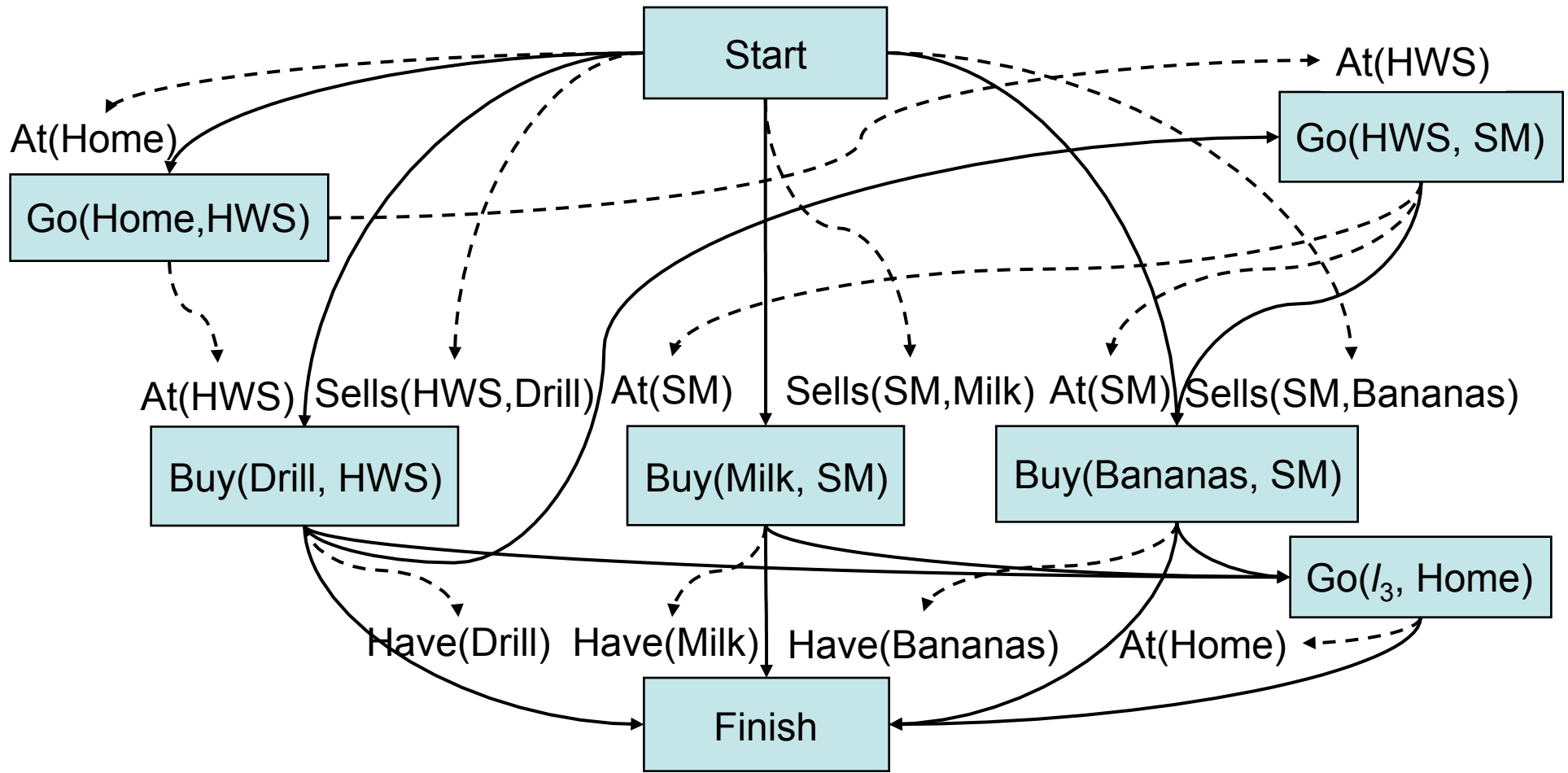
# Example (continued)

- The only possible way to establish At(Home) for Finish
  - ◆ This creates a bunch of threats



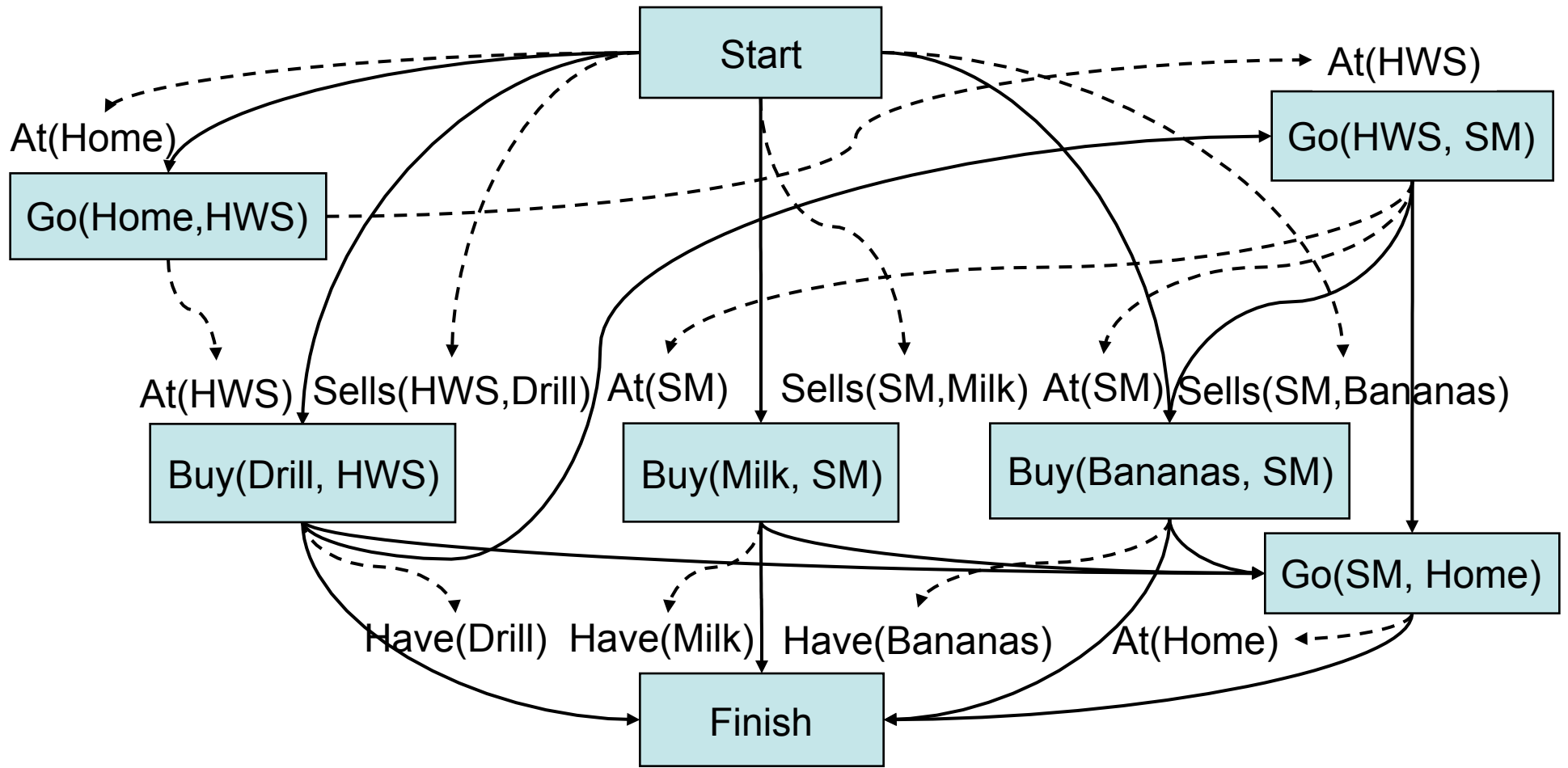
# Example (continued)

- To remove the threats to  $At(SM)$  and  $At(HWS)$ , make them precede  $Go(l_3, Home)$ 
  - ◆ This also removes the other threats



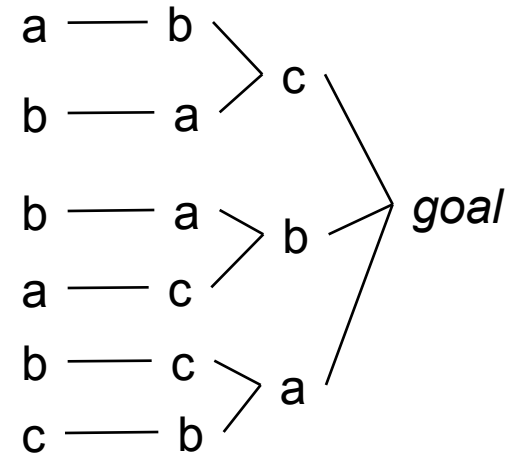
# Final Plan

- Establish  $At(l_3)$  with  $l_3=SM$



# Discussion

- How to choose which flaw to resolve first and how to resolve it?
  - ◆ We'll return to these questions in Chapter 10
- PSP doesn't commit to orderings and instantiations until necessary
  - ◆ Avoids generating search trees like this one:
- Problem: how to prune infinitely long paths?
  - ◆ Loop detection is based on recognizing states we've seen before
  - ◆ In a partially ordered plan, we don't know the states
- Can we prune if we see the same *action* more than once?
  - ... — go(b,a) — go(a,b) — go(b,a) — at(a)
  - ◆ No. Sometimes we might need the same action several times in different states of the world (see next slide)



# Example

- 3-digit binary counter starts at 000, want to get to 110

$$s_0 = \{d_3=0, d_2=0, d_1=0\}$$

$$g = \{d_3=1, d_2=1, d_1=0\}$$

- Operators to increment the counter by 1:

incr0

Precond:  $d_1=0$

Effects:  $d_1=1$

incr01

Precond:  $d_2=0, d_1=1$

Effects:  $d_2=1, d_1=0$

incr011

Precond:  $d_3=0, d_2=1, d_1=1$

Effects:  $d_3=1, d_2=0, d_1=0$

# A Weak Pruning Technique

- Can prune all partial plans of  $n$  or more actions, where  $n = |\{\text{all possible states}\}|$ 
  - ◆ This doesn't help very much
- I'm not sure whether there's a good pruning technique for plan-space planning