Lecture slides for

Automated Planning: Theory and Practice

Chapter 5
Plan-Space Planning

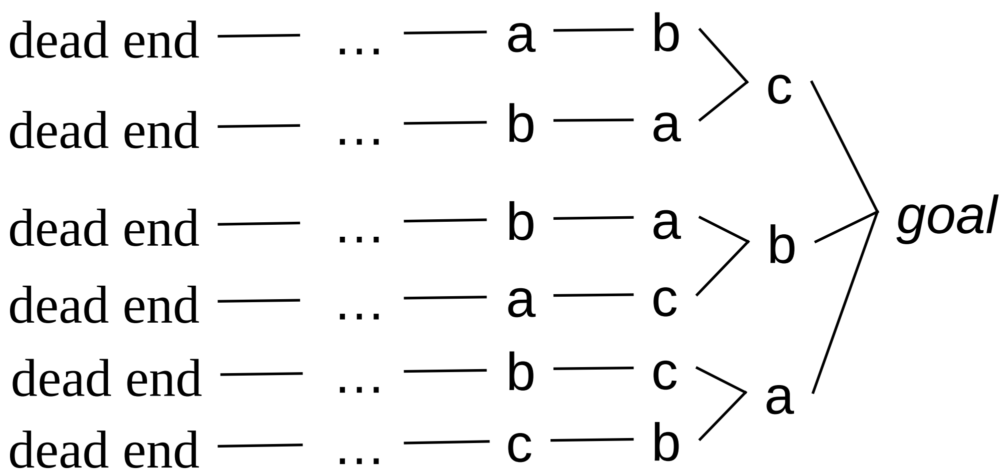
Dana S. Nau

University of Maryland

02:27 PM April 24, 2018

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



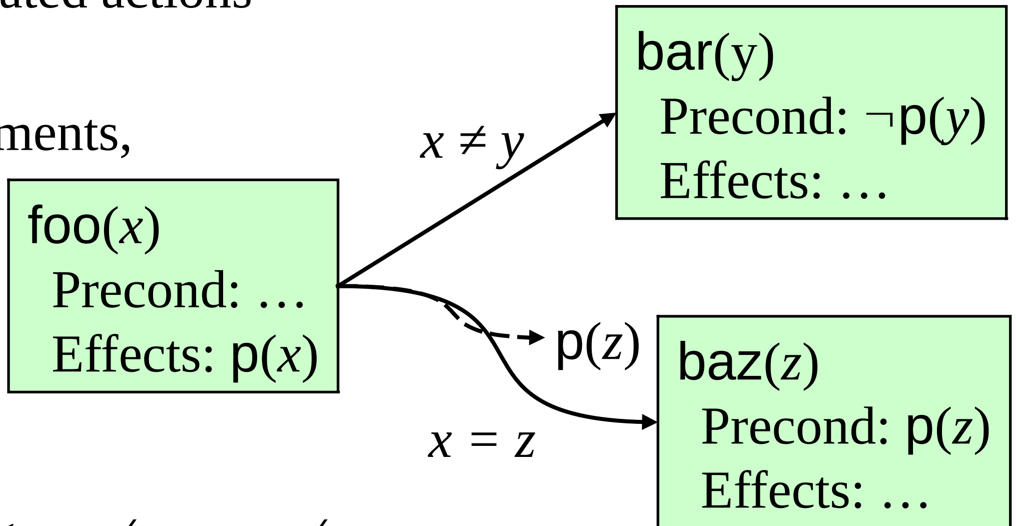
- *Least-commitment strategy:* don't commit to orderings, instantiations, etc., until necessary

Outline

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

Plan-Space Planning - Basic Idea

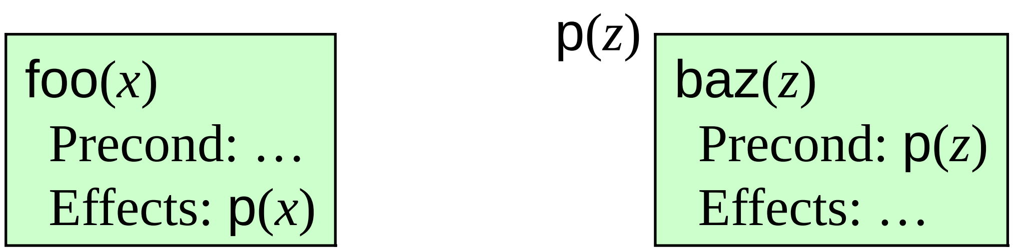
- Backward search from the goal
- Each node of the search space is a *partial plan*
  - » A set of partially-instantiated actions
  - » A set of constraints
  - ◆ Make more and more refinements, until we have a solution



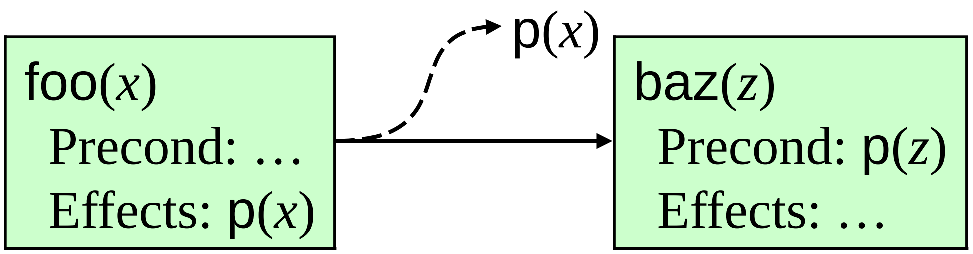
- Types of constraints:
  - ◆ *precedence constraint*: *a* must precede *b*
  - ◆ *binding constraints*:
    - » inequality constraints, e.g., $v_1 \neq v_2$ or $v \neq c$
    - » equality constraints (e.g., $v_1 = v_2$ or $v = c$) and/or substitutions
  - ◆ *causal link*:
    - » use action *a* to establish 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

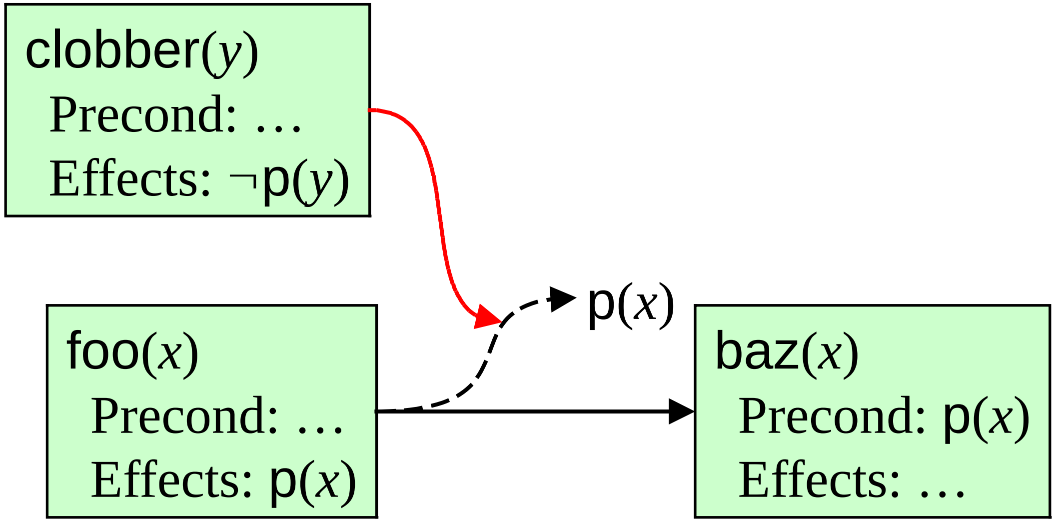


- Resolving the flaw:
  - Find an action b
    - (either already in the plan, or insert it)
  - that can be used to establish p
    - can precede a and produce p
  - Instantiate variables and/or constrain variable bindings
  - Create a causal link



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(π)
    flaws ← OpenGoals(π) ∪ Threats(π)
    if flaws = ∅ then return(π)
    select any flaw φ ∈ flaws
    resolvers ← Resolve(φ, π)
    if resolvers = ∅ then return(failure)
    nondeterministically choose a resolver ρ ∈ resolvers
    π′ ← Refine(ρ, π)
    return(PSP(π′))
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

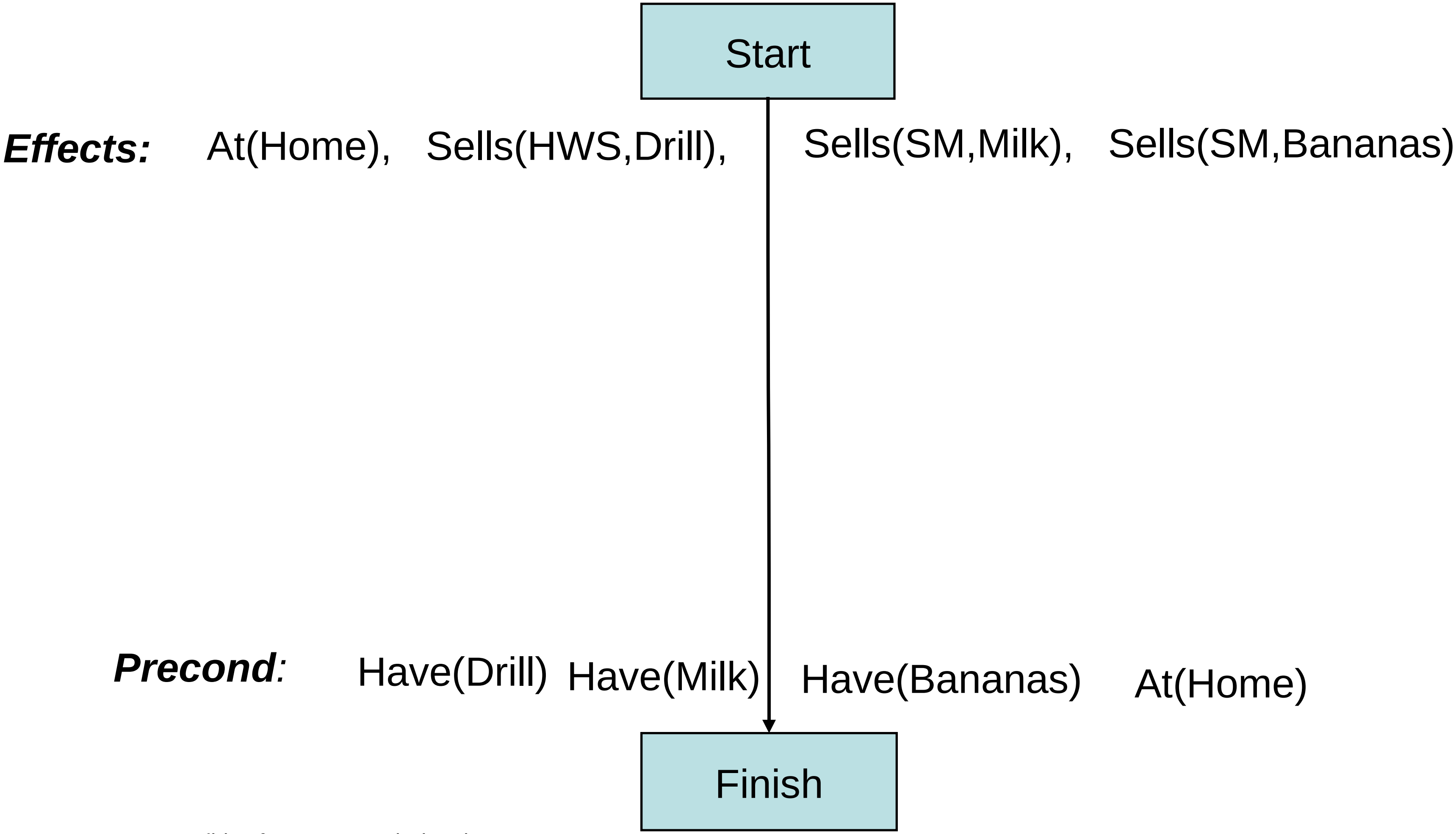
Example

- 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), ¬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

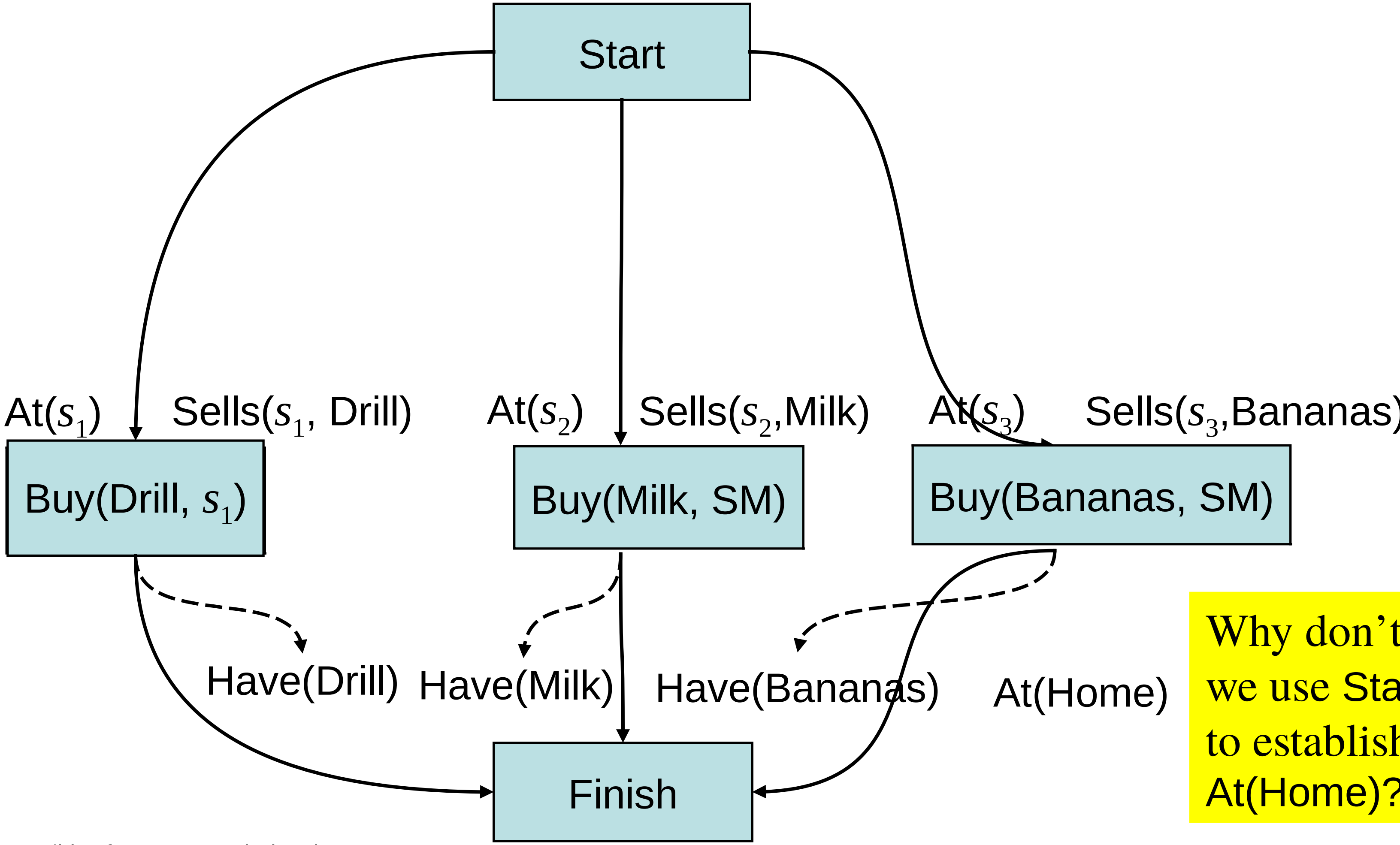
Example (continued)

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



Example (continued)

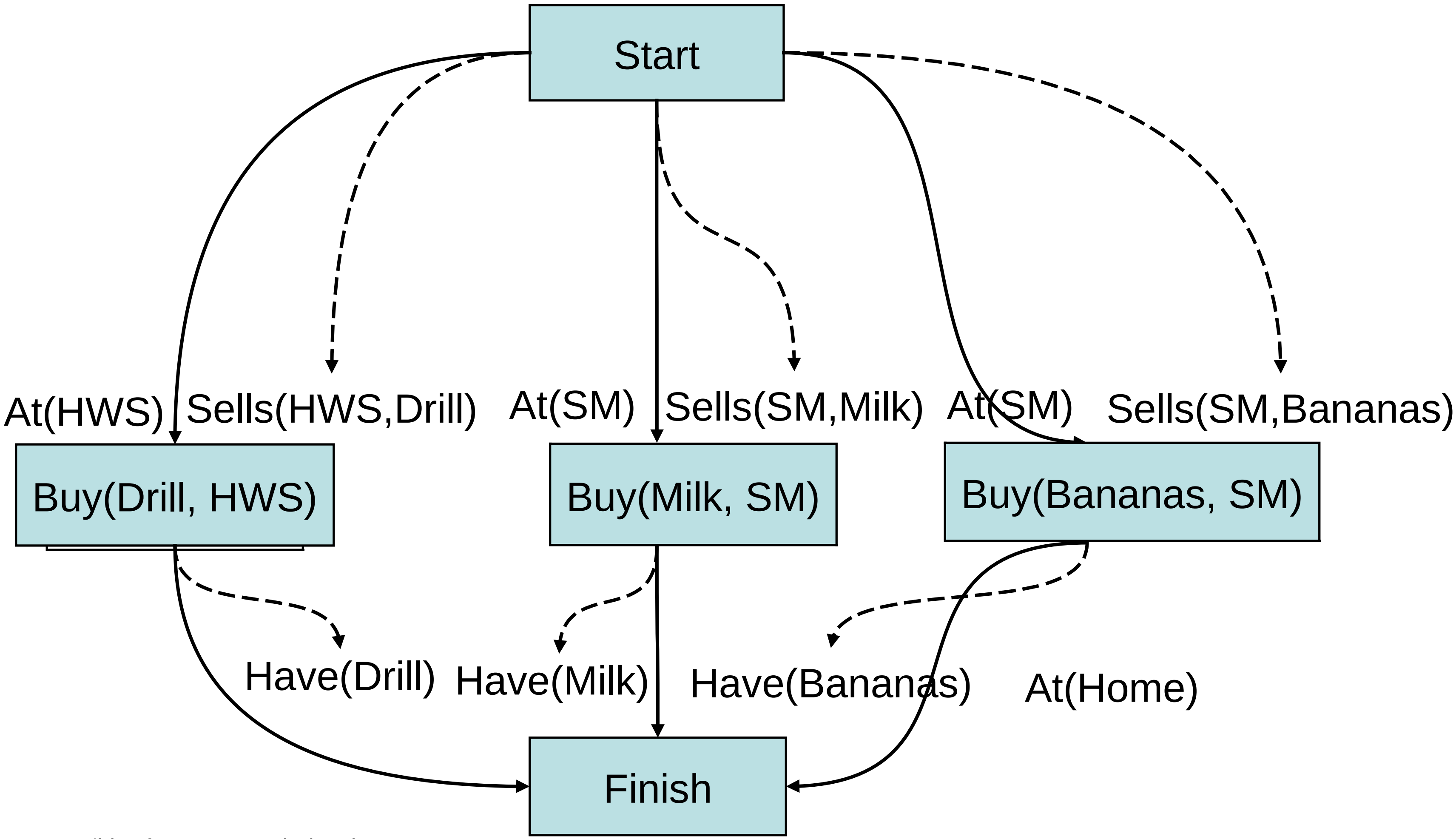
- The first three refinement steps
  - These are the only possible ways to establish the Have preconditions



Why don't we use Start to establish At(Home)?

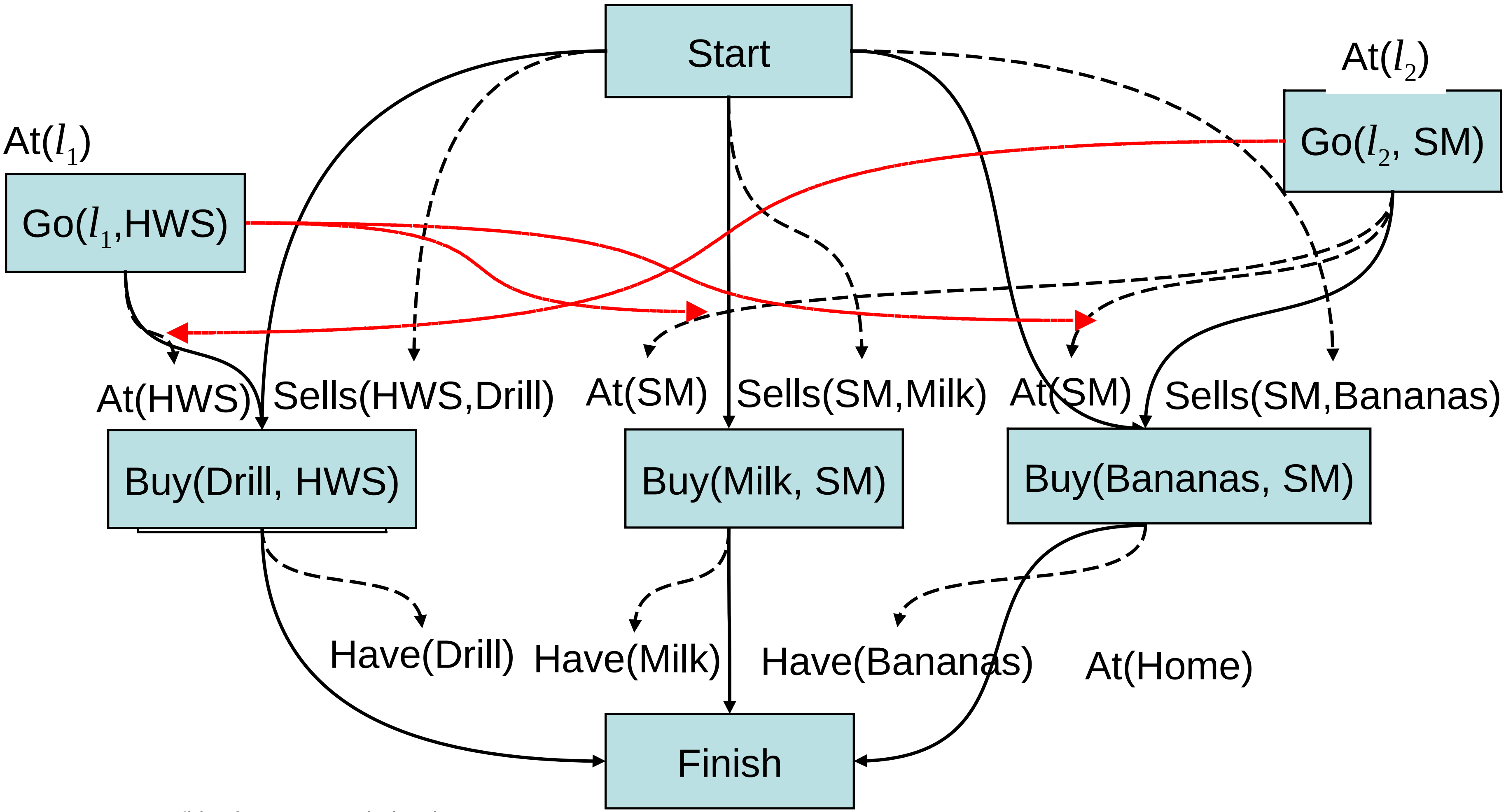
Example (continued)

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



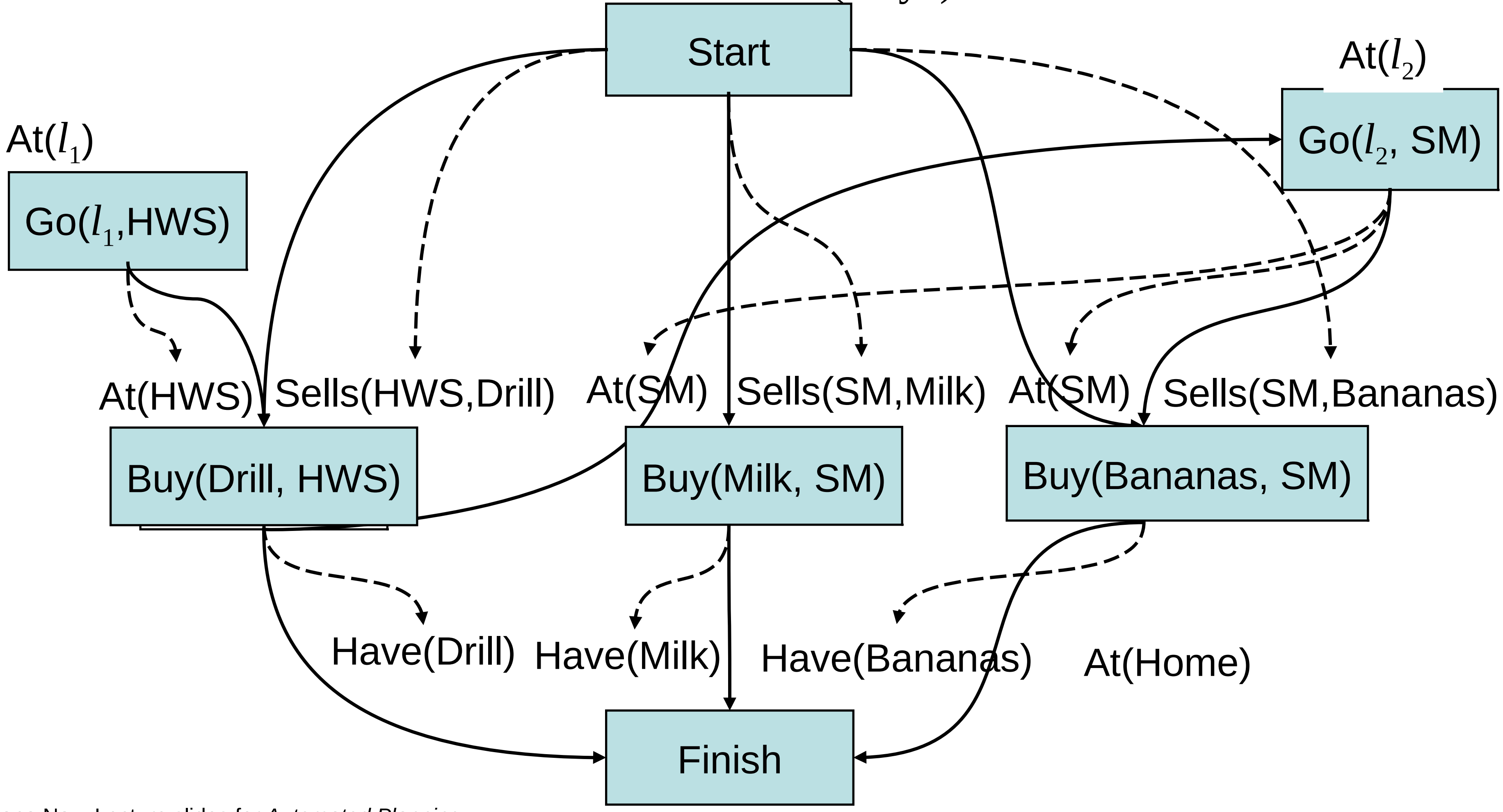
Example (continued)

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



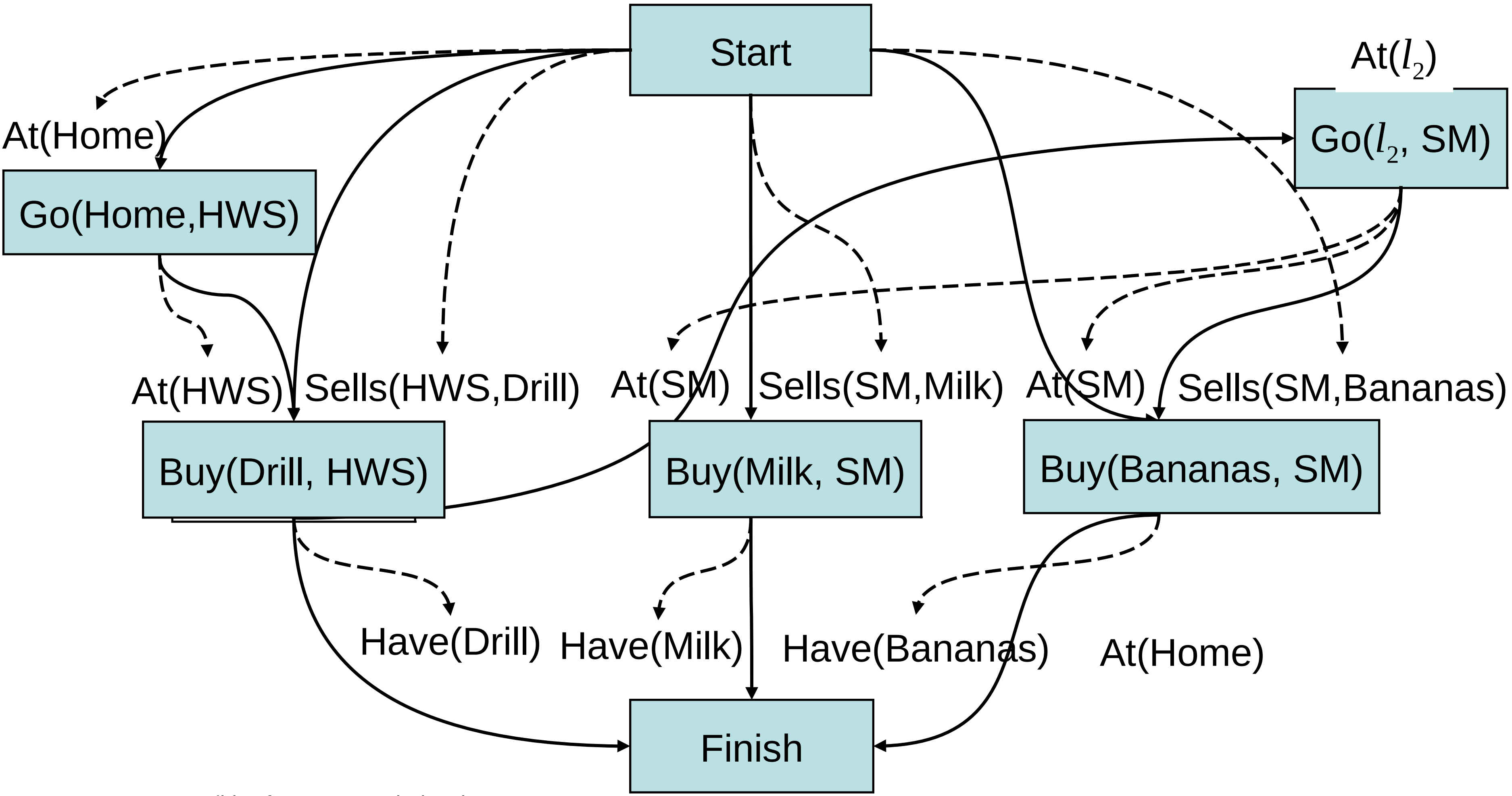
Example (continued)

- Nondeterministic choice: how to resolve the threat to At(s_1)?
  - Our choice: make Buy(Drill) precede Go(l_2, SM)
  - This also resolves the other two threats (why?)



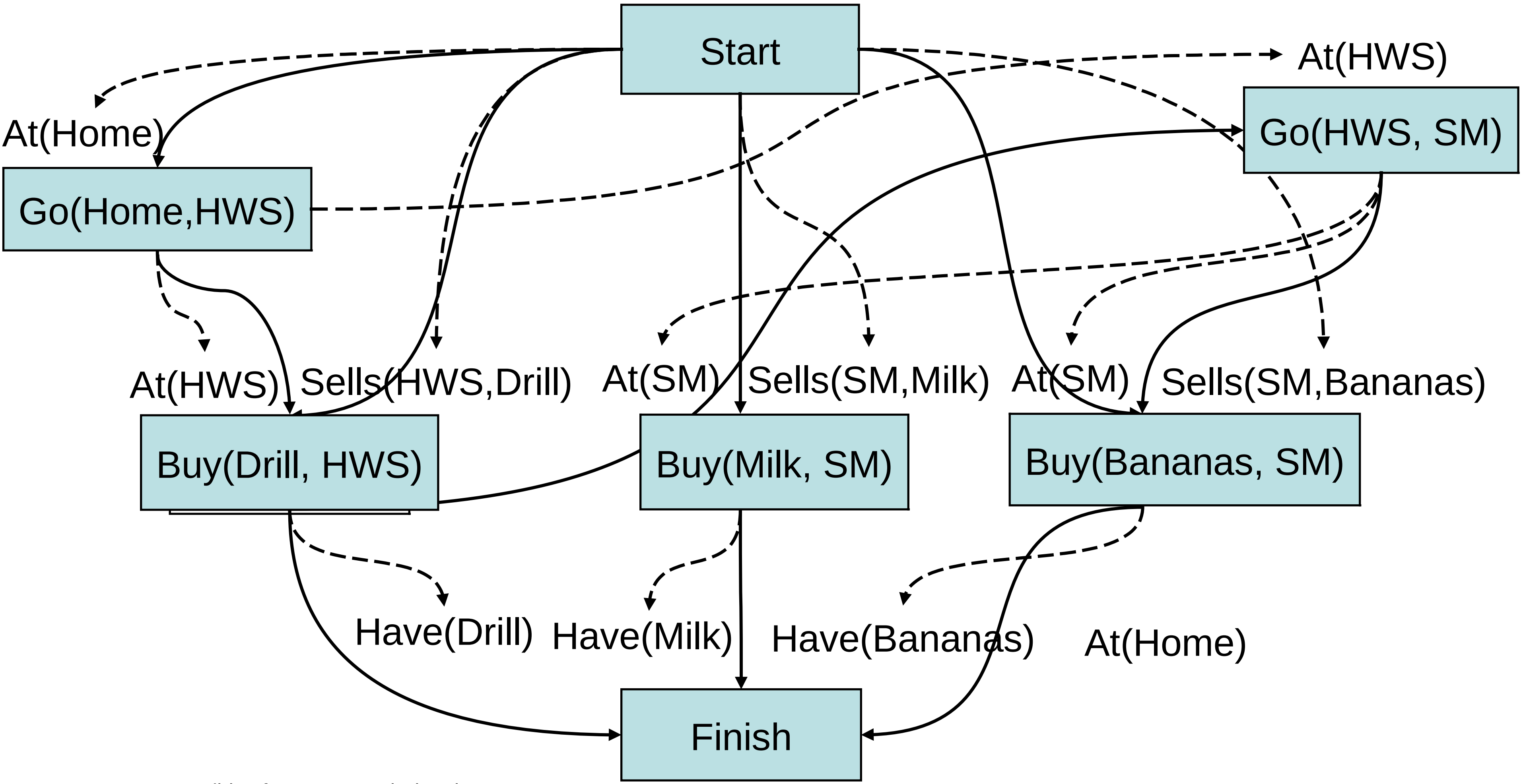
Example (continued)

- Nondeterministic choice: how to establish At(l_1)?
  - We'll do it from Start, with l_1=Home
  - How else could we have done it?



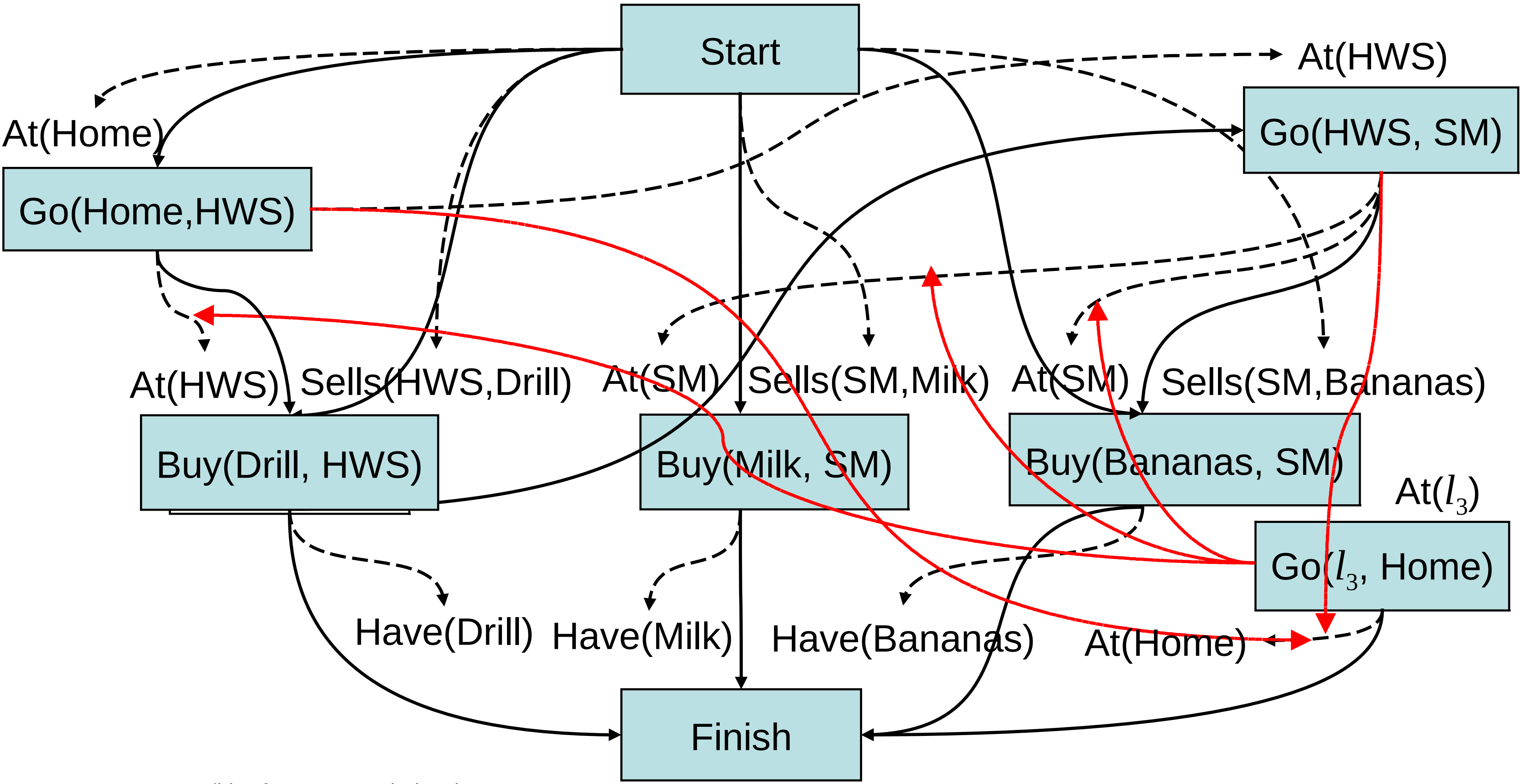
Example (continued)

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



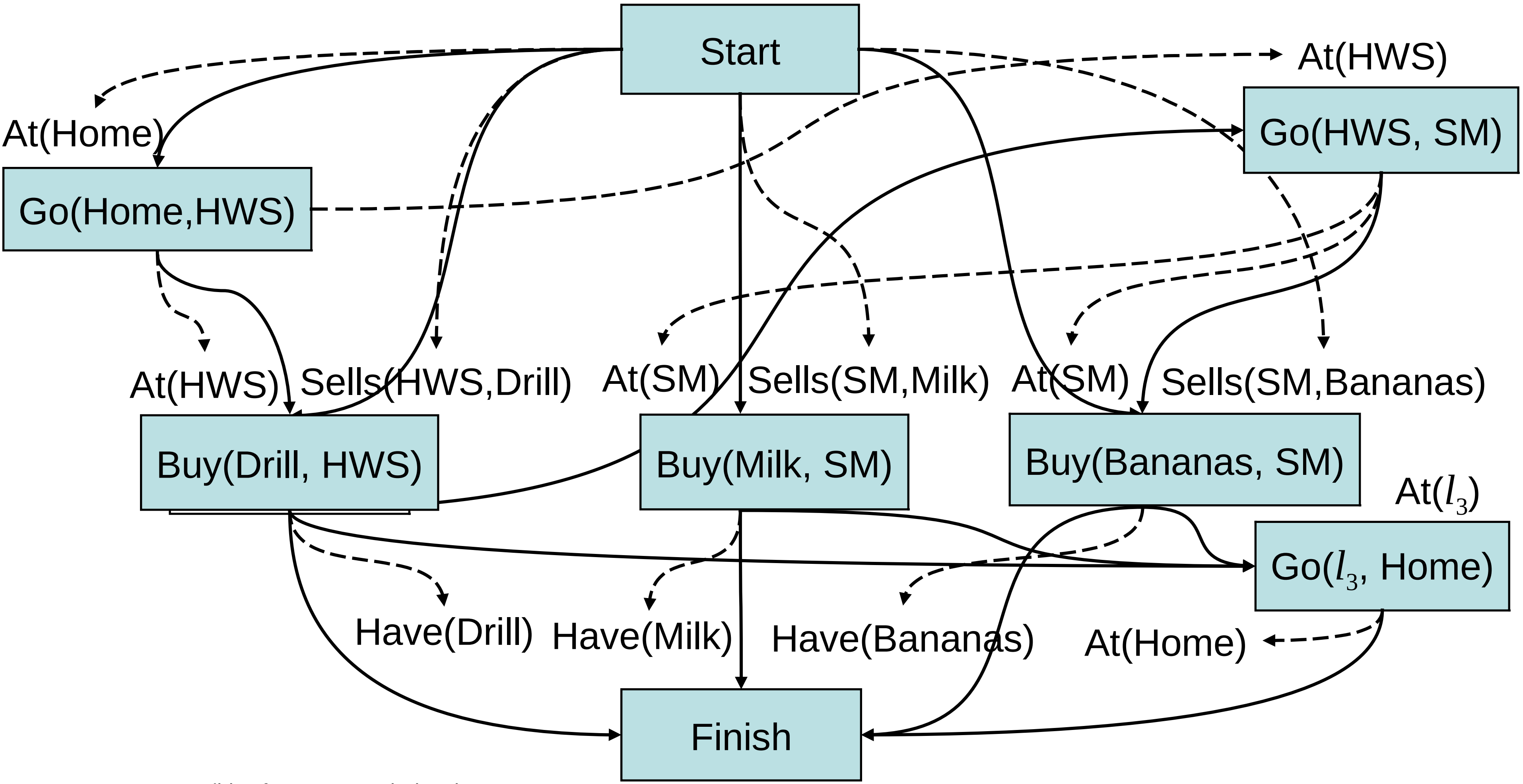
Example (continued)

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



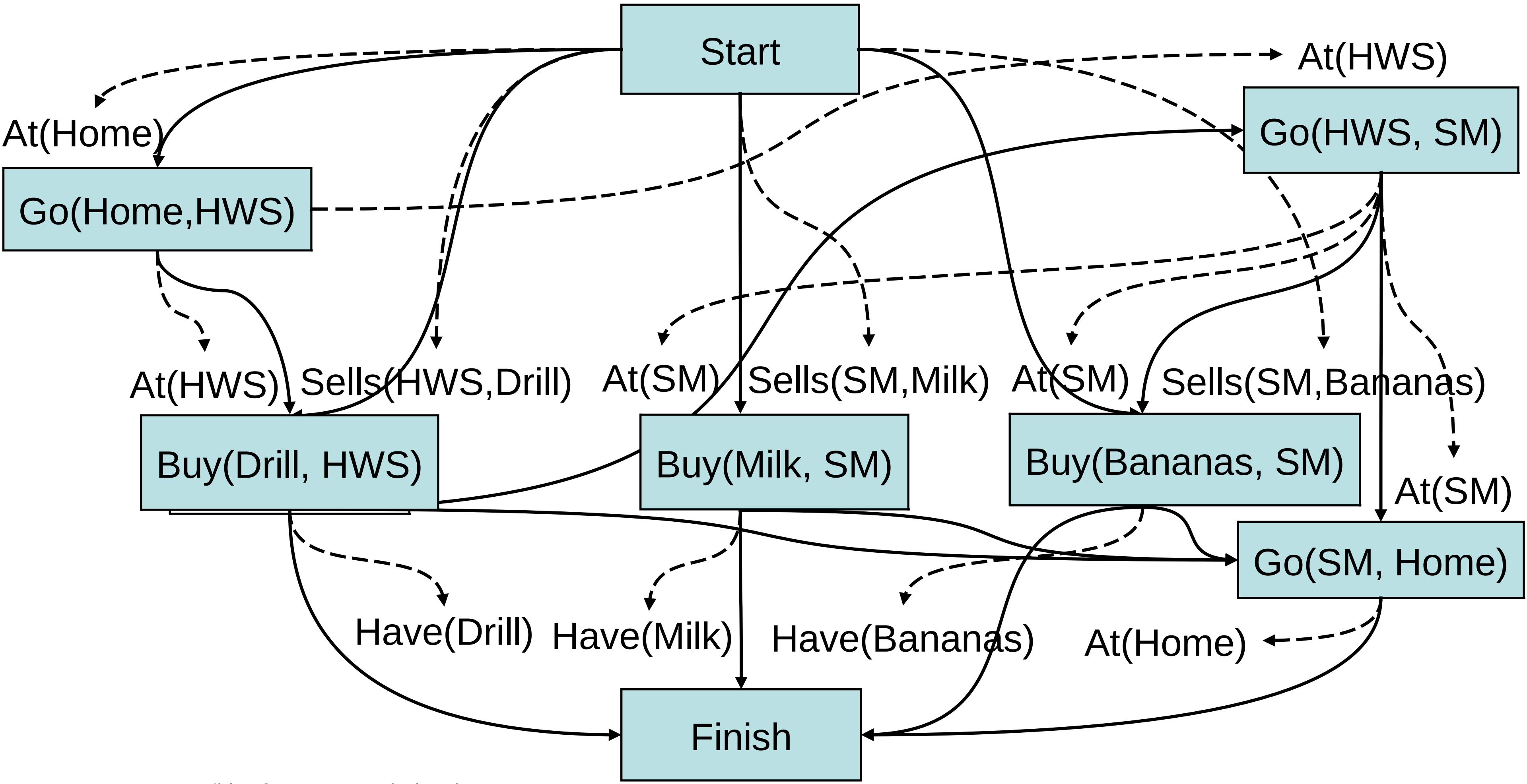
Example (continued)

- 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
- We're done!



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:

Backward state-space search:

… → act1 → act2 → *goal*

… → act2 → act1 → *goal*

- 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?

… → act1 → act2 → act1 → …

  - No. Sometimes we might need the same action several times in different states of the world
    - » Example on next slide

Example

- 3-digit binary counter starts at 000, want to get to 111

  s_0 = {d_3=0, d_2=0, d_1=0}, i.e., 000

  g = {d_3=1, d_2=1, d_1=1}, i.e., 111

- Operators to increment the counter by 1:

  incr-xx0-to-xx1

  Precond: *d_1=0*

  Effects: d_1=1

  incr-x01-to-x10

  Precond: d_2=0, d_1=1

  Effects: d_2=1, d_1=0

  incr-011-to-100

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

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

- Plan:

		d_3	d_2	d_1
initial state:		0	0	0
incr-xx0-to-xx1	→	0	0	1
incr-x01-to-x10	→	0	1	0
incr-xx0-to-xx1	→	0	1	1
incr-011-to-100	→	1	0	0
incr-xx0-to-xx1	→	1	0	1
incr-x01-to-x10	→	1	1	0
incr-xx0-to-xx1	→	1	1	1

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