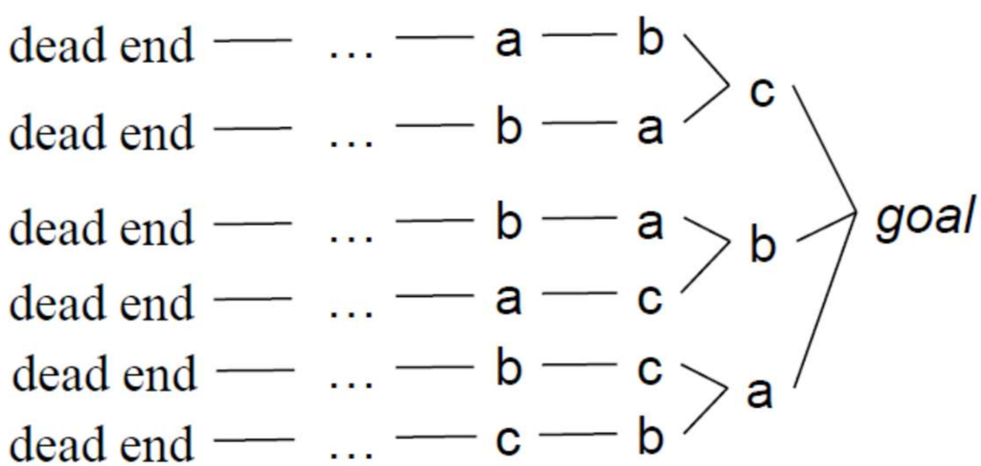
State Space vs. Plan Space

- Planning in the state space:
  - sequence of actions, from the initial state to the goal state
- Planning in the plan space:
  - Sequence of plan transformations, from an initial plan to the final one

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

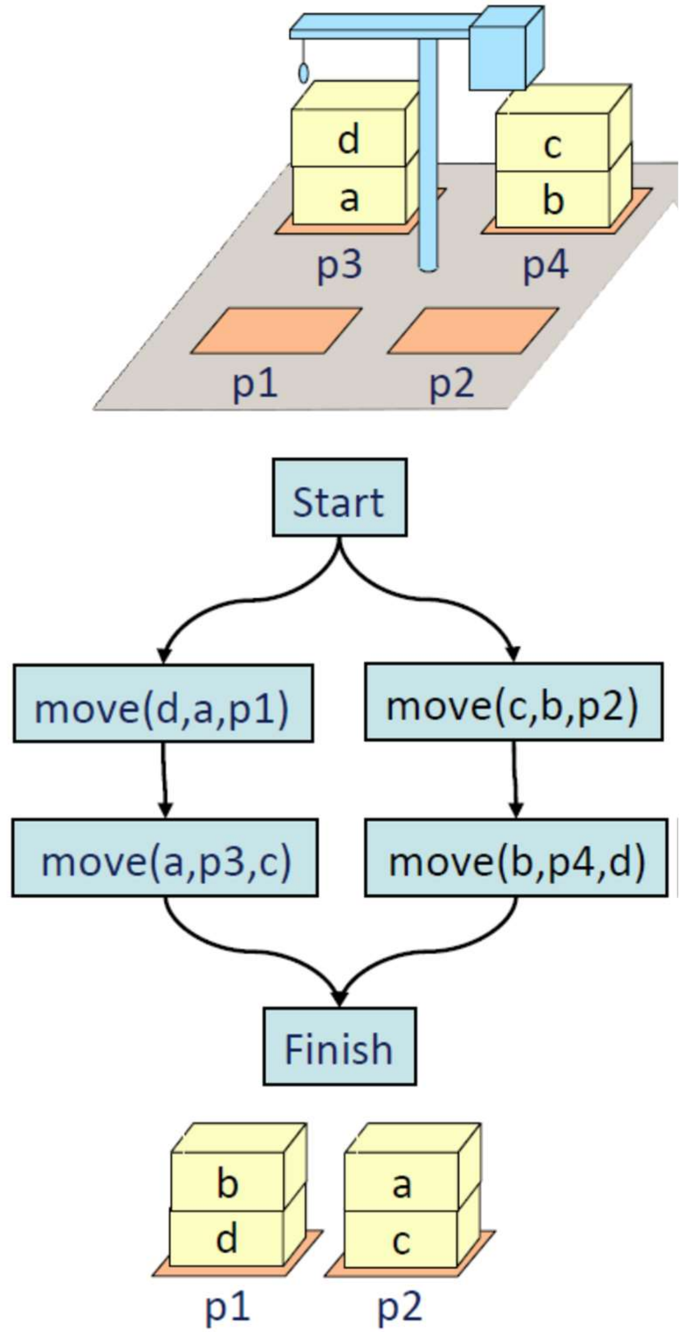


- *Least-commitment strategy:* don't commit to orderings, instantiations, etc., until necessary

Plan-Space Planning (Chapter 5)

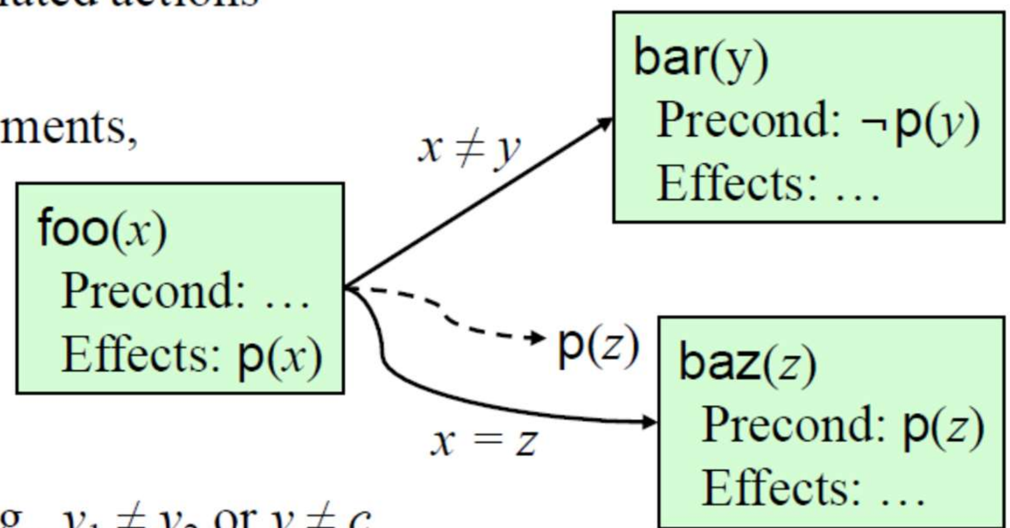
- Decompose sets of goals into the individual goals
- Plan for them separately
  - Bookkeeping info to detect and resolve interactions
- Produce a partially ordered plan that retains as much flexibility as possible

- The Mars rovers used a temporal-planning extension of this



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

Plan-State Search

- Search space is set of ***partial plans***
- Plan is tuple $<A, O, B>$
  - A: Set of ***actions***, of the form $(a_i : Op_j)$
  - O: Set of ***orderings***, of the form $(a_i < a_j)$
  - B: Set of ***bindings***, of the form $(v_i = C)$, $(v_i \neq C)$, $(v_i = v_j)$ or $(v_i \neq v_j)$
- Initial plan:
  - $<\{start, finish\}, \{start < finish\}, \{\}>$
  - *start* has no preconditions; Its effects are the initial state
  - *finish* has no effects; Its preconditions are the goals

State-Space vs Plan-Space

Planning problem

Find a sequence of actions that make instance of the goal true

Nodes in search space

Standard search: node = concrete world state

Planning search: node = partial plan

(Partial) Plan consists of

- Set of operator applications S_i
- Partial (temporal) order constraints $S_i \prec S_j$
- Causal links $S_i \xrightarrow{c} S_j$

  Meaning: "S_i achieves $c \in precond(S_j)$" (record purpose of steps)

Search in the Plan-Space

Operators on partial plans

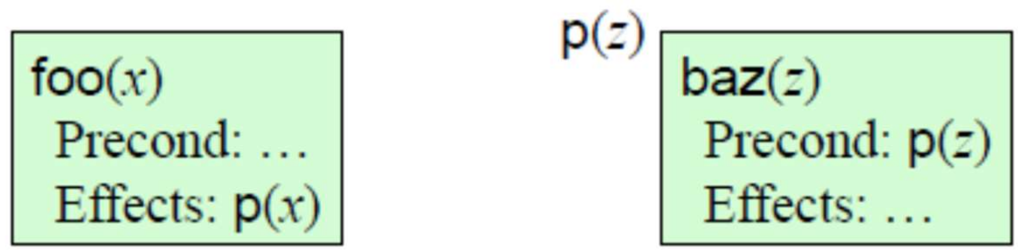
- add an action and a causal link to achieve an open condition
- add a causal link from an existing action to an open condition
- add an order constraint to order one step w.r.t. another

Open condition

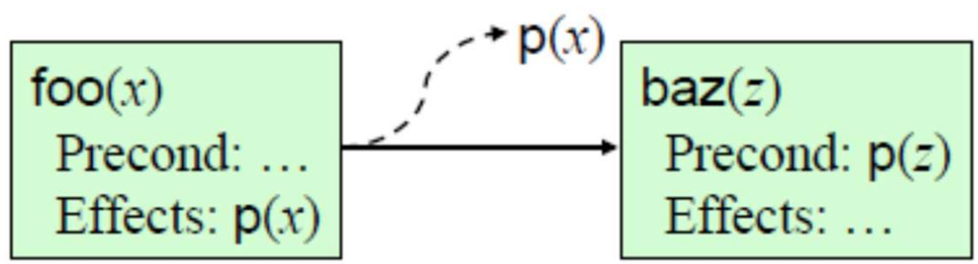
A precondition of an action not yet causally linked

Flaws: 1. Open Goals

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

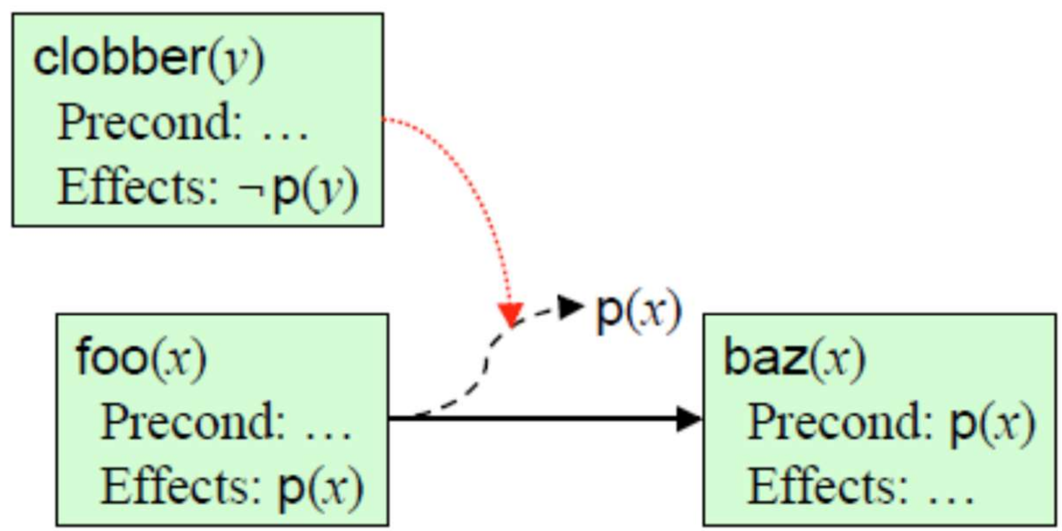


- Resolving the flaw:
  - Find an action b
    - (either already in the plan, or insert it)
  - that can be used to establish p
    - can precede a and produce p
  - Instantiate variables and/or constrain variable bindings
  - Create a causal link



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*

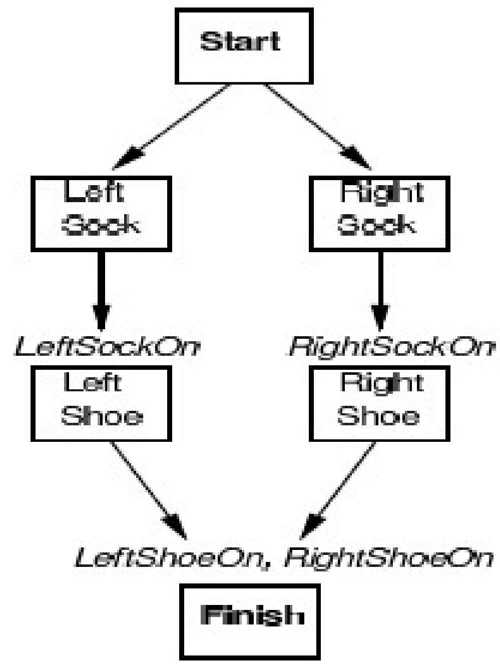


The PSP Procedure

```
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

Partially-Ordered Plans



Special steps with empty action

Start no precond, initial assumptions as effect)

Finish goal as precond, no effect

Partial-Order Plans

Complete plan

A plan is complete iff every precondition is achieved

A precondition c of a step S_j is achieved (by S_i) if

- $S_i \prec S_j$
- $c \in effect(S_i)$
- there is no S_k with $S_i \prec S_k \prec S_j$ and $\neg c \in effect(S_k)$
  (otherwise S_k is called a clobberer or threat)

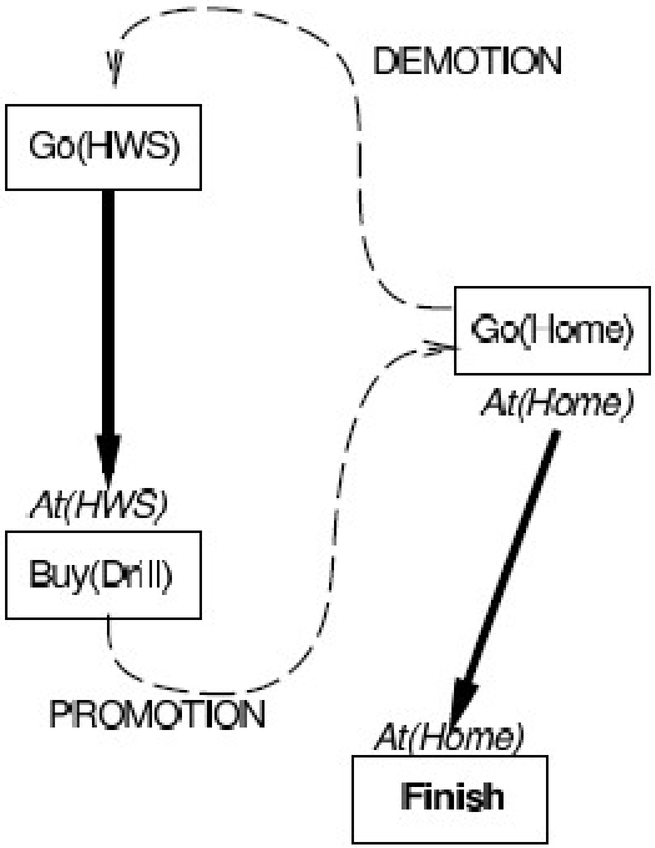
Clobberer / threat

A potentially intervening step that destroys the condition achieved by a causal link

Partial-Order Plans

Example

$Go(Home)$ **clobbers** $At(HWS)$



Demotion

Put before $Go(HWS)$

Promotion

Put after $Buy(Drill)$

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)

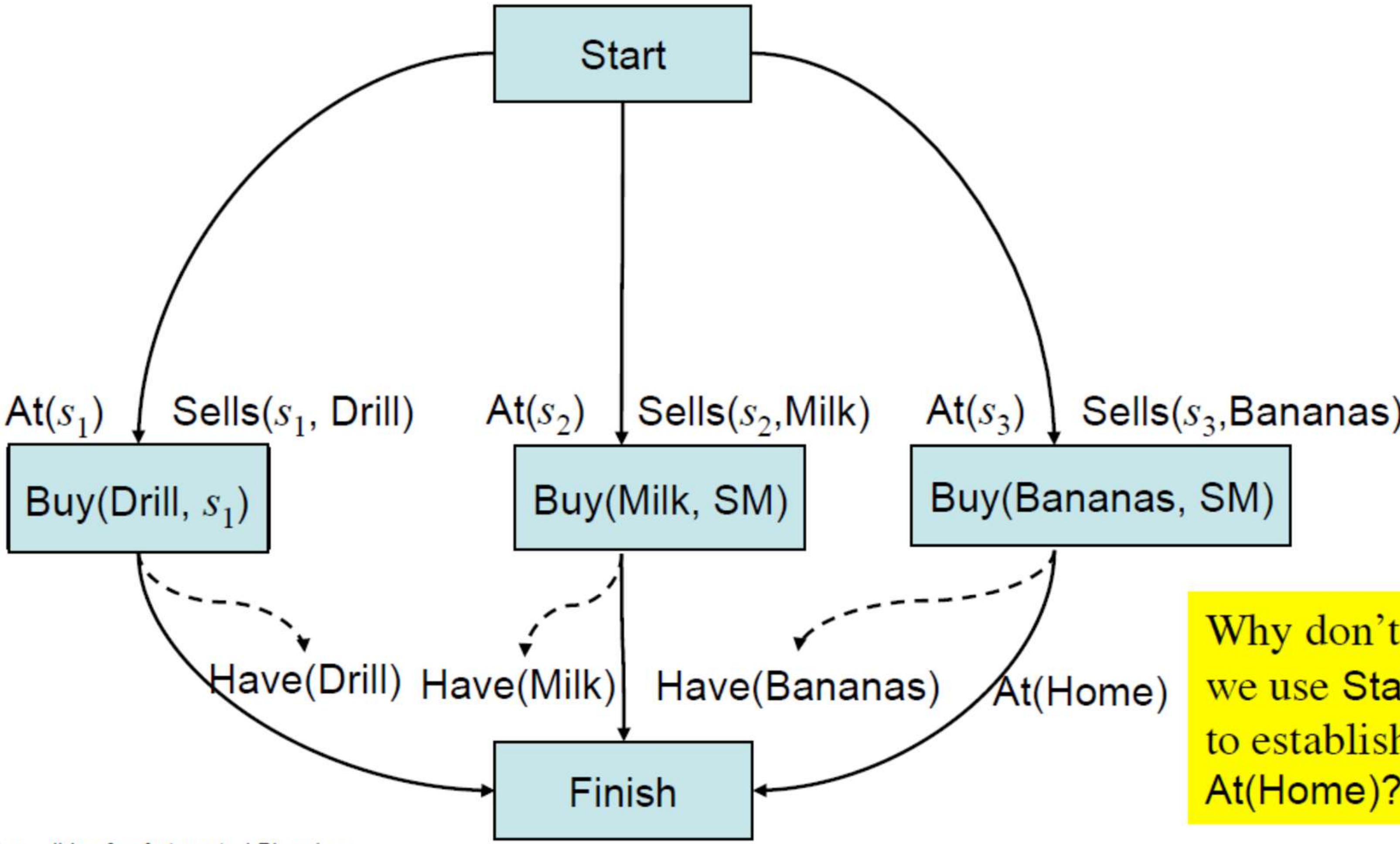
    Precond: At(s), Sells(s,p)

    Effects: Have(p)

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

Example (continued)

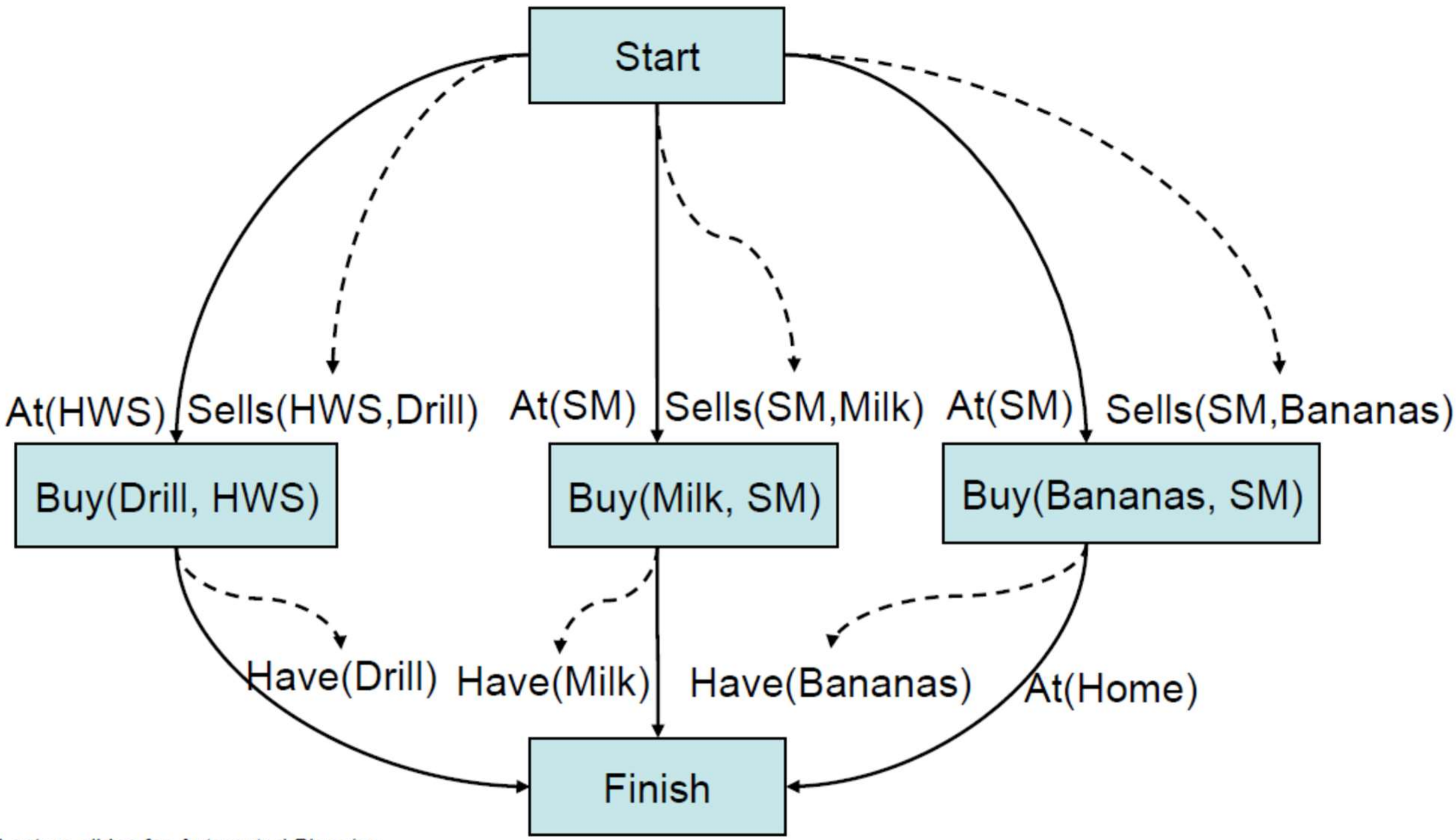
- The first three refinement steps
  - These are the only possible ways to establish the Have preconditions



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

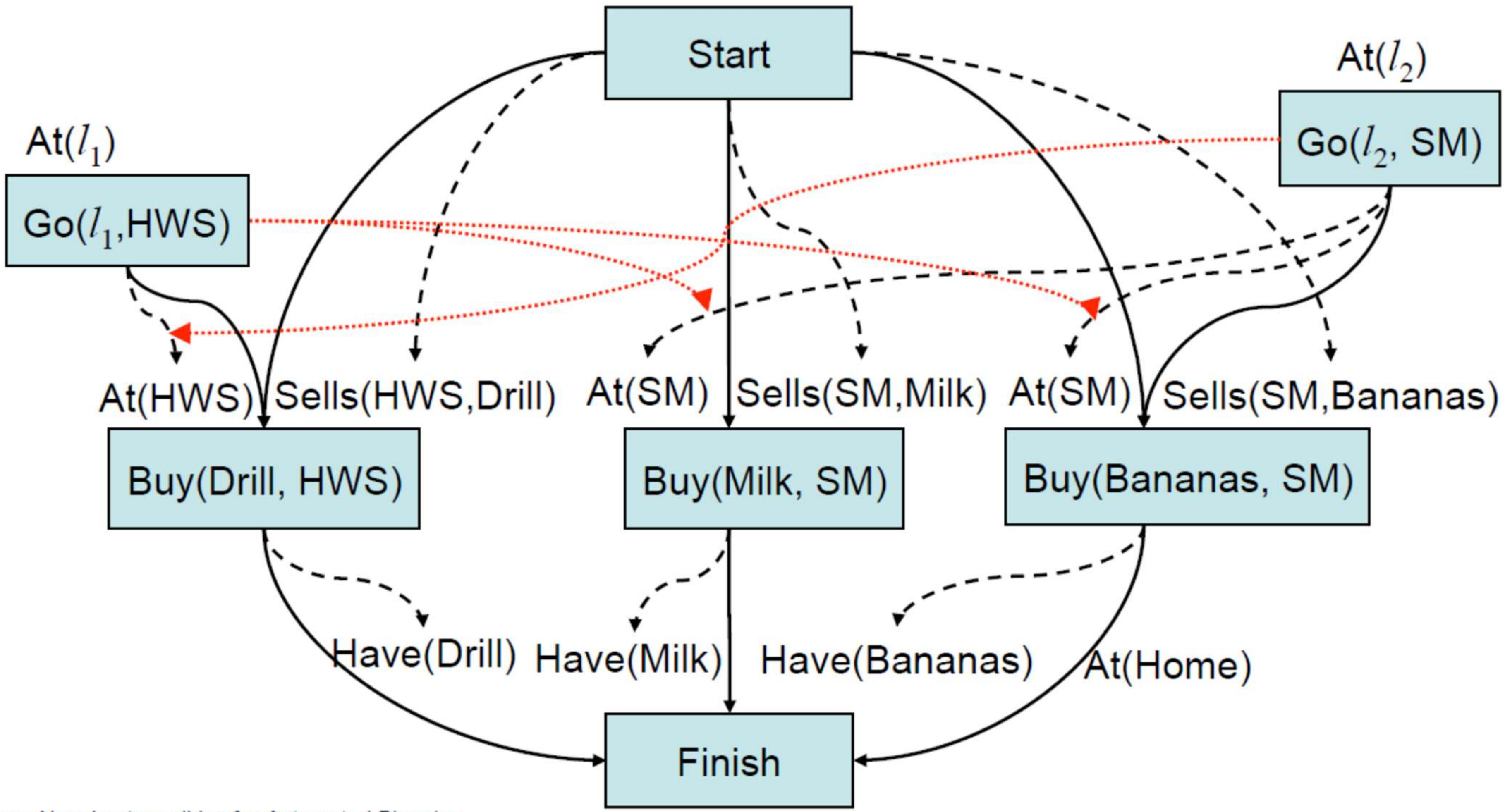
Example (continued)

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



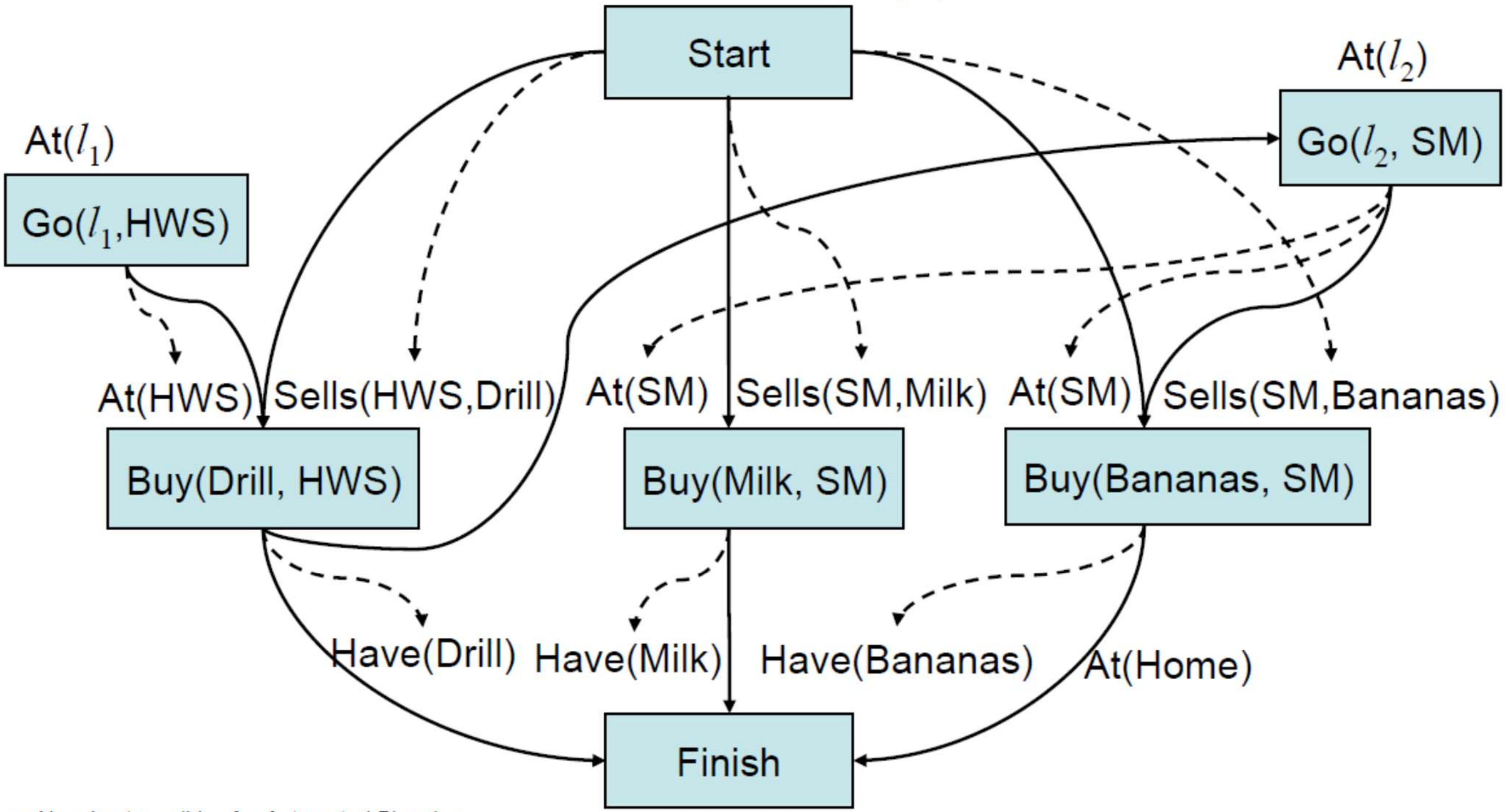
Example (continued)

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



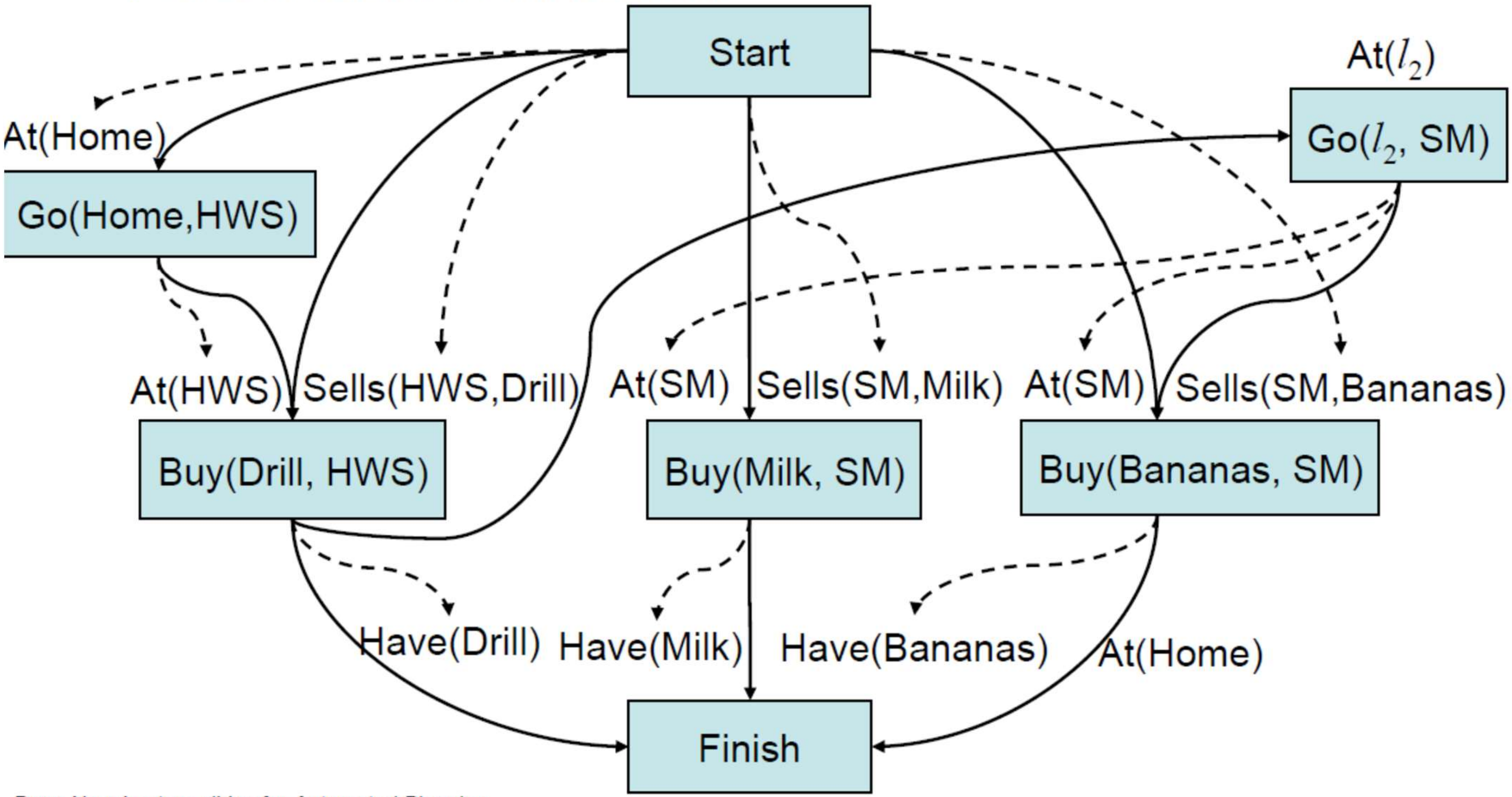
Example (continued)

- 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?)



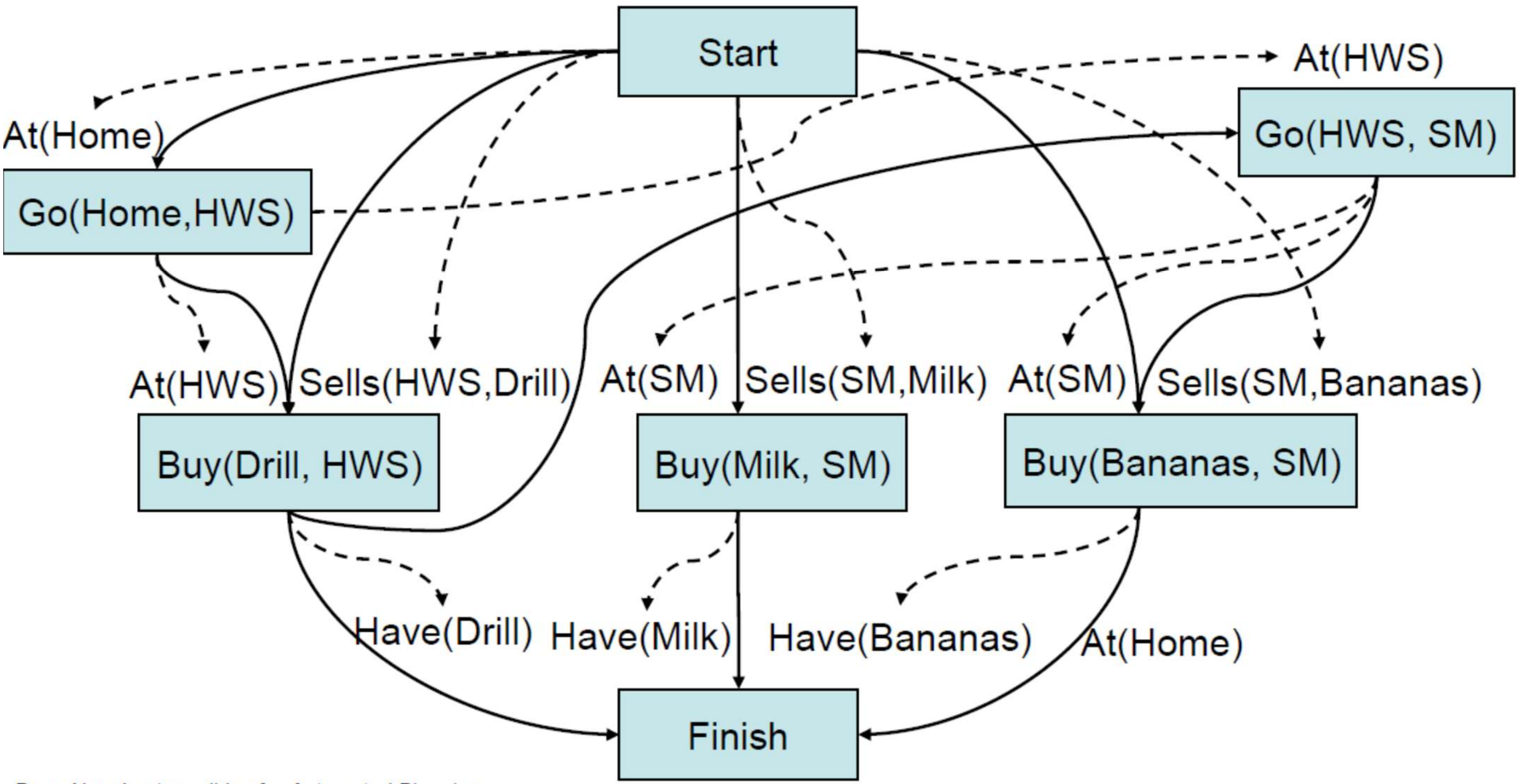
Example (continued)

- 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?



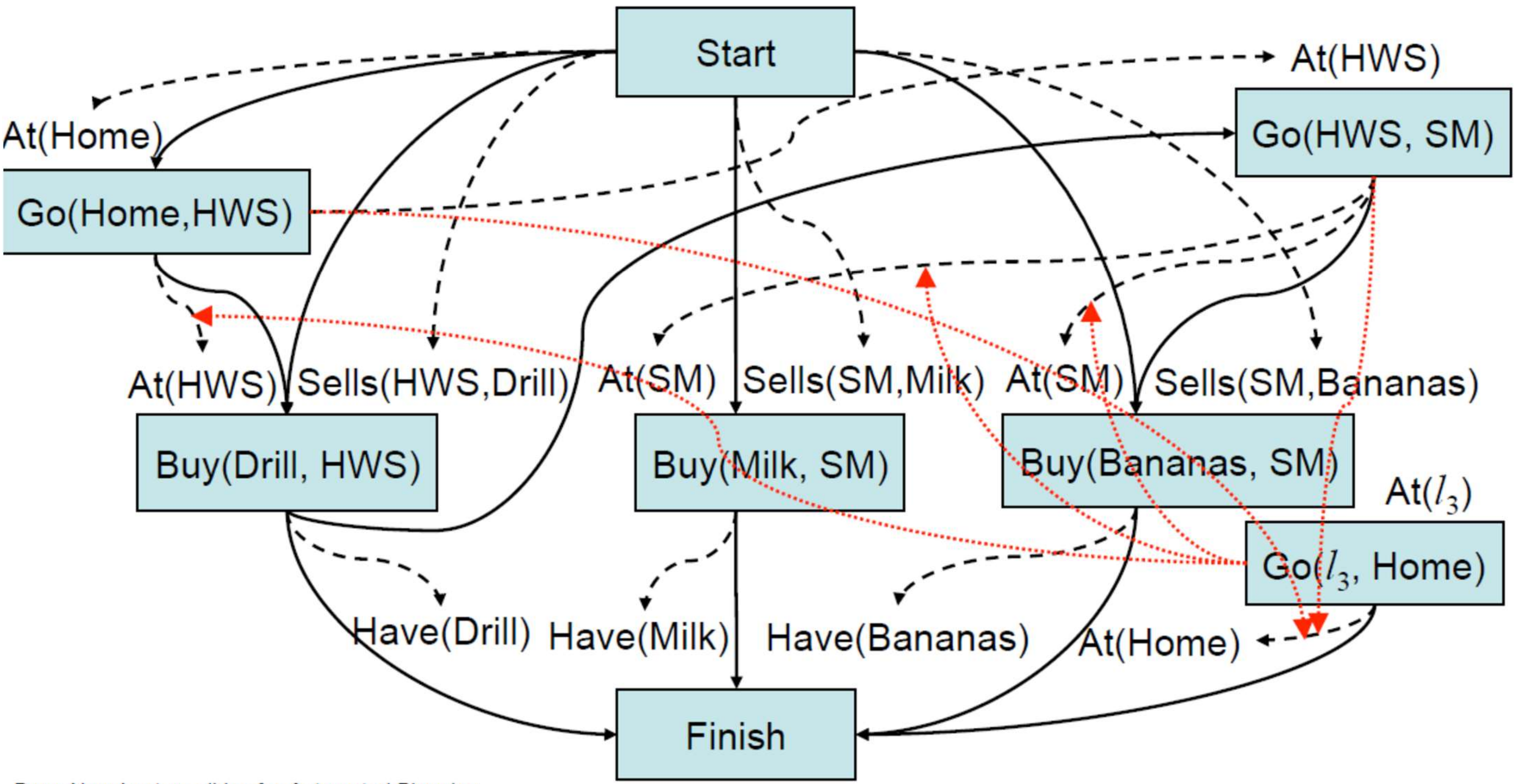
Example (continued)

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



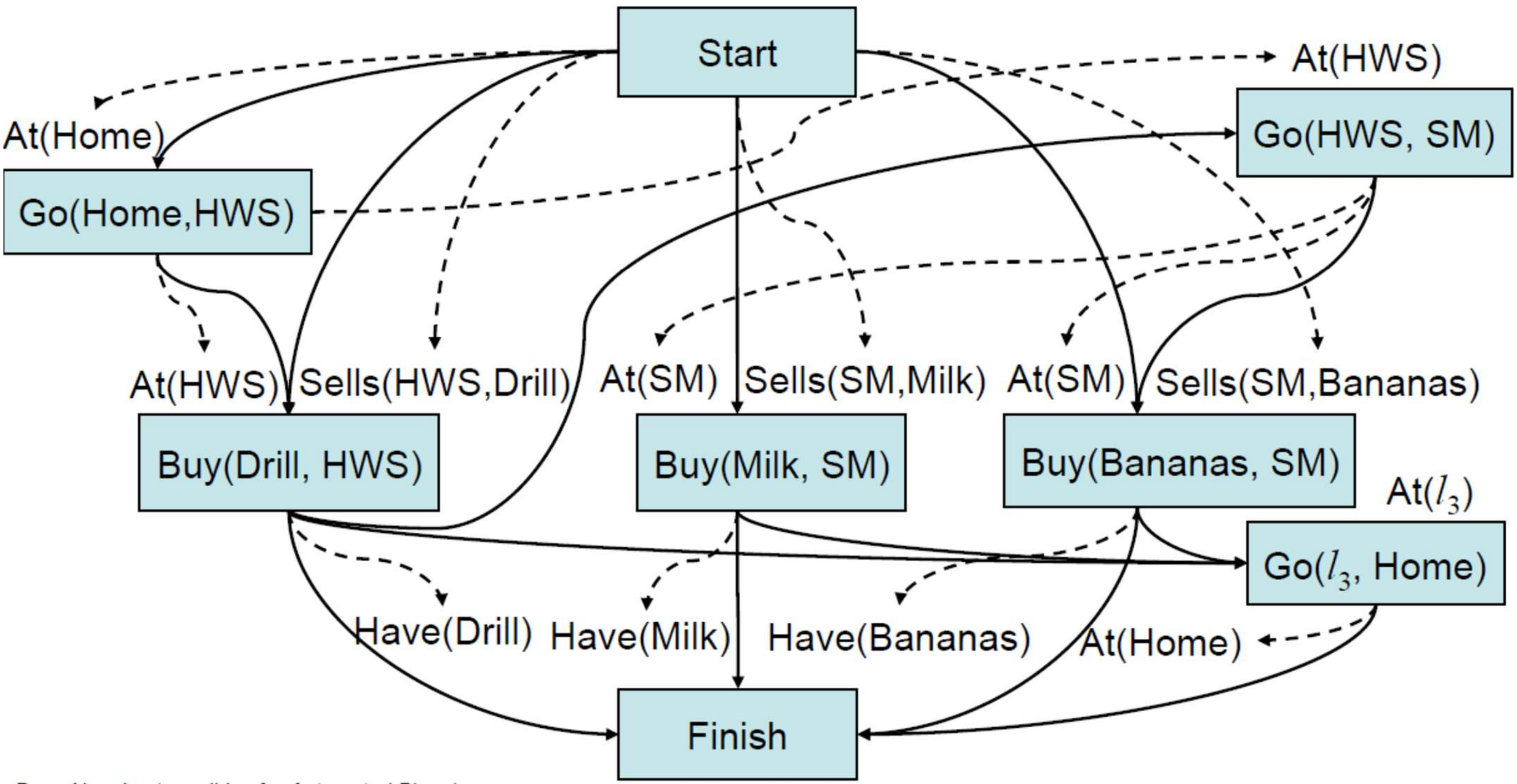
Example (continued)

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



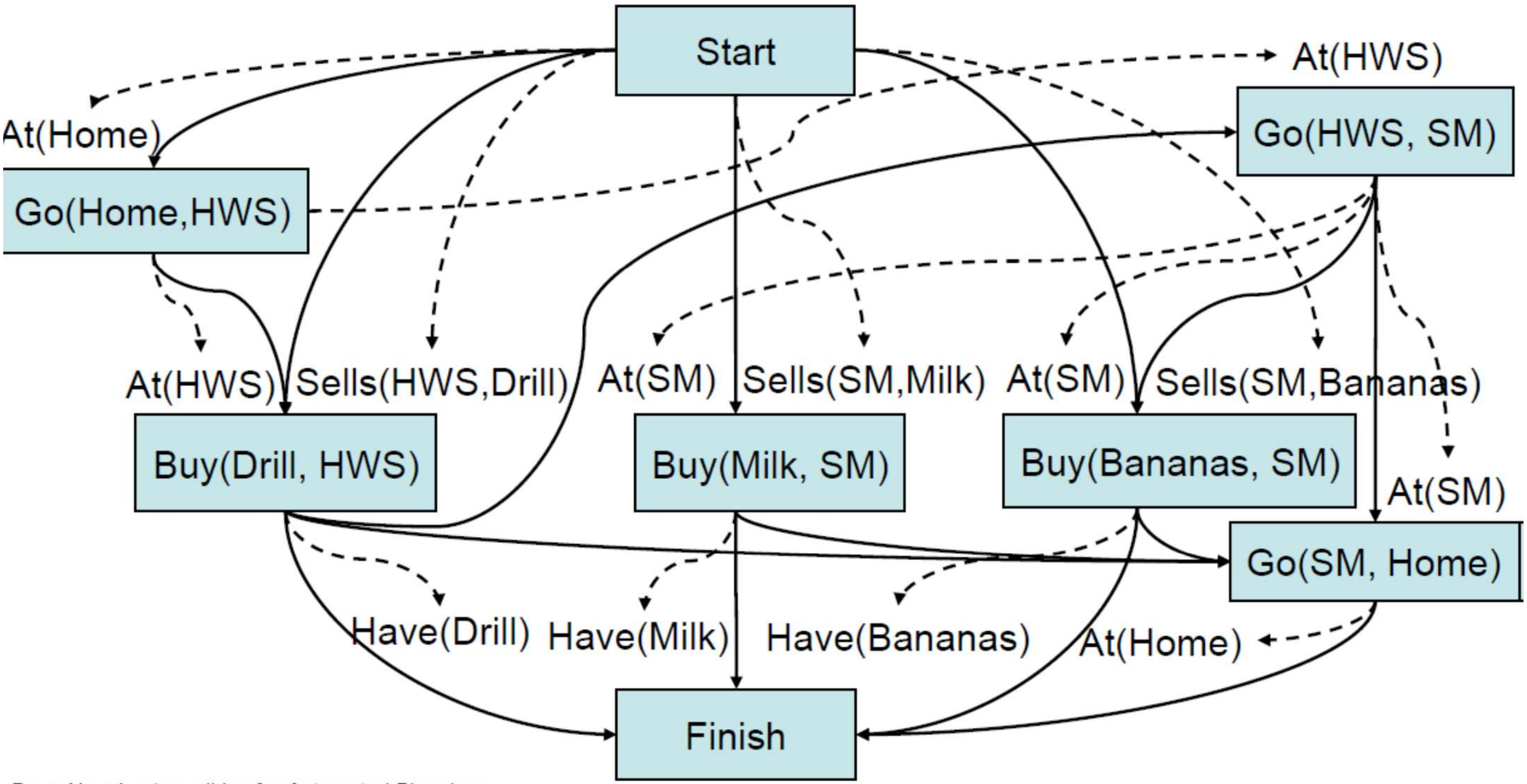
Example (continued)

- 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

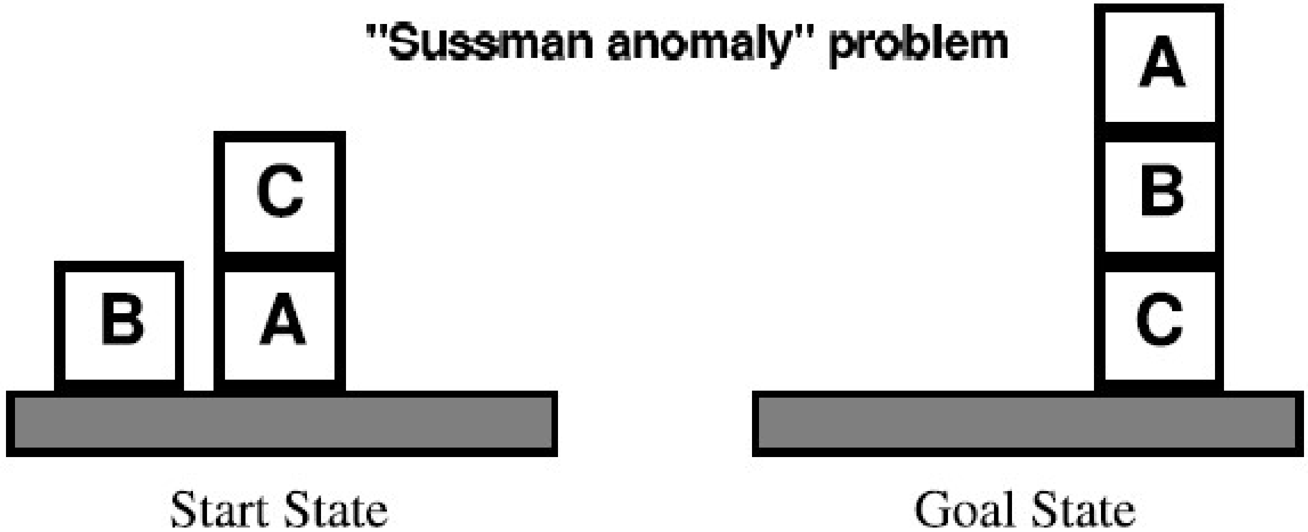
- Establish At(l_3) with l_3=SM
- We're done!



General Approach

- General Approach
  - Find unachieved precondition
    - Add new action *or* link to existing action
  - Determine if conflicts occur
    - Previously achieved precondition is "clobbered"
    - Fix conflicts (reorder, bind, …)
- Partial-order planning can easily (and optimally) solve blocks world problems that involve goal interactions (e.g., the "Sussman Anomaly" problem)

Blocks World



Clear(x) On(x,z) Clear(y)

PutOn(x,y)

~On(x,z) ~Clear(y)
Clear(z) On(x,y)

Clear(x) On(x,z)

PutOnTable(x)

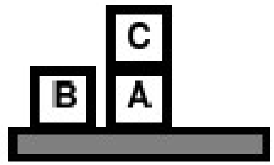
~On(x,z) Clear(z) On(x,Table)

+ several inequality constraints

Blocks World

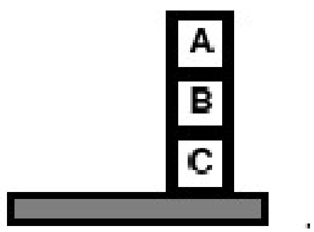
START

On(C,A) On(A,Table) Cl(B) On(B,Table) Cl(C)

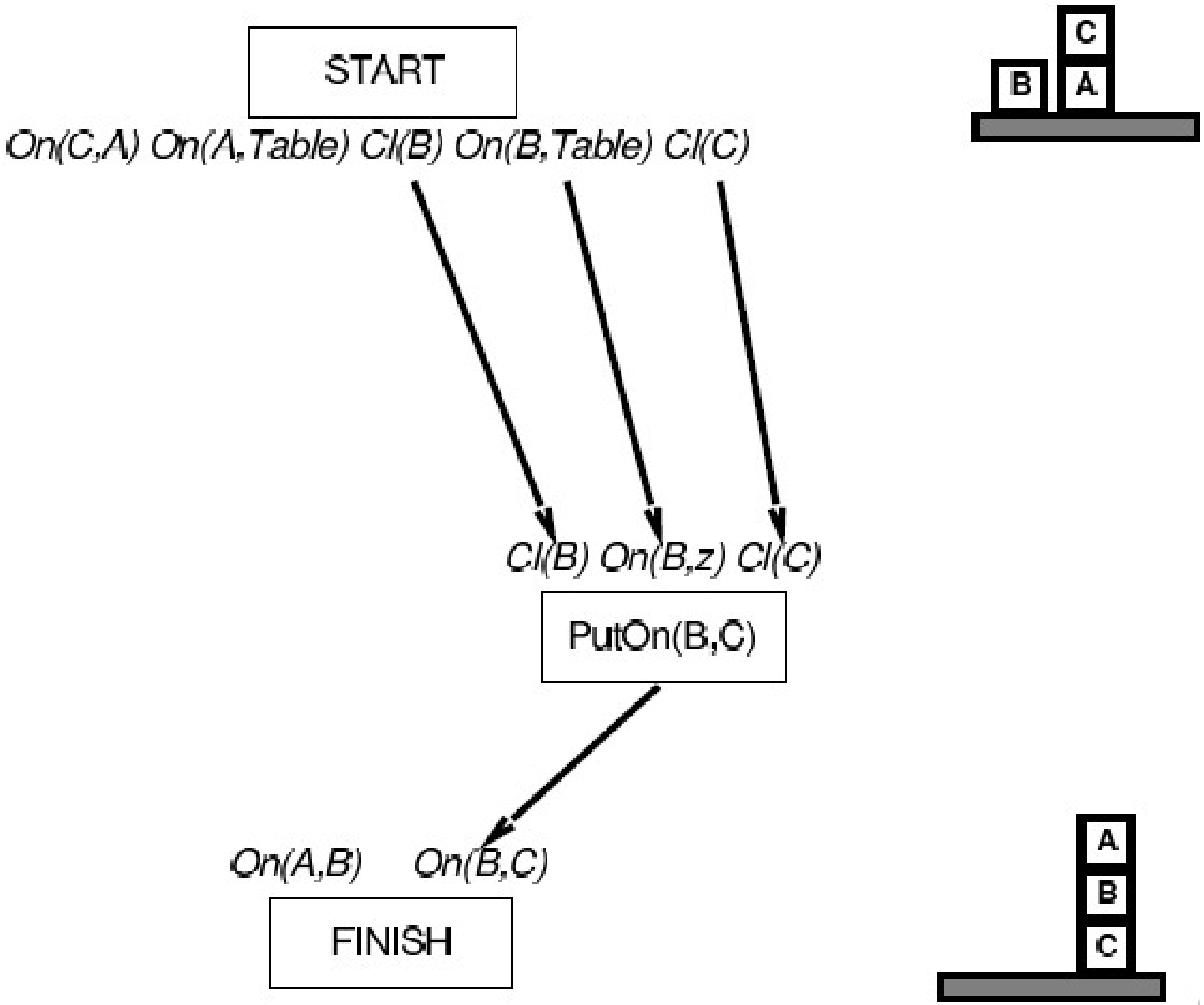


On(A,B)   *On(B,C)*

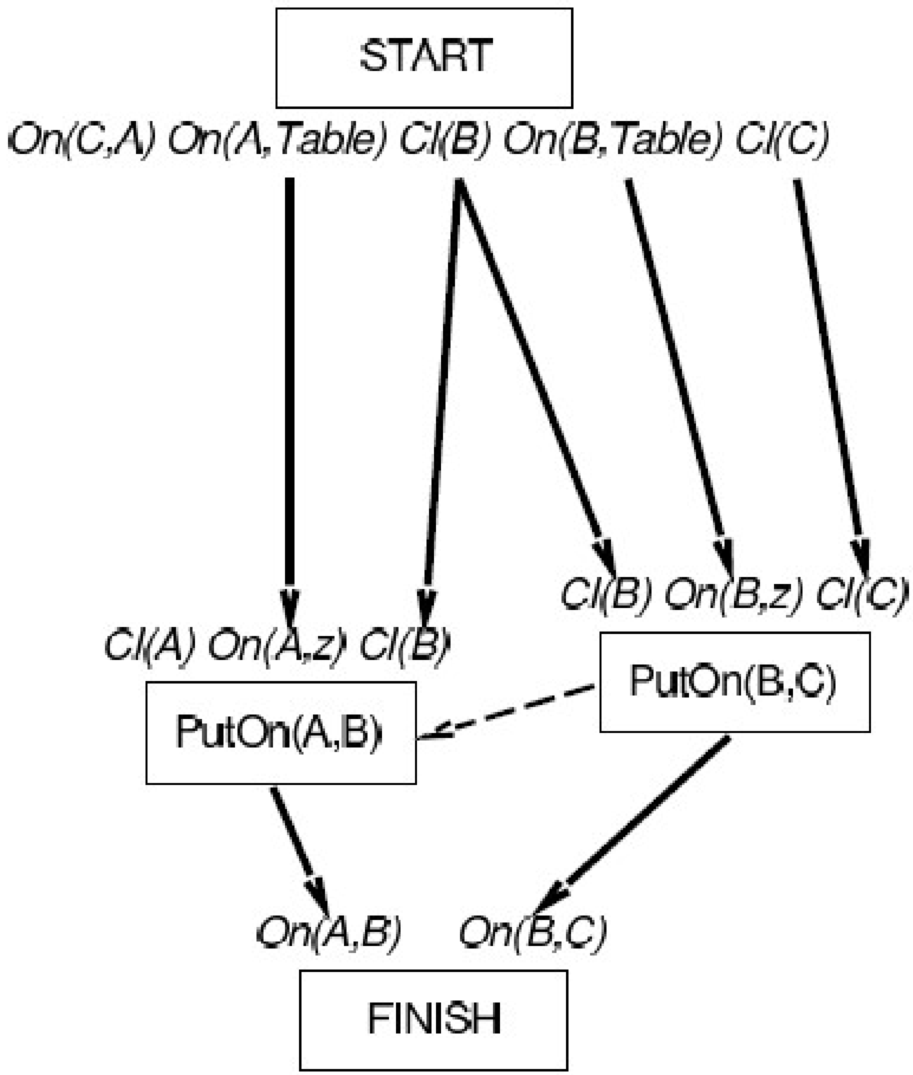
FINISH

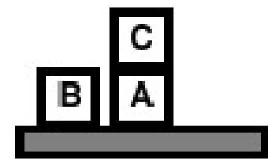


Blocks World

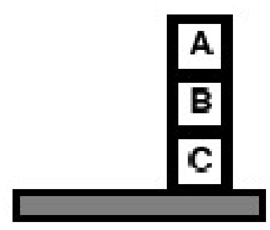


Blocks World

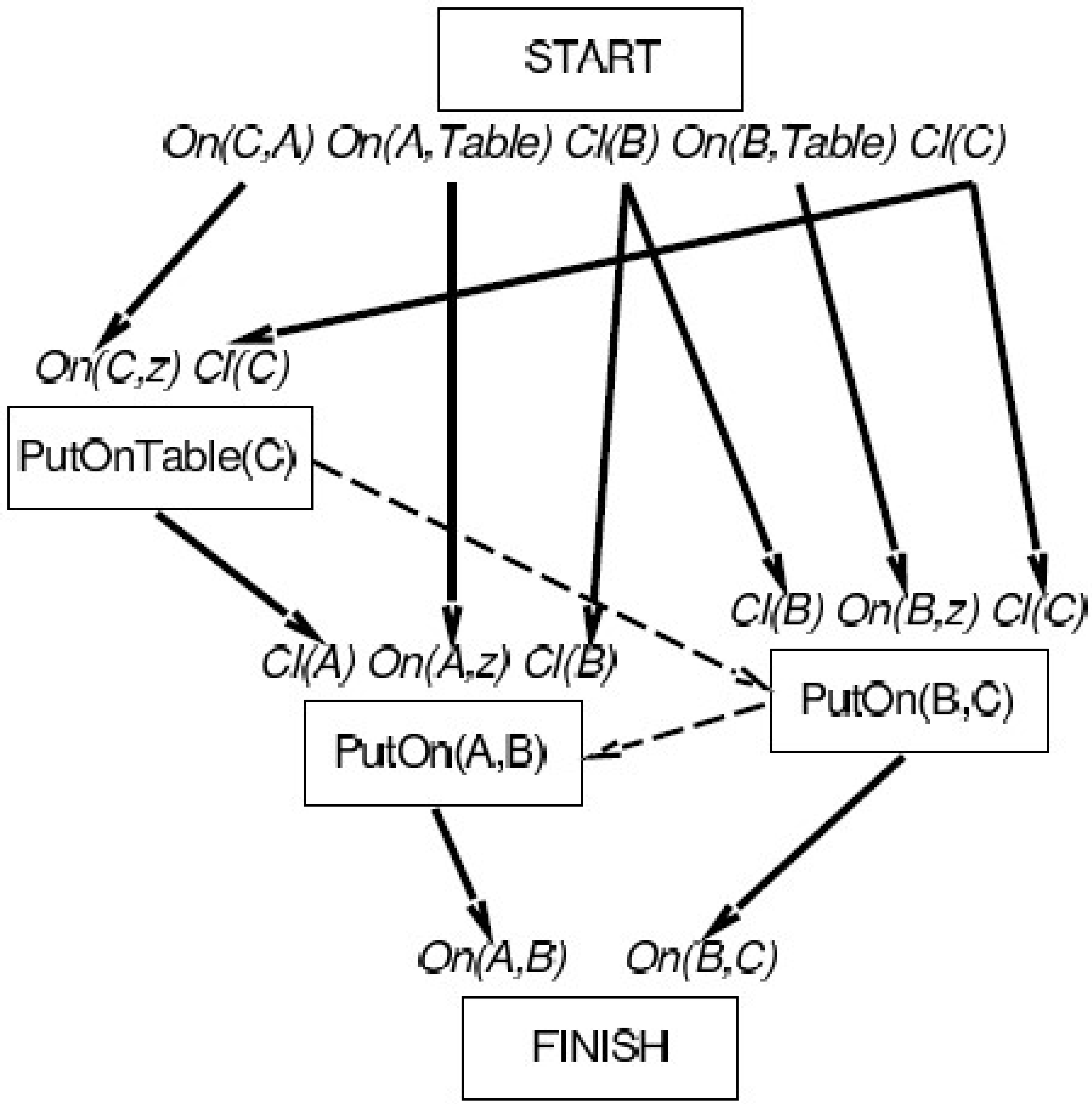


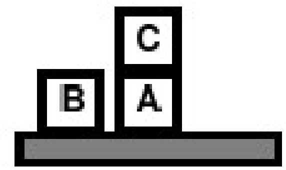


PutOn(A,B)
clobbers Cl(B)
=> order after
PutOn(B,C)



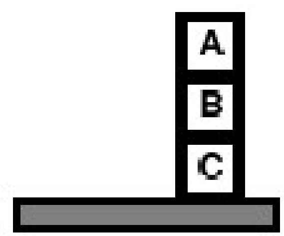
Blocks World





PutOn(A,B)
clobbers Cl(B)
=> order after
PutOn(B,C)

PutOn(B,C)
clobbers Cl(C)
=> order after
PutOnTable(C)



Least Commitment

- Basic Idea
  - *Make choices that are only relevant to solving the current part of the problem*

- Least Commitment Choices
  - **Orderings**: Leave actions unordered, unless they must be sequential
  - **Bindings**: Leave variables unbound, unless needed to unify with conditions being achieved
  - **Actions**: Usually not subject to "least commitment"
- Refinement
  - Only *add* information to the current plan
  - *Transformational* planning can remove choices

Terminology

- *Totally Ordered* Plan
  - There exists sufficient orderings *O* such that all actions in *A* are ordered with respect to each other
- *Fully Instantiated* Plan
  - There exists sufficient constraints in *B* such that all variables are constrained to be equal to some constant
- *Consistent* Plan
  - There are no contradictions in *O* or *B*
- *Complete* Plan
  - Every precondition *p* of every action a_i in *A* is *achieved*: There exists an effect of an action a_j that comes before a_i and unifies with *p*, and no action a_k that deletes *p* comes between a_j and a_i

POP-Algorithm

- **Advantages**
  - Partial order planning is *sound* and *complete*
  - Typically produces *optimal* solutions (plan length)
  - Least commitment may lead to shorter search times

- **Disadvantages**
  - Significantly more complex algorithms (higher *per-node* cost)
  - Hard to determine what is true in a state
  - Larger search space (**infinite**!)

Plan Monitoring

Execution monitoring

Failure: Preconditions of remaining plan not met

Action monitoring

Failure: Preconditions of next action not met

(or action itself fails, e.g., robot bump sensor)

Consequence of failure

Need to replan

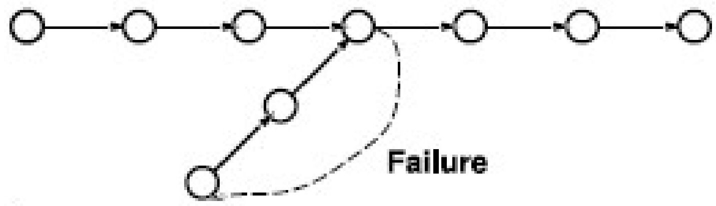
Replanning

Simplest

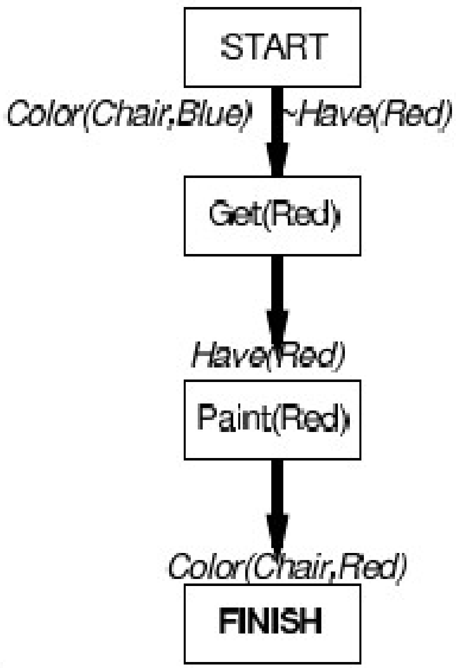
On failure, replan from scratch

Better

Plan to get back on track by reconnecting to best continuation



Replanning



PRECONDITIONS	FAILURE RESPONSE
none	N/A
Have(Red)	Fetch more red
Color(Chair,Red)	Repaint

Preconditions for the rest of the plan