Chapter 5
Plan-Space Planning

Dana S. Nau
University of Maryland

5:10 PM    February 6, 2012
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

```
foo(x)  
Precond: …  
Effects: p(x)  

p(z)  

baz(z)  
Precond: p(z)  
Effects: …
```
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(\(\pi\))
\[
\text{flaws} \leftarrow \text{OpenGoals}(\pi) \cup \text{Threats}(\pi) \\
\text{if } \text{flaws} = \emptyset \text{ then return}(\pi) \\
\text{select any flaw } \phi \in \text{flaws} \\
\text{resolvers} \leftarrow \text{Resolve}(\phi, \pi) \\
\text{if } \text{resolvers} = \emptyset \text{ then return}(\text{failure}) \\
\text{nondeterministically choose a resolver } \rho \in \text{resolvers} \\
\pi' \leftarrow \text{Refine}(\rho, \pi) \\
\text{return}(\text{PSP}(\pi'))
\]

- 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 $\pi$ as its argument
  - Initial plan: Start, Finish, and an ordering constraint

\[\text{Effects: } \text{At(Home), Sells(HWS,Drill), Sells(SM,Milk), Sells(SM,Bananas)}\]

\[\text{Precond: } \text{Have(Drill), Have(Milk), Have(Bananas), At(Home)}\]
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

```plaintext
- Buy(Drill, HWS)
- Sells(HWS, Drill)
- At(HWS)
- Sells(SM, Milk)
- At(SM)
- Sells(SM, Bananas)
- At(SM)
- Have(Drill)
- Have(Milk)
- Have(Bananas)
- At(Home)
- Buy(Bananas, SM)
- Buy(Milk, SM)
- Start
- Finish
```
Example (continued)

- Two more refinements: the only ways to establish $\text{At}(\text{HWS})$ and $\text{At}(\text{SM})$
  - This time, several threats occur
Example (continued)

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

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

Graph representation:

- Start
- At(Home)
- At(HWS)
- Sells(HWS, Drill)
- Buy(Drill, HWS)
- Have(Drill)
- Finish
- At(Home)
- At(SM)
- Sells(SM, Milk)
- Buy(Milk, SM)
- Have(Milk)
- At(SM)
- Sells(SM, Bananas)
- Buy(Bananas, SM)
- Have(Bananas)
- Go(Home, HWS)
- Go(HWS)
- At(HWS)
- Sells(HWS, Drill)
- Buy(Drill, HWS)
- Have(Drill)
- Go(l_2, SM)
- At(l_2)
Example (continued)

- Nondeterministic choice: how to establish \( \text{At}(l_2) \)?
  - We’ll do it from \( \text{Go}(\text{Home}, \text{HWS}) \), with \( l_2 = \text{HWS} \)
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 $\text{At}(\text{SM})$ and $\text{At}(\text{HWS})$, make them precede $\text{Go}(l_3, \text{Home})$
  - This also removes the other threats
Final Plan

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

Final Plan:

1. Establish At($l_3$) with $l_3=$SM
2. Go(SM, Home)
3. At(Home)
4. Go(Home, HWS)
5. At(HWS)
6. Sells(HWS, Drill)
7. Buy(Drill, HWS)
8. Sells(SM, Milk)
9. Buy(Milk, SM)
10. Sells(SM, Bananas)
11. Buy(Bananas, SM)
12. Sells(HWS, Drill)
13. Buy(Drill, HWS)
14. Have(Bananas)
15. At(HWS)
16. Go(HWS, SM)
17. Sells(SM, Milk)
18. Buy(Milk, SM)
19. Sells(SM, Bananas)
20. Buy(Bananas, SM)
21. Sells(HWS, Drill)
22. Buy(Drill, HWS)
23. Have(Bananas)
24. At(HWS)
25. Go(SM, Home)
26. At(Home)
27. Have(Drill)
28. Have(Milk)
29. Have(Bananas)
30. At(HWS)
31. Go(Home, HWS)
32. At(Home)
33. Finish
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?
  - 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
  
  \[s_0 = \{d_3=0, d_2=0, d_1=0\}, \text{ i.e., } 000\]
  
  \[g = \{d_3=1, d_2=1, d_1=1\}, \text{ i.e., } 111\]

- 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:
  
  \[
  \begin{array}{ccc}
    \text{initial state:} & d_3 & d_2 & d_1 \\
    \text{ incr-xx0-to-xx1 } & 0 & 0 & 1 \\
    \text{ incr-x01-to-x10 } & 0 & 1 & 0 \\
    \text{ incr-xx0-to-xx1 } & 0 & 1 & 1 \\
    \text{ incr-011-to-100 } & 1 & 0 & 0 \\
    \text{ incr-xx0-to-xx1 } & 1 & 0 & 1 \\
    \text{ incr-x01-to-x10 } & 1 & 1 & 0 \\
    \text{ incr-xx0-to-xx1 } & 1 & 1 & 1 \\
  \end{array}
  \]
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