Chapter 5
Plan-Space Planning
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

![Diagram](image)

● *Least-commitment strategy:* don’t commit to orderings, instantiations, etc., until necessary
Partial Order Plans:

Total Order Plans:
Outline

• Basic idea
• Open goals
• Threats
• The PSP algorithm
• Long example
• Comments
Plan Space
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 \( v = c \)) 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
Flaw:
- 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
- Create a causal link
Threat

- Flaw: a deleted-condition interaction
  - Action $a$ establishes a condition (e.g., $p(x)$) for action $b$
  - Another action $c$ is capable of deleting this condition $p(x)$
- Resolving the flaw:
  - Impose a constraint to prevent $c$ from deleting $p(x)$
- Three possibilities:
  - Make $b$ precede $c$
  - Make $c$ precede $a$
  - Constrain variable(s) to prevent $c$ from deleting $p(x)$
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
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)
Example (continued)

- Initial plan

```
Start

At(Home), Sells(HWS, Drill), Sells(SM, Milk), Sells(SM, Bananas)

Have(Drill), Have(Milk), Have(Bananas), At(Home)

Finish
```
Example (continued)

- The only possible ways to establish the “Have” preconditions

---

Start

- $\text{At}(s_1), \text{Sells}(s_1, \text{Drill})$
  - $\text{Buy}(\text{Drill}, s_1)$

- $\text{At}(s_2), \text{Sells}(s_2, \text{Milk})$
  - $\text{Buy}(\text{Milk}, s_2)$

- $\text{At}(s_3), \text{Sells}(s_3, \text{Bananas})$
  - $\text{Buy}(\text{Bananas}, s_2)$

Finish

$\text{Have(Drill), Have(Milk), Have(Bananas), At(Home)}$
The only possible way to establish the “Sells” preconditions.
Example (continued)

- The only ways to establish At(HWS) and At(SM)
  - Note the threats
Example (continued)

- To resolve the third threat, make Buy(Drill) precede Go(SM)
  - This resolves all three threats
Example (continued)

- Establish $\text{At}(l_1)$ with $l_1=\text{Home}$
Example (continued)

- Establish $\text{At}(l_2)$ with $l_2=\text{HWS}$
Example (continued)

- Establish At(Home) for Finish
Example (continued)

- Constrain Go(Home) to remove threats to At(SM)
Final Plan

- Establish At($I_3$) with $I_3$=SM

```
Establish At($I_3$) with $I_3$=SM

At(Home) Go(Home,HWS) At(HWS), Sells(HWS,Drill) Buy(Drill,HWS) Have(Drill), Have(Milk), Have(Bananas), At(Home)

Go(HWS,SM) At(HWS) At(SM), Sells(SM,Bananas) Buy(Bananas,SM) Go(SM,Home)

At(SM), Sells(SM,Milk) Buy(Milk,SM) Have(Milk), Have(Bananas), At(Home)

At(SM) Go(SM,Home)

Finish
```
Comments

- 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?
  - \[ ... \rightarrow \text{go(b,a)} \rightarrow \text{go(a,b)} \rightarrow \text{go(b,a)} \rightarrow \text{at(a)} \]

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 paths of length $> n$, where $n = |\{\text{all possible states}\}|$
  - This doesn’t help very much

- There’s not yet a good pruning technique for plan-space planning

- Iterative deepening depth first search
POP algorithm

POP((A, O, L), agenda, PossibleActions):
1. If agenda is empty, return (A, O, L)
2. Pick (Q, An) from agenda
3. Ad = \textit{choose} an action that adds Q.
   a. If no such action exists, \textit{fail}.
   b. Add the link Ad $\xrightarrow{Q}$ Ac to L and the ordering Ad < Ac to O
   c. If Ad is new, add it to A.
4. Remove (Q, An) from agenda. If Ad is new, for each of its preconditions P add (P, Ad) to agenda.
5. For every action At that threatens any link Ap $\xrightarrow{Q}$ Ac
   1. \textit{Choose} to add At < Ap or Ac < At to O.
   2. If neither choice is consistent, \textit{fail}.
6. POP((A, O, L), agenda, PossibleActions)
POP in the Blocks world

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

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

FINISH
POP in the Blocks world
POP in the Blocks world
POP in the Blocks world
Conditional Effects & Disjunctions

(defun (operator move)
  :parameters (?b ?x ?y)
  :precondition (and (on ?b ?x) (clear ?b) (clear ?y)
  :effect (and (on ?b ?y) (not (on ?b ?x))
            (clear ?x) (not (clear ?y))))

Figure 14: Variables, codesignation (=), and noncodesignation (≠) constraints allow specification of more general action schemata.

(defun (operator move)
  :parameters (?b ?x ?y)
  :precondition (and (on ?b ?x) (clear ?b) (clear ?y)
  :effect (and (on ?b ?y) (not (on ?b ?x)) (clear ?x)
            (when (≠ ?y Table) (not (clear ?y))))

Figure 15: Conditional effects allow the move operator to be used when the source or destination locations is the Table. Compare with Figure 14.
Universal Quantification

(define (operator move)
  :parameters (?b ?l ?m)
  :precondition (and (briefcase ?b) (at ?b ?l) (≠ ?m ?l))
  :effect (and (at ?b ?m)
              (not (at ?b ?l))
              (forall ((object ?x))
                (when (in ?x ?b)
                  (and (at ?x ?m) (not (at ?x ?l)))))))

Figure 17: Moving a briefcase causes all objects inside the briefcase to move as well. Describing this requires universally quantified conditional effects. The \texttt{forall} quantifies over all \(?x\) that have type \texttt{object}.
Building Universal Base

\[ \Upsilon(\Delta) = \Delta \text{ if } \Delta \text{ contains no quantifiers} \]

\[ \Upsilon(\forall_{x_1} \Delta(x)) = \Upsilon(\Delta_1) \land \ldots \land \Upsilon(\Delta_n) \]

book is \{moby, crime, dict\}

If \( \Delta \) is (forall \((\text{book } ?y)\) \((\text{in } ?y \ B)\)) then

universal base \( \Upsilon(\Delta) \) is the following conjunction:

\( (\text{and } (\text{in moby } B) (\text{in crime } B) (\text{in dict } B)) \)
Building Universal Base (Cont.)

\[
\begin{align*}
\gamma(\exists_t y \Delta(y)) &= t_1(y) \land \gamma(\Delta(y)) \\
\gamma(\forall_t x \exists_t y \Delta(x, y)) &= t_2(y_1) \land \gamma(\Delta_1) \land \ldots \land t_2(y_n) \land \gamma(\Delta_n)
\end{align*}
\]

(exists ((briefcase ?b))

(and (briefcase ?b) (in moby ?b) (in crime ?b) (in dict ?b))

(forall ((briefcase ?b))
   (exists ((book ?y)) (in ?y ?b))) \quad \text{briefcase is \{B1,B2\}}

An Example of Confrontation

*start*
(briefcase B) (at B home) (in P B) (at P home)

(at B office) (at P home)
*end*

*start*
(briefcase B) (at B home) (in P B) (at P home)

(briefcase B) (at B ?I) (in ?O1 B)
move B ?I office
(at B office) ~(at B ?I) (at(?O1 office) ~(at(?O1 ?I))

(at B office) (at P home)
*end*
*start*
(briefcase B) (at B home) (in P B) (at P home)

(briefcase B) (at B home) (in ?o1 B)

move B home office

(at B office) ~(at B home) (at ?o1 office) ~(at ?o1 home)

(at B office) (at P home)

*end*

*start*
(briefcase B) (at B home) (in P B) (at P home)

(briefcase B) (at B home) ~(in P B) (in ?o1 B)

move B home office

(at B office) ~(at B home) (at ?o1 office) ~(at ?o1 home)

(at B office) (at P home)

*end*
Final Plan
An Example of Universal Quantification

(define (operator put-in)
  :parameters  (?x ?b ?l)
  :effect     (in ?x ?b))

*start*

(object D) (object B) (briefcase B) (at B home) ~(in D B) (at D office)

(forall ((object ?x)) (at ?x home))

*end*
*start*

(object D) (object B) (briefcase B) (at B home) ~(in D B) (at D office)

(move B ?! home)

(at B home) ~(at B ?!) (at ?o1 home) ~(at ?o1 ?!)

(at B home) (at D home)

*end*

*start*

(object D) (object B) (briefcase B) (at B home) ~(in D B) (at D office)

(move B ?! home)

(at B home) ~(at B ?!) (at D home) ~(at D ?!) (at ?o1 home) ...

(at B home) (at D home)

*end*
Final Plan

*start*

(object D) (object B) (briefcase B) (at B home) ~(in D B) (at D office)

(briefcase B) (at B home) ~(in D B) (in ?o3 B)

**move B home office**

(at B office) ~(at B home) (at ?o3 office) ~(at ?o3 home)

(briefcase B) (at B office) (at D office)

**put-in D B**

(in D B)

(object D) (briefcase B) (at B office) (in D B) (in ?o1 B)

**move B office home**

(at B home) ~(at B office) (at D home) ~(at D office) (at ?o1 home) ...

(at B home) (at D home)

*end*
Algorithm: $\text{ucpop}(\langle A, O, L, B \rangle, \text{agenda}, \Lambda)$

1. **Termination:** If agenda is empty, return $\langle A, O, L, B \rangle$.

2. **Goal reduction:** Remove a goal $\langle Q, A_c \rangle$ from agenda.
   (a) If $Q$ is a quantified sentence then post the universal base $\langle \forall Q, A_c \rangle$ to agenda. Go to 2.
   (b) If $Q$ is a conjunction of $Q_i$ then post each $\langle Q_i, A_c \rangle$ to agenda. Go to 2.
   (c) If $Q$ is a disjunction of $Q_i$ then nondeterministically choose one disjunct, $Q_k$, and post $\langle Q_k, A_c \rangle$ to agenda. Go to 2.
   (d) If $Q$ is a literal and a link $A_p \rightarrow Q A_c$ exists in $L$, fail (an impossible plan).

3. **Operator selection:** Nondeterministically choose any existing (from $A$) or new (instantiated from $\Lambda$) action $A_p$ with effect conjunct $R$ such that $A_p < A_c$ is consistent with $O$, and $R$ (note $R$ is a consequent conjunct if the effect is conditional) unifies with $Q$ given $B$. If no such choice exists then fail. Otherwise, let
   (a) $L' = L \cup \{ A_p \rightarrow Q A_c \}$
   (b) $B' = B \cup \{(u, v) | (u, v) \in \text{MGU}(Q, R, B) \land u, v \text{ not universally quantified variables of the effect } \}$
   (c) $O' = O \cup \{ A_p < A_c \}$

4. **Enable new actions and effects:** Let $A' = A$ and agenda' = agenda.
   If $A_p \not\in A$ then add $A_p$ to $A'$, add $\langle \text{preconds}(A_p) \setminus \text{MGU}(Q, R, B), A_p \rangle$ to agenda', add $\{ A_0 < A_p < A_\infty \}$ to $O$, and add $\text{non-ccd-constraints}(A_p)$ to $B'$. If the effect is conditional and it has not already been used to establish a link in $L$, then add its antecedent to agenda after substituting with $\text{MGU}(Q, R, B)$.

5. **Causal link protection:** For each causal link $l = A_i \rightarrow A_j$ in $L$ and for each action $A_i$ which threatens $l$ nondeterministically choose one of the following (or, if no choice exists, fail):
   (a) **Promotion** If consistent, let $O' = O' \cup \{ A_j < A_i \}$.
   (b) **Demotion** If consistent, let $O' = O' \cup \{ A_i < A_j \}$.
   (c) **Confrontation** If $A_i$'s threatening effect is conditional with antecedent $S$ and consequent $R$, then add $\langle \neg S \setminus \text{MGU}(P, \neg R), A_i \rangle$ to agenda'.

6. **Recursive invocation:** If $B$ is inconsistent then fail; else call $\text{ucpop}(\langle A', O', L', B' \rangle, \text{agenda}', \Lambda)$. 39