PLANNING

CHAPTER 11

Search vs. planning contd.

Planning systems do the following:
1) open up action and goal representation to allow selection
2) divide-and-conquer by subgoaling
3) relax requirement for sequential construction of solutions

<table>
<thead>
<tr>
<th>Search</th>
<th>Planning</th>
</tr>
</thead>
<tbody>
<tr>
<td>States</td>
<td>Logical sentences</td>
</tr>
<tr>
<td>Actions</td>
<td>Precondition/outcomes</td>
</tr>
<tr>
<td>Goal</td>
<td>Logical sentence (conjunction)</td>
</tr>
<tr>
<td>Plan</td>
<td>Constraints on actions</td>
</tr>
</tbody>
</table>

Outline

- Search vs. planning
- STRIPS operators
- Partial-order planning

STRIPS operators

Tidily arranged actions descriptions, restricted language

**ACTION**: Buy(x)

**PRECONDITION**: At(p), Sells(p, x)

**EFFECT**: Have(x)

[Note: this abstracts away many important details!]

Restricted language ⇒ efficient algorithm
Precondition: conjunction of positive literals
Effect: conjunction of literals

A complete set of STRIPS operators can be translated into a set of successor-state axioms

Search vs. planning

Consider the task get milk, bananas, and a cordless drill
Standard search algorithms seem to fail miserably:

Partial ordered plans

Partially ordered collection of steps with

- Start step has the initial state description as its effect
- Finish step has the goal description as its precondition
- Causal links from outcome of one step to precondition of another
- Temporal ordering between pairs of steps

Open condition — precondition of a step not yet causally linked

A plan is complete iff every precondition is achieved
A precondition is achieved iff it is the effect of an earlier step and no possibly intervening step undoes it

After-the-fact heuristic/goal test is adequate
Example

Planning process

Operators on partial plans:
- add a link from an existing action to an open condition
- add a step to fulfill an open condition
- order one step wrt another to remove possible conflicts

Gradually move from incomplete/vague plans to complete, correct plans
Backtrack if an open condition is unachievable or if a conflict is irresolvable

POP algorithm sketch

function POP(initial, goal, operators) returns plan
    plan = MAKE-MEANS-END-PLAN(initial, goal)
    loop do
        if SOLUTION(plan) then return plan
        S_next = SELECT-SUCCESSOR(plan)
        CORDER(operators, plan, S_next)
        NEXTPLAN(operators, plan)
    end

function SELECT-SUCCESSOR(plan) returns S_next
    pick a plan step S_next from STEPS(plan)
    with a precondition c that has not been achieved
    return S_next, c

Example

POP algorithm cont'd.

procedure Choose-Operator(plan, operators, S_next, c)
    choose a step S_next from operators or STEPS(plan) that has c as an effect
    add the causal link $S_{next} \leftarrow S_{next}$ to LIKELY(plan)
    add the ordering constraint $S_{next} \leftarrow S_{next}$ to ORDERED(plan)
    if $S_{next}$ is a newly added step from operators then
        add $S_{next}$ to STEPS(plan)
        add $Start \leftarrow S_{next}$
        $End \leftarrow S_{next}$
    end

procedure Resolve-Threads(plan)
    for each S_thread that threatens a link $S_{i} \leftarrow S_{j}$ in LIKELY(plan) do
        choose either
        - add $S_{i} \leftarrow S_{j}$ to ORDERED(plan)
        - force $S_{i} \leftarrow S_{j}$
        - if not CONSISTENT(plan) then fail
    end
Clobbering and promotion/demotion

A **clobber** is a potentially intervening step that destroys the condition achieved by a causal link. E.g., **Go(Home)** clobbers **At(Supermarket)**:

- **Demotion**: put before **Go(Supermarket)**
- **Promotion**: put after **Buy(Milk)**

Properties of POP

- Nondeterministic algorithm: backtracks at choice points on failure
  - choice of $S_{add}$ to achieve $S_{new}$
  - choice of demotion or promotion for clobber
  - selection of $S_{new}$ is irrevocable

POP is sound, complete, and systematic (no repetition)

Extensions for disjunction, universal, negation, conditionals

Can be made efficient with good heuristics derived from problem description

Particularly good for problems with many loosely related subgoals

Example: Blocks world

"Sussman anomaly" problem

- Start State
  - Clear(x) On(x,z) Clear(y)
  - PutOnTable(x)
- Goal State
  - Clear(x) On(x,z)
  - PutOn(x,y)

+ several inequality constraints

Example contd.
Example contd.

START

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

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

PutOn(B,C)

Cl(B) On(B,z) Cl(C)

PutOn(A,B)

Cl(A) On(A,z) Cl(B)

PutOn(A,B)

Cl(B) clobbers Cl(C)

=> order after

PutOnTable(C)

Cl(C)

On(B,z)

PutOn(A,B)

PutOnTable(C)

PutOn(B,C) clobbers Cl(C)

=> order after

PutOnTable(C)

FINISH