Planning and Means Ends Analysis
15-381 Artificial Intelligence

Jaime Carbonell
23-January-2001

Topics Covered
- B* [previous slide packet]
- Macrooperators [previous slide packet]
- Planning Operators
- Means-Ends Analysis
- Search Control Rules in MEA
- Anytime Planning

Search =3D> Planning:
Parameterized Operators

Multi-state transitions
Instead of: \( \text{Op}_{i,j}: S_i \rightarrow S_j \)
We have: \( \text{Op}_{k,l}: \{S_k\} \rightarrow \{S_l\} \)

Preconditions and Postconditions
- Conjunctive set of first-order predicates
- Arguments can be constants or (typed) variables
- Intentional description of subset of all states
  - Pre-image
    - \( \{S_k\} \) states where preconditions are true
  - Post-image
    - \( \{S_l\} \) states where postconditions are true
- Requires Consistent variable bindings within and across preconditions and postconditions

First Example
OPERATOR DRIVE-CAR(<car>,<driver>,<keys>,<loc-1>)
[PRE: (AT <car> <loc-1>)
 (AT <driver> <loc-1>)
 (CONTAINS-GAS <car>)
 (HAVE <keys> <driver>)
 (CORRESPOND <keys> <car>)]
[POST: (AT <car> <loc-2>)
 (AT <driver> <loc-2>)
 (NOT (AT <car> <loc-1>))
 (NOT (AT <driver> <loc-1>))]}
Second Example

(Equivalent notation
Previous operator: LISP-style
Current operator: PROLOG-style)

OPERATOR: move-robot(r,x,y)
  TYPE: ROBOT(r) & LOC(x) & LOC(y)
  PRE: AT(r,x) & EMPTY(y) & CONNECTED(x,y)
  POST: AT(r,y) NOT(AT(r,x))

OPERATOR: pick-up(r,z)
  TYPE: ROBOT(r) & LOC(x) & LOC(y) &
            OBJ(z) & OBJ(w)
  PRE: AT(r,x) & AT(z,y) & NEXT-TO(x,y) &
                NOT(holding(r,w))
  POST: HOLDING(r,z) NOT(AT(z,y))

Interpretation

• A plan is an o-path: $S_0$ followed by
  an sequence of instantiated operators
  which result in the goal state.

• Variables match objects in state
  ...of specified types only
  ...for which the preconditions hold
  at plan execution time.

• Planning can proceed by forward or
  backward (or any other) search method.

Means-Ends Analysis

Backchaining/Subgoaling Search

1. Let $S_{\text{curr}} := \emptyset \ S$
2. If $S_{\text{curr}} = \emptyset$, then go to next goal (or DONE)
3. Let $\text{OPS}_{\text{app}} := \{ \text{POST}(\text{OP}) \}$
   Goals(Mill(p,s), Drill(p,l,d), Polish(p))
4. If $\text{OPS}_{\text{app}} = \emptyset$, then BACKTRACK
5. Select $\text{OP} \in \text{OPS}_{\text{app}}$, (save alt’s)
6. If match(PRE(\text{OP}), $S_{\text{curr}}$),
   a. let $S_{\text{curr}} := \text{apply}(\text{OP}, S_{\text{curr}})$
   b. Go to step 2
7. Else (i.e. If NOT(match(PRE(\text{OP}), $S_{\text{curr}}$))
   a. MEA($S_G := \{ \text{unmatched}(\text{PRE(\text{OP}))} \}_1 \ \text{S} := \emptyset$
   b. If fail, backtrack
   c. If succeed, apply $\text{OP}$ as above
Control Rules for MEA

Choice Points in MEA

- Choose Operator, if several applicable
- Choose Goal, if > 1 subgoals pending
- Choose Variable Binding, if > 1 possible

Types of Control Rules

- Select -- Choose an alternative and eliminate other contenders
- Reject -- Reject an alternative and retain other contenders
- Prefer -- Try one alternative first and retain others for possible backtracking

Example

CONTROL-RULE: Carry-before-move
TYPE: SELECT
PRE: Goals(Move(r,x,y), Pick-up(r,z,v))
POST: Pick-up(r,z)

CONTROL-RULE:
TYPE: REJECT
PRE: Goals(Mill(p,f), Drill(p,l,d,s), Polish(p))
POST: Polish(p)