Planning
Recap

• Basics of Propositional Logic (PL)
• Inferring new facts from existing knowledge: Modus ponens and resolution
• Reasoning about the world using PL
• Avoiding an agonizing death at the hands of the Wumpus
Exercise

- New rule: The Wumpus cannot be in the same square as a pit
- Can you determine where the Wumpus is?
- What about a pit?
Agenda

• Building on logic to form plans
• How do we choose from a number of possible actions:
  • to bring us closer to the goal?
  • in a more "intelligent" manner than search?
First-Order Logic (FOL)

- Propositional logic: propositions (sentences)
- FOL adds quantification ($\forall$, $\exists$) and predicates
Predicates

- assume that Spot and Fido are dogs
- then the predicate, Dog(x)
  - returns TRUE if x is Spot or Fido
Unification

- Let $p$ and $q$ be sentences in FOL
- Let $U$ be a unifier, i.e., some set of substitutions of values for variables
- $\text{subst}(U, x)$ is the result of applying the substitutions of $U$ to sentence $x$
- If $\text{subst}(U, p) = \text{subst}(U, q)$ then $\text{UNIFY}(p, q) = U$
  - The unification of $p$ and $q$ is the result of applying $U$ to both of them.
Example

Dog(Spot)
Man(John)

UNIFY((Bites(Spot,y) ∧ Dog(x) ∧ Man(y)),
     (Bites(z,John) ∧ Dog(z) ∧ Man(John))) =
     {x/Spot; y/John; z/Spot}
Motivation

• previously considered decision-making problems from the perspective of search and game-playing
  • Given current state, enumerate all possible future states
  • Pick best action to execute from current state
• when does this make sense?
Grocery Shopping as Search
“from home, get milk, some bananas, and a cordless drill”

- assume we will use search technique with heuristic: minimize the number of items we have not yet acquired
- how to choose best operator? may be thousands!
- before agent can purchase anything, it has to get to the store, but how does a search technique know this?
A Better Way (sometimes)

- many actions are obviously useless or unproductive
- can be quite expensive to examine all of these
- what if we know outcome of actions?

planning = find a sequence of actions that achieves a goal when performed in a given state
Language of Planning Problems

• States: conjunction of positive literals
  • e.g., Poor $\land$ Unknown or $At(Plane_1, Melbourne)$

• Goals: a partially specified state
  • e.g., Rich $\land$ Famous or $At(P_2, Tahiti)$
Comparison of Actions

**search**
- described by state transitions
- must be considered in-order

**planning**
- described by preconditions & effects
- can be considered in any order
STRIPS

• language used by most classical planners
• initial state: specifies everything that is true in the world (everything else assumed false)
  \( \text{At(Home)} \land \text{Have(Money)} \)
• goals: represented by conjunctions of literals
  \( \text{At(Home)} \land \text{Have(Milk)} \land \text{Have(Bananas)} \land \text{Have(Cookies)} \)
• goals can contain variables, e.g., “get to a store that sells milk”
  \( \text{At}(x) \land \text{Sells}(x, \text{Milk}) \)
Representation of Actions

- **preconditions**
  - what must be true before action can be performed
- **effects or post-conditions**
  - what must be true (what changed) after action is executed

**e.g.**,

\[
\text{Action} (\text{Fly} (p, \text{from}, \text{to}), \\
\text{PRECOND: } \text{At}(p,\text{from}) \land \text{Plane}(p) \land \text{Airport(from)} \land \text{Airport(to)} \\
\text{EFFECT: } \neg\text{At}(p,\text{from}) \land \text{At}(p,\text{to}))
\]

- effects may be divided into add list and delete list (for negative literals)
- assumption that every literal not mentioned in EFFECT remains unchanged
Forward State-Space Search (Progression) Planners

- search forward from initial state
- determine which operators apply using preconditions
- use effects lists to compute new state
Algorithm

procedure PROGRESSION-PLAN(s, plan)
    for some possible operator, $\alpha_i$
        $u \leftarrow$ result of UNIFY(s, preconditions of $\alpha_i$)
        if unification step succeeds then
            add $\alpha_i$ to plan
            apply substitution list of $u$ to DeleteList($\alpha_i$) and AddList($\alpha_i$)
            $t \leftarrow s$
            for each $d_i$ in DeleteList($\alpha_i$) delete $d_i$ from $t$
            for each $a_i$ in AddList($\alpha_i$) add $a_i$ to $t$
            if GOAL-TEST ($t$) succeeds then return plan
        return PROGRESSION-PLAN ($t$, plan)
Blocks World Example

• $\text{pickup}(x)$
  • P, D: ontable(x), clear(x), HE (i.e., hand empty)
  • A: holding(x)

• $\text{putdown}(x)$
  • P, D: holding(x)
  • A: ontable(x), clear(x), HE

• $\text{stack}(x, y)$ // $\text{putdown}(x)$ on another block(y)
  • P,D: holding(x), clear(y)
  • A: HE, on(x,y), clear(x)

• $\text{unstack}(x, y)$ // $\text{pickup}(x)$ sitting on block(y)
  • P,D: HE, clear(x), on(x,y)
  • A: holding(x), clear(y)

• Note: in general, P and D are not equivalent
Exercise

- write the initial state description
- apply progression planning to reach the goal state
Practical Considerations

• main problem: often have huge search space because of branching factor
• not practical for real-world problems
Backward State-Space (Regression) Planners

- search backwards from goal state to initial state
- **advantage**: consider only relevant actions; i.e., those that achieve one of the conjuncts of the goal
- significantly decreases branching factor
Regression Planner

procedure REGRESSION-PLAN \( (t, \, plan) \)
for some possible operator, \( \alpha \)
if current state contains \( \geq 1 \) literal \( L \), that unifies with a member \( a_i \) of AddList(\( \alpha \))
add \( \alpha \) to head of \( plan \)
\( u \leftarrow \) result of UNIFY \( (L, \, a_i) \)
\( p' \leftarrow \) apply substitution list of \( u \) to Preconditions(\( \alpha \))
\( t' \leftarrow \) apply substitution list of \( u \) to terms of \( t \)
regress each member \( m \) of \( t' \) through \( a \) as follows:
\( \) if \( m \) is in AddList(\( \alpha \)) then True
\( \) else if \( m \) is in DeleteList(\( \alpha \)) then False
\( \) else leave \( m \) as is
previous state \( s \leftarrow \) UNION \( (p', \, t') \)
if \( s = \) INITIAL-STATE then return \( plan \)
return REGRESSION-PLAN \( (s, \, plan) \)
Sussman Anomaly

- if we do On(B,C) first, likely to undo it as we try to achieve On(A,B)
- similarly, if we do On(A,B) first, likely to undo it with On(B,C)
- problem stems from interaction of goals in linear planners
- forward and backward state-space planners are capable of handling the Sussman anomaly, but inordinate effort required
Recap

- **planning** = find a sequence of actions that achieves a goal when performed in a given state
- Language of planning problems: *States*, *Goals*, *Preconditions*, and *Effects*
- Progression and regression planning and their limitations
- Introduced STRIPS notation of planning problems (actions, preconditions and effects)
Partial Order Planning (POP)

start with initial plan
• consists only of Start and Finish states
• at each iteration, add one more step
• if inconsistent, backtrack
only adds steps that achieve unachieved preconditions
Grocery Shopping Problem

• initial state: $At(Home)$

• goal: $Have(\text{Drill}) \land Have(\text{Milk}) \land Have(\text{Banana}) \land At(\text{Home})$

• actions:
  Action $(\text{Go (there)},$
  PRECOND: $At(\text{here})$
  EFFECT: $At(\text{there}) \land \neg At(\text{here})$)

  Action $(\text{Buy (x)},$
  PRECOND: $At(\text{store}) \land Sells(\text{store, x})$
  EFFECT: $Have(x)$)
Start and Finish Steps

ACTION: Start
• EFFECT: At(Home) ^ Sells (HWS, Drill) ^ Sells (SM, Milk) ^ Sells (SM, Banana)
  Notice: Start step adds, as its effect, all necessary facts to knowledge base.

ACTION: Finish
• PRECOND: Have(Drill) ^ Have(Milk) ^ Have(Banana) ^ At(Home)
Achieve preconditions of ‘Finish’
Achieve preconditions of ‘Buy’

- start with Sells()
Achieve preconditions of ‘Buy’

- now satisfy the At() preconditions
First Attempt

- Link to At(Home) from Start
Protecting Causal Links

Threat

$S_3$ demoted

$S_3$ promoted
Is promotion/demotion possible?
Second Attempt

- Add causal link from Go(HWS) to Go(SM)
Resolving the Threat

- Add temporal constraint for Buy(Drill) to precede Go(SM)
And now to get home…

- But again… beware of threats!

Careful!
Resolving the Threat

• Add temporal constraint for Buy(Milk), Buy(Bananas) to precede Go(Home)
Unrolled Solution
Recap

- **planning** = find a sequence of actions that achieves a goal when performed in a given state
- Language of planning problems: **States**, **Goals**, **Preconditions**, and **Effects**
- Progression planning and its limitations
- Partial-order-planning as a more flexible approach