Planning

- How are planning and problem solving different?
- Review representations of problem solving
  - Actions: extend-state procedures
  - States: complete descriptions of state (8-puzzle board)
  - Goals: goal-test and information in heuristic function
  - Plans: sequence of actions
- Develop a plan to win 8-Puzzle
  - Generate all possible successor states from initial-state

Difficulties with using search-based problem solving for planning

- Must look at all actions at every step
- Cannot use goal to focus choice of action
- Generate steps in chronological order
  - Only have access to local information, cannot look long-term
  - Frequently a later move will determine what an earlier move should be

3 Key Ideas of Planning (fixes)

- Open up representations
  - Make goal, state, and action information accessible when selecting an action
  - Actions now have preconditions and effects
  - Usually in 1st order logic
- Homework example
  - Goal: Finished(homework)
  - One of the actions: Program(x) achieves Finished(x)
  - Won’t consider irrelevant actions
- Allow for plan generation in any order
  - Make important decisions first such as what to eat in order to limit branching factor
  - Plan for each part of goal separately and then combine results
    - “Divide and conquer” approach
  - Often easy to solve the smaller problem first
  - Feasible because of independence in the world
    - Ex. Goal is to complete meal.
      - Eat(rice) before Eat(beans) usually doesn’t undo accomplishment of eating you rice.
      - Does not work with “tricky puzzles”

Situation Calculus Representation

- Planning problem in logical sentences
  - Initial state: Sentence about $S_0$
    - At(Home, $S_0$) ^ Not(Have(Milk, $S_0$) ^ Not(Have(Bananas, $S_0$) ^ Not(Have(Drill, $S_0$)
  - Goal state: Query for a given situation
    - At(Home, s) ^ Have(Milk, s) ^ Have(Bananas, s) ^ Have(Drill, s)
  - Operators:
    - Delta(a, s, Have(Milk, Result(s, a)) ^ ([a = Buy(Milk) ^ At(Supermarket, s) v (Have(Milk, s) ^ Drop(Milk)])

STRIPS Planning Language [Fikes and Nilsson, 1971]

- Most planners use STRIPS or its extensions
- Initial States
  - Conjunctions of predicates applied to constants
    - At(Home) ^ Not(Have(Milk)) ^ ...
- Goal States
  - Same as initial state but allows variables (universally quantified)
    - At(s) ^ Sells(s, Milk)
STRIPS Actions
- Action name: Go(there)
- Precondition
  - Conjunction of positive literals that must be true in order to do the action
- Effect
  - Conjunction of positive or negative literals that hold once the action is performed
- Action can have variables (universally quantified)
- Must be fully instantiated to be applied

GPS Algorithm
Means-ends analysis:
1. Input state
2. Identify differences from goal. If none, done. Else...
3. Pick most important difference
4. Pick operator to decrease it.
5. Recursively apply GPS to repair any differences between situation and operator's preconditions
6. Apply operator
7. Recursively apply GPS to the result
8. Go to 2.

Pro's and Con's of Means Ends Analysis
- Works effectively on minor problems
  - If your table is perfect except for its color, you pick a way to change its color.
  - If your table is perfect except for missing a leg, pick a way to add a leg.
- Potential problems
  - Can undo needed actions
  - Fixing one difference can make others worse
    - Examples
  - Can miss the big picture [Hammond 89]

Dimensions of planners
- Direction of planning
  - Progression planner
    - Searches forward from initial state to goal
    - High branching factor, huge search space
  - Regression planner
    - Searches from the goal to the initial state
    - Usually smaller branching factor and space
    - Not as many operators have effects that are the preconditions for the goal state
    - Ex. STRIPS

Planning search space
- Situation space
  - Searches through possible situations
  - What we have done so far in search problems
- Plan space
  - Searches through plans
  - Starts with an initial plan, (initial state -> goal state)
  - Operates on partial plan until it is complete
    - add a step
    - impose an ordering of the steps
    - instantiate a variable

Representation of plans
- Set of steps (operators)
- Set of ordering constraints on steps
- Set of variable bindings
  - e.g. If v is a variable in a step, the v = x, where x is either a constant or another variable in a step
- Set of causal links
  - $S_i \rightarrow S_j$ says that $S_i$ achieves c, a precondition of $S_j$
Partial order plan generation and solution

- Use principle of “least commitment” when generating plan
  - Only impose orderings on steps that need to be ordered leaving a partially ordered plan
  - Bind only the variables that need to be bound during the plan creation
- Solution: The plan that an agent can execute to achieve goal
  - Complete
    - Every step’s precondition must be satisfied
  - Consistent
    - No contradictions in ordering or bindings

Partial Order Planning Example
Putting on Shoes

- Initial State: Empty
- Goal State: RightShoeOn ^ LeftShoeOn
- Operators
  - Op(ACTION: RightShoe, PRECOND: RightSockOn, EFFECT: RightShoeOn)
  - Op(ACTION: RightSock, EFFECT: RightSockOn)
  - Op(ACTION: LeftShoe, PRECOND: LeftSockOn, EFFECT: LeftShoeOn)
  - Op(ACTION: LeftSock, EFFECT: LeftSockOn)

Shoe example con’t

Plan (STEPS: {S1: Op(ACTION:Start)
               S2: Op(ACTION:Finish,
               PRECOND: RightShoeOn ^ LeftShoeOn),
               ORDERINGS: {S1 < S2}
               BINDINGS{}.
               LINKS{}})

Find the solution.