Informatics 2D: Reasoning and Agents

Alex Lascarides

School of informatics

THE UNIVERSITY OF EDINBURGH

Lecture 17a: Forward State-Space Search in Planning

Where are we?

So far ...

- we defined the planning problem
- discussed problem with using search and logic in planning
- introduced representation languages for planning
- looked at blocks world example

Over next few slots ...

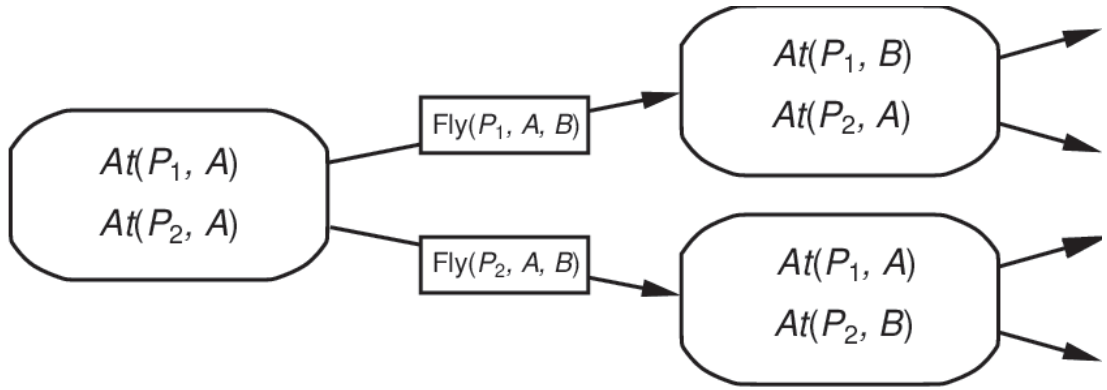
- **State-space search and partial-order planning**
- Now: **state-space search**

Planning with state-space search

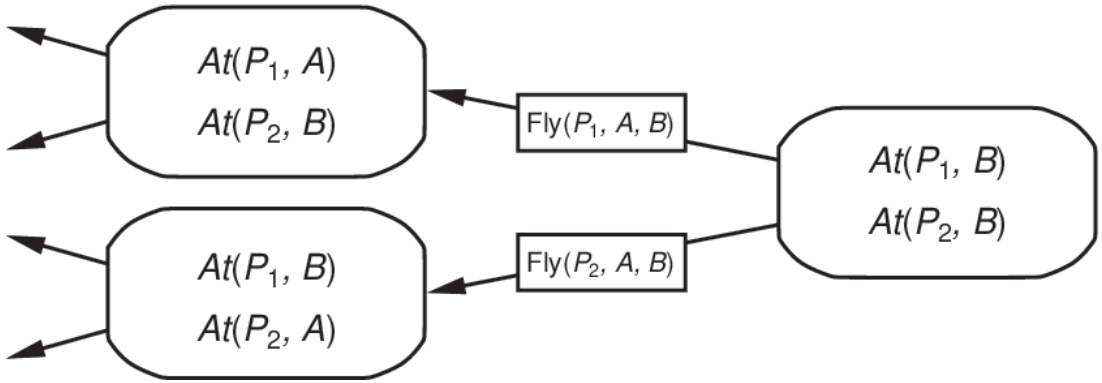
- Most straightforward way to think of planning process: search the space of states using action schemata
- Since actions are defined both in terms of preconditions and effects we can search in both directions
- Two methods:
  1. **forward state-space search**: Start in initial state; consider action sequences until goal state is reached.
  2. **backward state-space search**: Start from goal state; consider action sequences until initial state is reached

Planning with state-space search

(a)



(b)



Forward state-space search

- Also called **progression** planning
- Formulation of planning problem:
  - Initial state of search is initial state of planning problem (=set of positive literals)
  - Applicable actions are those whose preconditions are satisfied
  - Single successor function works for all planning problems (consequence of action representation)
  - Goal test = checking whether state satisfies goal of planning problem
  - Step cost usually 1, but different costs can be allowed

Forward state-space search

- Search space is finite in the absence of function symbols
- Any complete graph search algorithm (like A*) will be a complete graph planning algorithm
- Forward search does not solve problem of irrelevant actions (all actions considered from each state)
- Efficiency depends largely on quality of heuristics
- Example:
  - Air cargo problem, 10 airports with 5 planes each, 20 pieces of cargo
  - Task: move all 20 pieces of cargo at airport A to airport B
  - Each of 50 planes can fly to 9 airports, each of 200 packages can be unloaded or loaded (individually)
  - So approximately 10K executable actions in each state ($50 \times 9 \times 200$)
  - Lots of irrelevant actions get considered, although solution is trivial!

Backward state-space search

- In normal search, backward approach hard because goal described by a set of constraints (rather than being listed explicitly)
- Problem of how to generate predecessors, but planning representations allow us to consider only **relevant** actions
- Exclusion of irrelevant actions decreases branching factor
- In example, only about 20 actions working backward from goal
- **Regression planning** = computing the states from which applying a given action leads to the goal
- Must ensure that actions are **consistent**, i.e. they don't undo any desired literals

Air cargo domain example

- Goal can be described as

$$At(C_1, B) \wedge At(C_2, B) \wedge \ldots At(C_{20}, B)$$

- To achieve $At(C_1, B)$ there is only one action, $Unload(C_1, p, B)$ (p unspecified)
- Can do this action only if its preconditions are satisfied.
- So the predecessor to the goal state must include $In(C_1, p) \wedge At(p, B)$, and should not include $At(C_1, B)$ (otherwise irrelevant action)
- Full predecessor:

$$In(C_1, p) \wedge At(p, B) \wedge \ldots \wedge At(C_{20}, B)$$

- $Load(C_1, p)$ would be inconsistent (negates $At(C_1, B)$)

Backward state-space search

- General process of constructing predecessors for backward search given goal description G, relevant and consistent action A:
  - Any positive effects of A that appear in G are deleted
  - Each precondition of A is added unless it already appears
- Any standard search algorithm can be used, terminates when predecessor description is satisfied by initial (planing) state
- First-order case may require additional substitutions which must be applied to actions leading from state to goal

Heuristics for state-space search

- Two possibilities:
  1. Divide and Conquer (**subgoal decomposition**)
  2. Derive a **Relaxed Problem**
- Subgoal decomposition is ...
  - optimistic (admissible) if negative interactions exist (e.g. subplan deletes goal achieved by other subplan)
  - pessimistic (inadmissible) if positive interactions exist (e.g. subplans contain redundant actions)
- Relaxations:
  - drop all preconditions (all actions always applicable, combined with subgoal independence makes prediction even easier)
  - remove all negative effects (and count minimum number of actions so that union satisfies goals)
  - empty delete lists approach (involves running a simple planning problem to compute heuristic value)

Summary

- State-space search approaches (forward/backward)
- Heuristics for state-space search planning

Next time...

- Partial-order planning