Knowledge representation and reasoning
Lecture 18: Planning

Natasha Alechina
natasha.alechina@nottingham.ac.uk

Plan of the lecture

1. What is Planning
2. Classical Planning
3. Planning problem
4. Forward and regression planning

What is Planning

- Planning is a reasoning problem:
- what actions (plan) to perform in order to make some condition (goal) true
- Planning is central to AI as the study of intelligent behaviour achieved through computational means
- In general, in real world, it is a very difficult problem

Classical Planning

Classical planning approach makes several assumptions to simplify this problem:

- Environment is deterministic
- Environment is observable
- Environment is static (it only changes in response to the agent's actions)

STRIPS operators

- STRIPS planning language (Fikes and Nilsson, 1971, derives from work at SRI International on robot called Shakey)
- goals are conjunctions of atoms (positive literals). We will often replace conjunctions of literals with sets of literals (meaning, all of the literals in the set are true)
- actions descriptions (action schemas) assume finite preconditions and effects of fixed form:
  - Precondition: conjunction of positive literals (we will write it as a set)
  - Effect: conjunction of literals (or a set of literals)

ACTION: $buy(x)$
PRECONDITION: $At(p), Sells(p, x)$
EFFECT: $Have(x)$

PDDL

- Planning Domain Definition Language
- Less restrictive than STRIPS
- Preconditions and goals can contain negative literals
- Other levels in PDDL also allow action durations, resource requirements etc. (not in this lecture)

Add and Delete lists

- Given an action schema

  ACTION: a
  PRECONDITION: some literals
  EFFECT: $E_1, \ldots, E_m$

- $Add(a) = \{E \mid E$ is a positive literal in EFFECT$\}$ (positive effects of a)
- $Del(a) = \{P \mid E = \neg P$ where $E \in$ EFFECT$\}$ (atoms appearing with negation in the effect of a)

Planning domain

- Planning domain is described by giving a list of fluents and action schemas
- Fluents are predicates, have no situation argument
- States are sets of ground fluents; fluents which are not mentioned in a state description are false (closed world assumption)
- a is possible in s if precondition of a is true in s
- the state resulting from executing a in s,

$$do(s, a) = (s - Del(a)) \cup Add(a)$$

Example (slightly modified)

ACTION: $buy(x)$
PRECONDITION: $At(p), Sells(p,x), Have(Money)$
EFFECT: $Have(x), \neg Have(Money)$

- $Del(buy(Jaguar)) = \{Have(Money)\}$
- $Add(buy(Jaguar)) = \{Have(Jaguar)\}$
- If $s =$ $\{At(JDealer), Sells(JDealer, Jaguar), Blue(Sky), Have(Money)\}$,
- $buy(Jaguar)$ is possible in s
- $do(s, buy(Jaguar) = (s - \{Have(Money)\}) \cup \{Have(Jaguar)\} =$ $\{At(JDealer), Sells(JDealer, Jaguar), Blue(Sky), Have(Jaguar)\}$

Planning problem

- Planning problem = planning domain + objects + initial state + goal
- Goal is a conjunction of literals: $\mathit{Have}(\mathit{Jaguar}) \wedge \neg \mathit{At}(\mathit{Jail})$
- Can solve planning problem using search

State and goal description

- State descriptions are always ground (no variables)
- Goal description may have variables: $At(x) \wedge Have(y)$
- A property with a variable such as $At(x)$ is satisfied at a state if there is a way of substituting an object for x so that the resulting formula is true in the state
- An atomic ground formula $At(Home)$ is true iff it is in the state description
- A negation of a ground atom $\neg At(G)$ is true iff the atom $At(G)$ is not in the state description.

Forward and regression planning

- Usual search: forward search from the initial state to a goal state
- Nothing prevents us from searching from a goal state back to the initial state
- Sometimes given the branching factor it is more efficient to search backward
- Motivating example: imagine trying to figure out how to get to some small place with few traffic connections from somewhere with a lot of traffic connections

Simple example of forward planning

Planning domain:

- Fluents : $At(x)$ (at place x), $Sells(x, y)$ (shop x sells y), $Have(x)$ (have x)
- Two action schemas:

ACTION: $buy(x)$
PRECONDITION: $At(p), Sells(p, x), Have(Money)$
EFFECT: $Have(x), \neg Have(Money)$

ACTION: $go(x, y)$
PRECONDITION: $At(x)$, $x \neq y$
EFFECT: $At(y)$, $\neg At(x)$

Simple example of forward planning 2

- Planning problem: planning domain above plus
- Objects: *Money*, *J* (for Jaguar), *Home*, *G* (for Garage)
- Initial state: $At(Home), Have(Money), Sells(G, J)$
- Goal state: $Have(J)$

Simple example of forward planning 3

$$s_1 = \{At(Home), Have(Money), Sells(G, J)\}$$

buy(x) *not available for any* x *(don't have* $Sells(Home, x)$*)*

$$\downarrow go(Home, G)$$

$$s_2 = \{At(G), Have(Money), Sells(G, J)\}$$

$go(G, Home)$ *also available*

$$\downarrow buy(J)$$

$$s_3 = \{At(G), Have(J), Sells(G, J)\}$$

Heuristics for forward planning

- Similar to search, cf A^*: performance improves by orders of magnitude if a good heuristic is used
- Number of fluents in the goal which will be satisfied by the next action
- add more edges to the graph (make more actions possible), and use solutions to the resulting problem as a heuristic. (There are often more efficient algorithms to solve the relaxed problem.) Examples: remove (some) preconditions, ignore delete lists
- For example, in 8 puzzle assume tiles can move to occupied spaces = Manhattan distance heuristic
- abstract the problem (make the search space smaller)

Backward (regression) planning

- Also called relevant-states search
- Start at the goal state(s) and do regression (go back).
- To be precise, there we start with a ground goal description g which describes a set of states (all those where $Have(J)$ holds but $Have(Money)$ may or may not hold, for example).

Backward (regression) planning 2

- Given a goal description g and a ground action a, the regression from g over a gives a state description g':
- $g' = (g - Add(a)) \cup \{Precondition(a)\}$
- For example, if the goal is $Have(J)$, and a is $buy(J)$
- $g' = (\{Have(J)\} - \{Have(J)\}) \cup$ $\{At(p), Sells(p, J), Have(Money)\}$ = $\{At(p), Sells(p, J), Have(Money)\}$
- note that g' is partially uninstantiated (p is a free variable). In our example, there is only one match for p, namely G, but in general there may be several.

Backward (regression) planning 3

- Which actions to regress over?
- Relevant actions: have an effect which is in the set of goal elements and no effect which negates an element of the goal.
- For example, $buy(J)$ is a relevant action.
- Search backwards from g, remembering the actions and checking whether we reached an expression applicable to the initial state.

Simple example of backward planning

$$Have(J)$$

$$\uparrow buy(J)$$

$$At(x), Have(Money), Sells(x, J)$$

Does not match the initial state yet

$$\uparrow go(y, x)$$

$$At(y), Have(Money), Sells(x, J)$$

Matches the initial state with $y/Home$ and x/G

Comparison of forward and backward planning

- Usually lots of actions available for forward planning
- Easier to find heuristics for forward planning
- Backward planning considers a lot fewer actions/relevant states than forward search, but uses sets of states (g, g') - hard to come up with good heuristics.

Use of logic and deduction

- In situation calculus, planning *is* deduction
- In 'normal' planning, we only need to check whether a state description entails some property, for example, $s \models P(A, B) \wedge \neg Q(A)$
- In simple cases, like in this lecture, this just involves checking that $P(A, B)$ is in the list of properties s has, and $Q(A)$ is not (closed world assumption: if $Q(A)$ is not listed, then $\neg Q(A)$ must be true)
- However, often planning domains are described using additional axioms, and then checking $s \models P(A, B)$ may involve more complex reasoning (whether $P(A, B)$ follows from the description of s and the axioms).

What next

- Please do an informal exercise on planning (on moodle, together with an answer)
- Formal exercise will be along the same lines
- Next lecture: Planning continued.
- Brachman and Levesque, Chapter 15.
- Russell and Norvig, Chapter on Classical Planning.
- Rich and Knight, Chapter on Goal Stack Planning.