Informatics 2D: Reasoning and Agents

Alex Lascarides

School of Informatics
The University of Edinburgh

Lecture 18a: Planning and acting in the real world: Introduction

Where are we?

So far ...

- Discussed PDDL
- Algorithms for finding valid plans in PDDL
  - State-space search
  - Partial-order planning
- But so far, confined to **classical planning**

Today ...

- **Planning and acting in the real world I**

Planning/acting in Nondeterministic Domains

- So far only looked at **classical** planning,
  i.e. environments are fully observable, static, deterministic
- Also assumed that action descriptions are correct and complete
- Unrealistic in many real-world applications:
  - Don't know everything; may even hold incorrect information
  - Actions can go wrong
- Distinction: **bounded** vs. **unbounded** indeterminacy: can possible preconditions and effects be listed at all?
- Unbounded indeterminacy related to qualification problem

Methods for handling indeterminacy

- **Sensorless/conformant planning**: achieve goal in all possible circumstances, relies on coercion
- **Contingency planning**: for partially observable and non-deterministic environments; includes sensing actions and describes different paths for different circumstances
- **Online planning and replanning**: check whether plan requires revision during execution and replan accordingly

Example Problem: Paint table and chair same colour

Initial State: We have two cans of paint and table and chair, but colours of paint and of furniture is unknown:

$$Object(Table) \wedge Object(Chair) \wedge Can(C_1) \wedge Can(C_2) \wedge InView(Table)$$

Goal State: Chair and table same colour:

$$Color(Chair, c) \wedge Color(Table, c)$$

Actions: To look at something; to open a can; to paint.

Formal Representation of the Three Actions

Now we allow variables in preconditions that aren't part of the actions's variable list!

$$
\begin{aligned}
&Action(RemoveLid(can),\\
&\quad \text{Precond:} Can(can)\\
&\quad \text{Effect:} Open(can))
\end{aligned}
$$

$$
\begin{aligned}
&Action(Paint(x, can),\\
&\quad \text{Precond:} Object(x) \wedge Can(can) \wedge Color(can, c) \wedge Open(can)\\
&\quad \text{Effect:} Color(x, c))
\end{aligned}
$$

$$
\begin{aligned}
&Action(LookAt(x),\\
&\quad \text{Precond:} InView(y) \wedge (x \neq y)\\
&\quad \text{Effect:} InView(x) \wedge \neg InView(y))
\end{aligned}
$$

Sensing with Percepts

- A percept schema models the agent's sensors.
- It tells the agent what it knows, given certain conditions about the state it's in.

$$Percept(Color(x, c),$$
$$\text{Precond:} Object(x) \wedge InView(x))$$

$$Percept(Color(can, c),$$
$$\text{Precond:} Can(can) \wedge Open(can) \wedge InView(can))$$

- A fully observable environment has a percept axiom for each fluent with no preconditions!
- A sensorless planner has no percept schemata at all!

Planning

- One could coerce the table and chair to be the same colour by painting them both—a sensorless planner would have to do this!
- But a contingent planner can do better than this:
  1. Look at the table and chair to sense their colours.
  2. If they're the same colour, you're done.
  3. If not, look at the paint cans.
  4. If one of the can's is the same colour as one of the pieces of furniture, then apply that paint to the other piece of furniture.
  5. Otherwise, paint both pieces with one of the cans.
- Next time: we'll look at these types of planning in more detail. . .

Summary

Planning must deal with:

- Actions with non-determinate outcomes
- States that are partially observable

Some new techniques:

- Percepts
- Planning strategies:
  conformant, contingent, online monitoring and re-planning

Next time:

- Sensorless and contingent planning in more detail