

Informatics 2D: Reasoning and Agents

Slides provided by Prof. Alex Lascarides

Lecture 18: Planning and acting in the Real World I

Where are we?

Last time ...

- Discussed planning with state-space search
- Identified weaknesses of this approach
- Introduced partial-order planning
 - Search in plan space rather than state space
 - Described the POP algorithm and examples

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!

$Action(RemoveLid(can),$
Precond: $Can(can)$
Effect: $Open(can)$)

$Action(Paint(x, can),$
Precond: $Object(x) \wedge Can(can) \wedge Color(can, c) \wedge Open(can)$
Effect: $Color(x, c)$)

$Action(LookAt(x),$
Precond: $InView(y) \wedge (x \neq y)$
Effect: $InView(x) \wedge \neg InView(y)$)

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.

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

- 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.

What's needed?

When sensors aren't powerful enough

- Don't know the value of all relevant fluents
- So you must plan using your **beliefs**, not the representation of the actual state.
- How do we represent beliefs?

When actions can have more than one outcome

- Need to represent **conditional effects** in action schemata.

What's needed?

When sensors aren't powerful enough

- Don't know the value of all relevant fluents
- So you must plan using your **beliefs**, not the representation of the actual state.
- How do we represent beliefs?

When actions can have more than one outcome

- Need to represent **conditional effects** in action schemata.

How to represent belief states

1. Sets of state representations, e.g.

$$\{(AtL \wedge CleanR \wedge CleanL), (AtL \wedge CleanL)\}$$

(2^n states!)

2. Logical sentences can capture a belief state, e.g. $AtL \wedge CleanL$ shows ignorance about $CleanR$ by not mentioning it!
 - This often offers a more compact representation, but
 - Many equivalent sentences; need **canonical** representation to avoid general theorem proving; E.g:
 - All representations are ordered conjunctions of literals (under open-world assumption)
 - But this doesn't capture everything (e.g. $AtL \vee CleanR$)
3. Knowledge propositions, e.g. $K(AtR) \wedge K(CleanR)$ (closed-world assumption)
 - Will use second method, but clearly loss of expressiveness

Beliefs and Sensorless Planning

- When you have no sensors, you need:
 - to represent and track your (changing) **beliefs** as you perform actions ...
 - ... and so cope with **sensorless planning**

Example

Table and chair, two cans of paint

you know these objects exist, but you can't see them

You can open cans, and paint furniture

Goal: table and chair to be same colour

Sensorless Planning Example: The Belief States

- There are no *InView* fluents, because there are no sensors!
- There are unchanging facts:
 $Object(Table) \wedge Object(Chair) \wedge Can(C_1) \wedge Can(C_2)$
- And we know that the objects and cans have colours:
 $\forall x \exists c Color(x, c)$
- After skolemisation this gives an initial belief state:

$$b_0 = Color(x, C(x))$$

- A **belief state** corresponds exactly to the set of possible worlds that satisfy the formula—**open world assumption**.

The Plan

$[RemoveLid(C_1), Paint(Chair, C_1), Paint(Table, C_1)]$

Rules:

- You can only apply actions whose preconditions are satisfied by your current belief state b .
- The **update of a belief state b given an action a** is the set of all states that result (in the physical transition model) from doing a in each possible state s that satisfies belief state b :

$$b' = \text{Result}(b, a) = \{s' : s' = \text{Result}_P(s, a) \wedge s \in b\}$$

Or, when a belief b is expressed as a formula:

- 1 If action adds l , l becomes a conjunct of the formula b' (and the conjunct $\neg l$ removed, if necessary); so $b' \models l$
- 2 If action deletes l , $\neg l$ becomes a conjunct of b' (and l removed).

Showing the Plan Works

$$\begin{aligned} b_0 &= \text{Color}(x, C(x)) \\ b_1 &= \text{Result}(b_0, \text{RemoveLid}(C_1)) \\ &= \text{Color}(x, C(x)) \wedge \text{Open}(C_1) \\ b_2 &= \text{Result}(b_1, \text{Paint}(\text{Chair}, C_1)) \\ &\quad (\text{binding } \{x/C_1, c/C(C_1)\} \text{ satisfies Precond}) \\ &= \text{Color}(x, C(x)) \wedge \text{Open}(C_1) \wedge \text{Color}(\text{Chair}, C(C_1)) \\ b_3 &= \text{Result}(b_2, \text{Paint}(\text{Table}, C_1)) \\ &= \text{Color}(x, C(x)) \wedge \text{Open}(C_1) \wedge \\ &\quad \text{Color}(\text{Chair}, C(C_1)) \wedge \text{Color}(\text{Table}, C(C_1)) \end{aligned}$$

Conditional Effects

- So far, we have only considered actions that have the **same effects** on all states where the preconditions are satisfied.
- This means that any initial belief state that is a conjunction is updated by the actions to a belief state that is also a conjunction.
- But some actions are best expressed with conditional effects.
- This is especially true if the effects are non-deterministic, but in a bounded way.

Extending action representations

- Disjunctive effects: $Action(Left, Precond: AtR, Effect: AtL \vee AtR)$

- Conditional effects:

$Action(Vacuum,$

Precond:

Effect: (**when** $AtL : CleanL$) \wedge (**when** $AtR : CleanR$))

- Combination:

$Action(Left,$

Precond: AtR

Effect: $AtL \vee (AtL \wedge (\text{when } CleanL : \neg CleanL))$)

- Conditional steps: **if** $AtL \wedge CleanL$ **then** $Right$ **else** $Vacuum$

The earlier painting furniture example

Planning Problem

Table and chair, two cans of paint,
can open can, paint furniture with paint inside
Goal: table and chair the same colour

Contingent Plan

- 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.

The Three Actions (reminder)

$Action(RemoveLid(can),$
Precond: $Can(can)$
Effect: $Open(can)$)

$Action(Paint(x, can),$
Precond: $Object(x) \wedge Can(can) \wedge Color(can, c) \wedge Open(can)$
Effect: $Color(x, c)$)

$Action(LookAt(x),$
Precond: $InView(y) \wedge (x \neq y)$
Effect: $InView(x) \wedge \neg InView(y)$)

Percepts (reminder)

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

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

Formal Representation of the Contingent Plan

```
[LookAt( Table), LookAt( Chair)
  if Color( Table, c)  $\wedge$  Color( Chair, c) then NoOp
  else [RemoveLid( C1), LookAt( C1), RemoveLid( C2), LookAt( C2),
    if Color( Chair, c)  $\wedge$  Color( can, c) then Paint( Table, can)
    else if Color( Table, c)  $\wedge$  Color( can, c) then Paint( Chair, can)
    else [Paint( Chair, C1), Paint( Table, C1)]]]
```

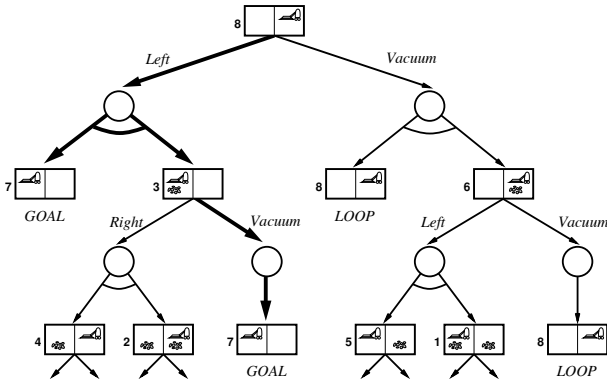
- Variables (e.g., c) are **existentially quantified**.

Games against nature

- Conditional plans should succeed regardless of circumstances
- Nesting conditional steps results in trees
- Similar to adversarial search, **games against nature**
- Game tree has state nodes and chance nodes where nature determines the outcome
- Definition of solution: A subtree with
 - a goal node at every leaf
 - specifies one action at each state node
 - includes every outcome at chance node
- AND-OR graphs can be used in similar way to the minimax algorithm (basic idea: find a plan for every possible result of a selected action)

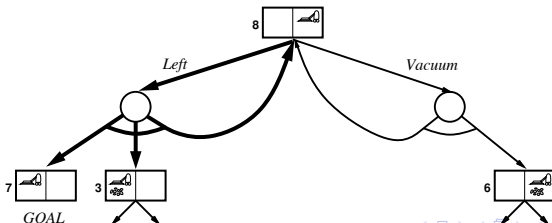
Example: "double Murphy" vacuum cleaner

- Vacuum cleaner sometimes deposits dirt at its destination when moving or when vacuuming in a clean square
- Solution: `[Left, if CleanL; then [] else Vacuum]`



Acyclic vs. cyclic solutions

- If identical state is encountered (on same path), terminate with failure (if there is an acyclic solution it can be reached from previous incarnation of state)
- However, sometimes all solutions are cyclic!
- E.g., “triple Murphy” (also) sometimes fails to move.
- Plan [*Left*, if *CleanL* then [] else *Vacuum*] now doesn't work
- Cyclic plan:
 [*L* : *Left*, if *AtR* then *L* elseif *CleanL* then [] else *Vacuum*]

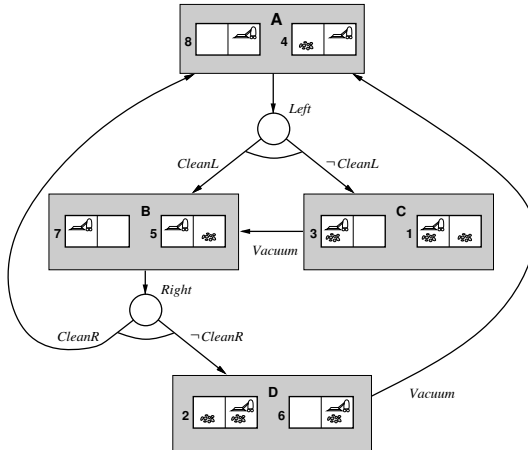


Nondeterminism and partially observable environments

“alternate double Murphy”:

- Vacuum cleaner can sense cleanliness of square it's in, but not the other square, and
- dirt can sometimes be left behind when leaving a clean square.
- Plan in fully observable world: “Keep moving left and right, vacuuming up dirt whenever it appears, until both squares are clean and in the left square”
- But now goal test cannot be performed!

Housework in partially observable worlds



Conditional planning, partial observability

- Basically, we can apply our AND-OR-search to belief states (rather than world states)
- Full observability is special case of partial observability with singleton belief states
- Is it really that easy?
- Not quite, need to describe
 - representation of belief states
 - how sensing works
 - representation of action descriptions

Summary

- Methods for planning and acting in the real world
- Dealing with indeterminacy
- Contingent planning: use percepts and conditionals to cater for all contingencies.
- Fully observable environments: AND-OR graphs, games against nature
- Partially observable environments: belief states, action and sensing
- Next time: **Planning and acting in the real world II**