Informatics 2D: Reasoning and Agents

Alex Lascarides

School of informatics

The University of Edinburgh

Lecture 19c: Planning and acting in the real world:
Searching for hierarchical plans

Where are we?

Hierarchical Plans

- You can represent high level actions (HLAs) using alternative *refinements*
- A valid high level plan (HLP) is a sequence of HLAs for which there is an implementation that achieves the goal.
- Now: **searching for a valid HLP**

Searching for Primitive Solutions

- The HLA plan library is a hierarchy:
  - (Ordered) Daughters to an HLA are the sequences of actions provided by one of its refinements;
  - Because a given HLA can have more than one refinement, there can be more than one node for a given HLA in the hierarchy.
- This hierarchy is essentially a search space of action sequences that conform to knowledge about how high-level actions can be broken down.
- So you can search this space for a plan!

Searching for Primitive Solutions: Breadth First

- Start your plan P with the HLA [*Act*],
- Take the first HLA A in P (recall that P is an *action sequence*).
- Do a breadth-first search in your hierarchical plan library, to find a refinement of A whose preconditions are satisfied by the outcome of the action in P that is prior to A.
- Replace A in P with this refinement.
- Keep going until your plan P has no HLAs and either:
  1. Your plan P's outcome is the goal, in which case return P; or
  2. Your plan P's outcome is not the goal, in which case backtrack,
     and if nowhere to backtrack then return *failure*.

Problems!

- Like forward search, you consider lots of irrelevant actions.
- The algorithm essentially refines HLAs right down to primitive actions so as to determine if a plan will succeed.
- This contradicts common sense!
- Sometimes you know an HLA will work *regardless* of how it's broken down!
- We don't need to know which route to take to SFOParking to know this plan works:

$$[Drive(Home, SFOParking), Shuttle(SFOParking, SFO)]$$

- We can capture this if we add to HLAs *themselves* a set of preconditions and effects.

Adding Preconditions and Effects to HLAs

- One challenge in specifying preconditions and effects of an HLA is that the HLA may have more than one refinement, each one with slightly different preconditions and effects!
    - If you refine *Go*(*Home*, *SFO*) with *Taxi* action: you need *Cash*.
    - If you refine it with *Drive*, you don't!
    - This difference may affect your choice on how to refine the HLA!
- Recall that an HLA achieves a goal if one of its refinements does this.
- And you can choose the refinement!

Getting Formal

- $s' \in \text{Reach}(s, h)$ iff s' is reachable from *at least one* of HLA h's refinements, given (initial) state s.

$$\text{Reach}(s, [h_1, h_2]) = \bigcup_{s' \in \text{Reach}(s, h_1)} \text{Reach}(s', h_2)$$

- HLP p achieves goal g given initial state s iff $\exists s'$ st

$$s' \models g \text{ and } s' \in \text{Reach}(s, p)$$

- So we should search HLPs to find a p with this relation to g, and then focus on refining it.
- But a pre-requisite to this algorithm is to define $\text{Reach}(s, h)$ for each h and s.
- In other words, we still need to determine how to represent effects (and preconditions) of HLAs...

Defining Reach

- A primitive action makes a fluent true, false, or leaves it unchanged.
- But with HLAs you sometimes get to *choose*, by choosing a particular refinement!
- We add new notation to reflect this:

  $\tilde{+}A$: you can possibly add A (or leave A unchanged)
  $\tilde{-}A$: you can possibly delete A (or leave A unchanged)
  $\tilde{\pm}A$: you can possibly add A, or
  possibly delete A (or leave A unchanged)

- You should now *derive* the correct preconditions and effects from its refinements!

Our SFO Example

$$
\begin{array}{l}
Refinment(Go(Home, SFO), \\
\quad \text{Precond:} At(Car, Home) \\
\quad \text{Steps:}[Drive(Home, SFOLongTermParking) \\
\qquad\qquad\quad Shuttle(SFOLongTermParking, SFO)])
\end{array}
$$

$$
\begin{array}{l}
Refinment(Go(Home, SFO), \\
\quad \text{Precond:} Cash, At(Home) \\
\quad \text{Steps:}[Taxi(Home, SFO)])
\end{array}
$$

The 'Primitive' Actions

$Action(Taxi(a, b),$
Precond: $Cash, At(Taxi, a)$
Effect: $\neg Cash, \neg At(Taxi, a), At(Taxi, b))$

$Action(Drive(a, b),$
Precond: $At(Car, a)$
Effect: $\neg At(Car, a), At(Car, b))$

$Action(Shuttle(a, b),$
Precond: $At(Shuttle, a)$
Effect: $\neg At(Shuttle, a), At(Shuttle, b))$

Deriving the Preconds and Effects of the HLA

- $\neg Cash$ is Effect of one HLA refinement, but not the other.
- So $\tilde{\neg} Cash$ in HLA Effect!

Not so Simple!

- Similar argument for $At(Car, SFOParking)$
- But you can't choose the combination: $\neg Cash \wedge At(Car, SFOParking)$
- Solution is to write approximate descriptions.

Approximate Descriptions

Optimistic Description: $\text{Reach}^+(s,h)$

- Take union of all possible outcomes from all refinements.
- So this includes $\tilde{\neg} Cash$ *and* $\tilde{+} At(Car, SFOParking)$.
- This overgenerates reachable states.

Pessimistic Description: $\text{Reach}^-(s,h)$

- Only states that satisfy effects from *all* refinements survive.
- So this does *not* include $\tilde{\neg} Cash$ or $\tilde{+} At(Car, SFOParking)$.
- This undergenerates reachable states.

$$\text{Reach}^-(s,h) \subseteq \text{Reach}(s,h) \subseteq \text{Reach}^+(s,h)$$

Algorithm for Finding a Plan

Two Important Facts:

1. If $\exists s' \in \text{Reach}^-(s,h)$ st $s' \models g$, you know h can succeed.
2. If $\neg\exists s' \in \text{Reach}^+(s,h)$ st $s' \models g$, you know h will fail!

The Algorithm:

- Do breadth first search as before.
- But now you can stop searching and implement instead when you reach an h where 1. is true.
- And you can drop h (and all its refinements) when 2. is true.
- If 1. and 2. are both false for the current h, then you don't know if h will succeed or fail, but you can find out by refining it.

Summary

- HLAs and HLPs
- Using refinements and preconditions and effects of primitive actions to *approximate* which states are reachable.
- Such approximate descriptions of HLAs help to inform search and when to refine an HLP so as to reach a goal.
- Next time: **Acting under Uncertainty**