CSE 3402: Intro to Artificial Intelligence
Reasoning about action

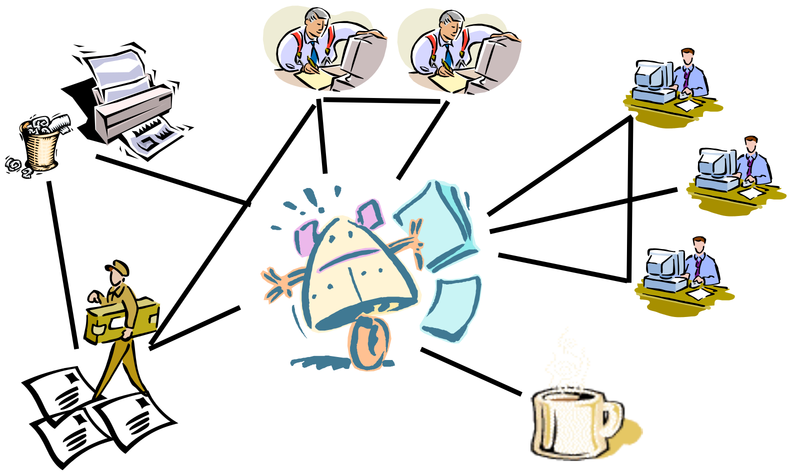
- Readings: Chapter 10.3

Why Planning

- Intelligent agents must operate in the world. They are not simply passive reasoners (Knowledge Representation, reasoning under uncertainty) or problem solvers (Search), they must also act on the world.
- We want intelligent agents to act in "intelligent ways". Taking purposeful actions, predicting the expected effect of such actions, composing actions together to achieve complex goals.

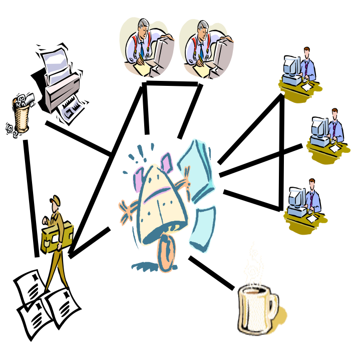
Why Planning

- E.g. if we have a robot we want robot to decide what to do; how to act to achieve our goals



A Planning Problem

- How to *change* the world to suit our needs
- Critical issue: we need to reason about ***what the world will be like*** after doing a few actions, not *just* what it is like now



GOAL: Craig has coffee
CURRENTLY: robot in mailroom, has no coffee, coffee not made, Craig in office, etc.
TO DO: goto lounge, make coffee,...

Planning

- Reasoning about what the world will be like after doing a few actions is similar to what we have already examined.
- However, now we want to reason about dynamic environments.
  - in(robby,Room1), lightOn(Room1) are true: will they be true after robby performs the action turnOffLights?
  - in(robby,Room1) is true: what does robby need to do to make in(robby,Room3) true?
- Reasoning about the effects of actions, and computing what actions can achieve certain effects is at the heart of decision making.

Planning under Uncertainty

- Our knowledge of the world probabilistic.
- Sensing is subject to noise (especially in robots).
- Actions and effectors are also subject to error (uncertainty in their effects).

Planning

- But for now we will confine our attention to the deterministic case.
- We will examine:
  - Determining the effects of actions.
  - finding sequences of actions that can achieve a desired set of effects.
    - This will in some ways be a lot like search, but we will see that representation also plays an important role.

Situation Calculus

- First we look at how to model dynamic worlds within first-order logic.
- The situation calculus is an important formalism developed for this purpose.
- Situation Calculus is a first-order language.
- Include in the domain of individuals a special set of objects called situations. Of these s_0 is a special distinguished constant which denotes the "initial" situation.

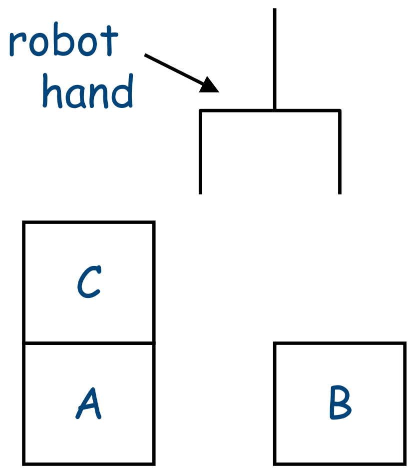
Situation Calculus

- Situations are used to index “states” of the world. When dealing with dynamic environments, the world has different properties at different points in time.
- e.g., in(robby,room1, s_0), ¬in(robby,room3,s_0)
  ¬in(robby,room3,s_1), in(robby,room1,s_1).
  - Different things are true in situation s_1 than in the initial situation s_0.
  - Contrast this with the previous kinds of knowledge we examined.

Fluents

- The basic idea is that properties that change from situation to situation (called fluents) take an extra situation argument.
  - clear(b) ➔ clear(b,s)
    - “clear(b)” is no longer statically true, it is true contingent on what situation we are talking about

Blocks World Example.



clear(c,s_0)
on(c,a,s_0)
clear(b,s_0)
handempty(s_0)

Actions in the Situation Calculus

- Actions are also part of language
  - A set of "primitive" action objects in the (semantic) domain of individuals.
  - In the syntax they are represented as functions mapping objects to primitive action objects.
  - pickup(X) function mapping blocks to actions
    - pickup(c) = "the primitive action object corresponding to 'picking up block c'
  - stack(X,Y)
    - stack(a,b) = "the primitive action object corresponding to 'stacking a on top of b'

Actions modify situations.

- There is a "generic" action application function do(A,S). do maps a primitive action and a situation to a new situation.
  - The new situation is the situation that results from applying A to S.

- do(pickup(c), s_0) = the new situation that is the result of applying action "pickup(c)" to the initial situation s_0.

What do Actions do?

- Actions affect the situation by changing what is true.
  - on(c,a,s_0); clear(a,do(pickup(c),s_0))
- We want to represent the effects of actions, this is done in the situation calculus with two components.

Specifying the effects of actions

- Action preconditions. Certain things must hold for actions to have a predictable effect.
  - pickup(c) this action is only applicable to situations S where "clear(c,S) ∧ handempty(S)" are true.

- Action effects. Actions make certain things true and certain things false.
  - holding(c, do(pickup(c), S))
  - ∀ X.¬handempty(do(pickup(X),S))

Specifying the effects of actions

- Action effects are conditional on their precondition being true.

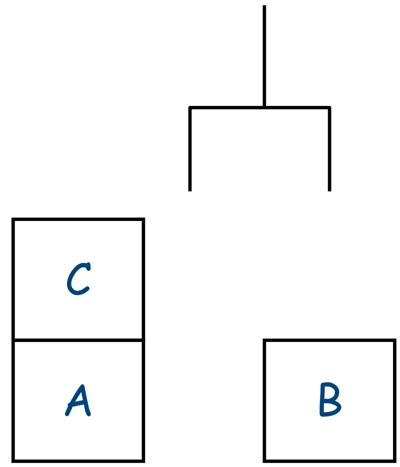
∀S,X.
ontable(X,S) ∧ clear(X,S) ∧ handempty(S)
→ holding(X, do(pickup(X),S))
∧ ¬handempty(do(pickup(X),S))
∧ ¬ontable(X,do(pickup(X,S))
∧ ¬clear(X,do(pickup(X,S)).

Reasoning with the Situation Calculus.

1. clear(c,s_0)
2. on(c,a,s_0)
3. clear(b,s_0)
4. ontable(a,s_0)
5. ontable(b,s_0)
6. handempty(s_0)

Query:
∃Z.holding(b,Z)
7. (¬holding(b,Z), ans(Z))

does there exists a situation in which we are holding b? And if so what is the name of that situation.



Resolution

- Convert "pickup" action axiom into clause form:

∀S,Y.
ontable(Y,S) ∧ clear(Y,S) ∧ handempty(S)
→ holding(Y, do(pickup(Y),S))
∧ ¬handempty(do(pickup(Y),S))
∧ ¬ontable(Y,do(pickup(Y,S))
∧ ¬clear(Y,do(pickup(Y,S)).

8. (¬ontable(Y,S), ¬clear(Y,S), ¬handempty(S), holding(Y,do(pickup(Y),S))
9. (¬ontable(Y,S), ¬clear(Y,S), ¬handempty(S), ¬handempty(do(pickup(X),S)))
10. (¬ontable(Y,S), ¬clear(Y,S), ¬handempty(S), ¬ontable(Y,do(pickup(Y,S)))
11. (¬ontable(Y,S), ¬clear(Y,S), ¬handempty(S), ¬clear(Y,do(pickup(Y,S)))

Resolution

12. R[8d, 7]{Y=b,Z=do(pickup(b),S)}
 (¬ontable(b,S), ¬clear(b,S), ¬handempty(S),
 ans(do(pickup(b),S)))
13. R[12a,5] {S=s_0}
 (¬clear(b,s_0), ¬handempty(s_0),
 ans(do(pickup(b),s_0)))
14. R[13a,3] {}
 (¬handempty(s_0), ans(do(pickup(b),s_0)))
15. R[14a,6] {}
 ans(do(pickup(b),s_0))

The answer?

- ans(do(pickup(b),s_0))
- This says that a situation in which you are holding b is called "do(pickup(b),s_0)"
- This name is informative: it tells you what actions to execute to achieve "holding(b)".

Two types of reasoning.

- In general we can answer questions of the form:
  $on(b,c,do(stack(b,c), do(pickup(b), s_0)))$

  $\exists S.\ on(b,c,S) \wedge on(c,a,S)$

- The first involves predicting the effects of a sequence of actions, the second involves computing a sequence of actions that can achieve a goal condition.

The Frame Problem

- Unfortunately, logical reasoning won't immediately yield the answer to these kinds of questions.

- e.g., query: $on(c,a,do(pickup(b),s_0))$?
  - is c still on a after we pickup b?
  - Intuitively it should be
  - Can logical reasoning reach this conclusion?

The Frame Problem

1. clear(c,s_0)
2. on(c,a,s_0)
3. clear(b,s_0)
4. ontable(a,s_0)
5. ontable(b,s_0)
6. handempty(s_0)

8. (¬ontable(Y,S), ¬clear(Y,S), ¬handempty(S),
   holding(Y,do(pickup(Y),S))
9. (¬ontable(Y,S), ¬clear(Y,S), ¬handempty(S),
   ¬handempty(do(pickup(X),S)))
10. (¬ontable(Y,S), ¬clear(Y,S), ¬handempty(S),
   ¬ontable(Y,do(pickup(Y,S)))
11. (¬ontable(Y,S), ¬clear(Y,S), ¬handempty(S),
   ¬clear(Y,do(pickup(Y,S)))
12. ¬on(c,a,do(pickup(b),s_0)) {QUERY)

Nothing can resolve with 12!

Logical Consequence

- Remember that resolution only computes logical consequences.
- We stated the effects of pickup(b), but did not state that it doesn't affect on(c,a).
- Hence there are models in which on(c,a) no longer holds after pickup(b) (as well as models where it does hold).

- The problem is that representing the non-effects of actions very tedious and in general is not possible.
  - Think of all of the things that pickup(b) does not affect!

The Frame Problem

- Finding an effective way of specifying the non-effects of actions, without having to explicitly write them all down is the frame problem.

- Very good solutions have been proposed, and the situation calculus has been a very powerful way of dealing with dynamic worlds:
  - logic based high level robotic programming languages

Computation Problems

- Although the situation calculus is a very powerful representation. It is not always efficient enough to use to compute sequences of actions.

- The problem of computing a sequence of actions to achieve a goal is "planning"
- Next we will study some less rich representations which support more efficient planning.