Knowledge representation and reasoning
Lecture 17: Reasoning about actions

Natasha Alechina
natasha.alechina@nottingham.ac.uk

Plan of the lecture

1. New topic: planning and reasoning about actions
2. Situation calculus
3. Preconditions and postconditions of actions
4. Frame problem
5. Using situation calculus to make plans

How to use reasoning to make plans

- This lecture: situation calculus (based mostly on Stuart Russell slides for Russell and Norvig, Artificial Intelligence)
- Next two lectures: planning

Using FOL to reason about actions

- Suppose we want AI program to reason about applicability, outcomes and the choice of actions
- What is missing currently is being able to express change
- Instead of describing a static world, we need to be able to talk about states of the world or situations

Situation Calculus

- Situation calculus is a dialect of FOL where situations (static states of the world) and actions are basic terms:
- variables over situations are denoted $s, s_1, s_2, \ldots,$.
- a distinguished initial situation is denoted by a constant S_0.
- actions are terms like $move(x, y, z)$ (move thing z to coordinates x, y) etc. Note that actions are also terms, not formulas: they denote an 'action' and are not true or false.
- a special function *do* takes an action and a situation and returns a new situation: $do(a, s)$ denotes a new situation which results from performing an action a in a situation s.

Fluents

- Predicates and functions whose values vary from situation to situation are called fluents.
- Last argument in a fluent is a situation:
- $\neg Holding(r, x, s) \wedge Holding(r, x, do(pickup(r, x), s))$ (a robot r is not holding x in s but is holding it in the situation resulting from s by picking x up).
- A distinguished fluent *Poss* says which actions are possible in a given situation:
  $Poss(pickup(r, blockA), S_0)$

Preconditions of actions

- A precondition is a condition which makes an action possible.
- It can be expressed by a formula, sometimes called precondition axiom.
- For example:

  $$\forall r \forall x \forall s\, (Poss(pickup(r, x), s) \equiv \forall z(\neg Holding(r, z, s) \wedge \neg Heavy(x) \wedge NextTo(r, x, s)))$$

  (a robot can pick x up if it is not holding anything else, x is not heavy, and the robot is next to it).

Postconditions or effects of actions

- A postcondition or effect of an action is a change resulting from executing the action.
- Formulas expressing postconditions are sometimes called effect axioms.
- For example, $\forall x \forall s \forall r (Fragile(x) \supset Broken(x, do(drop(r, x), s)))$
- Effect axioms for fluents which become true as a result of an action are called positive, and those where the fluent becomes false are called negative.

Frame axioms

- Classical planning assumption: actions are deterministic, and the world changes only as a result of clearly specified actions.
- For every action, we can also say which fluents it *does not* affect.
- The formulas which specify which properties are not changed as a result of an action are called frame axioms.
- For example,
  $\forall x \forall y \forall s \forall r$
  $(\neg Broken(x, s) \wedge (x \neq y \vee \neg Fragile(x)) \supset$
  $\neg Broken(x, do(drop(r, y), s)))$

Frame axioms continued

- Frame axioms do not logically follow from precondition and effect axioms.
- They are called frame axioms because they limit or frame the effects of actions.

Why do we need frame axioms 1

A typical kind of task in reasoning about actions is to check whether

- a certain sequence of actions $a_1, \ldots, a_n$ will succeed (bring about some desired state of the world)
- a certain sequence of actions is possible

In both cases, some relevant information about S_0 is given (which fluents hold in S_0).

Why do we need frame axioms 2

- The precondition and effects of actions are used to determine which fluents will be true in $do(a_n, do(a_{n-1}, \ldots do(a_1, S_0) \ldots)$.
- Some fluent may be a precondition of some action a_i which is true in S_0 and is unchanged by $a_1, \ldots, a_{i-1}$.
- However we cannot derive that it is unchanged from just the precondition and effect axioms for $a_1, \ldots, a_{i-1}$: need to also have explicit frame axioms.

Frame problem

- Frame problem is the problem of representing frame conditions coincisely (*not* with an axiom for each pair of action and fluent!).

Solution to the frame problem

- For each fluent $F(\bar{x}, s)$ (where $\bar{x}$ are all the free variables of the fluent) we collect together all positive effect axioms.

- For example, if $Broken(x, s)$ has two positive effect axioms:
  $\forall x \forall s\ (Fragile(x) \supset Broken(x, do(drop(x), s)))$

  $\forall x \forall s\ (Broken(x, do(break(x), s)))$

- and together they can be written as:
  $\forall x \forall a \forall s((Fragile(x) \wedge a = drop(x)) \vee (a = break(x))$
  $\supset Broken(x, do(a, s)))$

- In general, have an expression
  $\forall \bar{x} \forall a \forall s(\Pi_F(\bar{x}, a, s) \supset F(\bar{x}, do(a, s)))$

Solution to the frame problem continued

- Same for the negative effect axioms:
  $\forall \bar{x} \forall a \forall s (N_F(\bar{x}, a, s) \supset \neg F(\bar{x}, do(a, s)))$
- For example:
  $\forall \bar{x} \forall a \forall s (a = fix(x) \supset \neg Broken(x, do(a, s)))$

Solution to the frame problem continued

- Once we have a single formula Π_F for all actions which make $F(x, s)$ true and a single formula N_F for all actions which make F false, we can write explanation closure axioms:
  $\forall\bar{x}\forall a\forall s(\neg F(\bar{x}, s) \wedge F(\bar{x}, do(a, s)) \supset \Pi_F(\bar{x}, a, s))$
  $\forall\bar{x}\forall a\forall s(F(\bar{x}, s) \wedge \neg F(\bar{x}, do(a, s)) \supset N_F(\bar{x}, a, s))$
- They *replace all frame axioms* by saying that
  F only becomes true if Π_F holds (only certain actions in certain circumstances make F true)
  F only becomes false if N_F is true
- Π_F and N_F are short, and explanation axioms entail all the frame axioms (under the assumptions of deterministic actions and only change as a result of actions).

Successor state axioms

- If some additional assumptions hold, namely:
  1. no action has both a positive and negative effect on a fluent F,
  2. action terms can only be equal if they have the same action name applied to the same arguments

  then explanation closure axioms can be combined into a successor state axiom for a fluent:
  $\forall \bar{x} \forall a \forall s (F(\bar{x}, do(a, s)) \equiv \Pi_F(\bar{x}, a, s) \vee (F(\bar{x}, s) \wedge \neg N_F(\bar{x}, a, s)))$
- Under those assumptions, all that is needed to solve the frame problem and describe the actions and fluents completely are: precondition axioms and successor state axioms

Summary: Describing actions I

- "Effect" axiom—describe changes due to action
  $\forall s\ (AtGold(s) \supset Holding(Gold, do(Grab, s)))$
- "Frame" axiom—describe *non-changes* due to action
  $\forall s\ (HaveArrow(s) \supset HaveArrow(do(Grab, s)))$
- Frame problem: find an elegant way to handle non-change
  (a) representation—avoid frame axioms
  (b) inference—avoid repeated "copy-overs" to keep track of state
- Qualification problem: true descriptions of real actions require endless caveats—what if gold is slippery or nailed down or ...
- Ramification problem: real actions have many secondary consequences—what about the dust on the gold, wear and tear on gloves, ...

Describing actions II

- Successor-state axioms solve the representational frame problem
- Each axiom is about a *predicate* rather than about an action:

  P true afterwards $\equiv$ an action made P true or P true already and no action made P false
- For holding the gold: $\forall a \forall s Holding(Gold, do(a, s)) \equiv$
  $[(a = Grab \wedge AtGold(s)) \ \vee (Holding(Gold, s) \wedge a \neq Release)]$

Making plans in situation calculus

- Initial condition in KB:
  $At(Agent, [1, 1], S_0)$
  $At(Gold, [1, 2], S_0)$
- Query: $KB \models (?)\exists s\ Holding(Gold, s))$ i.e., in what situation will I be holding the gold?
- Answer: $s/do(Grab, do(Forward, S_0))$ i.e., go forward and then grab the gold

Making plans: A better way

- Represent plans as action sequences $[a_1, a_2, \ldots, a_n]$
- $doPlan(p, s)$ is the result of executing p in s
- Then the query $(KB, \exists p\ (Holding(Gold, doPlan(p, S_0))))$ has the solution $p/[Forward, Grab]$
- Definition of *doPlan* in terms of *do*:
  $\forall s doPlan([\,], s) = s\ \forall a \forall p \forall s (doPlan([a|p], s) = doPlan(p, do(a, s))$
- Planning systems are special-purpose reasoners designed to do this type of inference more efficiently than a general-purpose reasoner

Summary

- Situation calculus provides conventions for describing actions and change in FOL
- can formulate planning as inference on a situation calculus KB
- Next lecture: Planning.
  Brachman and Levesque, Chapter 15.
  Russell and Norvig, 3rd ed., Chapter 10.1-10.2. (or other editions, Classical Planning).