Higher-Order Logic

Higher-order logic

- FOL only allows to quantify over variables, and variables can only range over objects.
- HOL allows us to quantify over relations
- Example: (quantify over functions)

  "two functions are equal iff they produce the same value for all arguments"

  $\forall f\ \forall g\ (f = g) \leftrightarrow (\forall x\ f(x) = g(x))$
- Example: (quantify over predicates)

  $\forall r$ transitive(r) $\rightarrow (\forall xyz)\ r(x,y) \wedge r(y,z) \rightarrow r(x,z))$
- More expressive, but **undecidable**. (there isn't an effective algorithm to decide whether all sentences are valid)
  - First-order logic is decidable only when it uses predicates with only one argument.

Expressing uniqueness

- Sometimes we want to say that there is a single, unique object that satisfies a certain condition
- “There exists a unique x such that king(x) is true”
  - $\exists$x king(x) $\wedge$ $\forall$y (king(y) $\rightarrow$ x=y)
  - $\exists$x king(x) $\wedge$ $\neg\exists$y (king(y) $\wedge$ x$\neq$y)
  - $\exists$! x king(x)
- “Every country has exactly one ruler”
  - $\forall$c country(c) $\rightarrow$ $\exists$! r ruler(c,r)
- Iota operator: “ι x P(x)” means “the unique x such that p(x) is true”
  - “The unique ruler of Freedonia is dead”
  - dead(ι x ruler(freedonia,x))

Notational differences

- **Different symbols** for *and, or, not, implies, ...*
  - $\forall$ $\exists$ $\Rightarrow$ $\Leftrightarrow$ $\wedge$ $\vee$ $\neg$ $\bullet$ $\supset$
  - p v (q ^ r)
  - p + (q * r)
  - etc
- **Prolog**

  cat(X) :- furry(X), meows (X), has(X, claws)
- **Lispy notations**

```
(forall ?x (implies (and (furry ?x)
                         (meows ?x)
                         (has ?x claws))
                    (cat ?x)))
```

Situation calculus

- A **situation** is a snapshot of the world at an interval of time during which nothing changes
- Every true or false statement is made with respect to a particular situation.
  - Add **situation variables** to every predicate.
  - at(Agent,1,1) becomes at(Agent,1,1,s0): at(Agent,1,1) is true in situation (i.e., state) s0.
  - Alternatively, add a special 2nd-order predicate, **holds(f,s),** that means "f is true in situation s." E.g., holds(at(Agent,1,1),s0)
- Add a new function, **result(a,s),** that maps a situation s into a new situation as a result of performing action a. For example, result(forward, s) is a function that returns the successor state (situation) to s
- Example: The action agent-walks-to-location-y could be represented by
  - $(\forall x)(\forall y)(\forall s)\ (at(Agent,x,s) \wedge \neg onbox(s)) \rightarrow at(Agent,y,result(walk(y),s))$

Deducing hidden properties

- From the perceptual information we obtain in situations, we can **infer properties of locations**

  $\forall l,s\ at(Agent,l,s) \wedge Breeze(s) \rightarrow Breezy(l)$

  $\forall l,s\ at(Agent,l,s) \wedge Stench(s) \rightarrow Smelly(l)$

- Neither Breezy nor Smelly need situation arguments because pits and Wumpuses do not move around

Deducing hidden properties II

- We need to write some rules that relate various aspects of a single world state (as opposed to across states)
- There are two main kinds of such rules:
  - **Causal rules** reflect the assumed direction of causality in the world:
    $(\forall l1,l2,s)\ At(Wumpus,l1,s) \wedge Adjacent(l1,l2) \rightarrow Smelly(l2)$
    $(\forall\ l1,l2,s)\ At(Pit,l1,s) \wedge Adjacent(l1,l2) \rightarrow Breezy(l2)$

    Systems that reason with causal rules are called **model-based reasoning systems**
  - **Diagnostic rules** infer the presence of **hidden properties** directly from the percept-derived information. We have already seen two diagnostic rules:
    $(\forall\ l,s)\ At(Agent,l,s) \wedge Breeze(s) \rightarrow Breezy(l)$
    $(\forall\ l,s)\ At(Agent,l,s) \wedge Stench(s) \rightarrow Smelly(l)$

Representing change: The frame problem

- **Frame axioms**: If property x doesn't change as a result of applying action a in state s, then it stays the same.
  - $On(x, z, s) \wedge Clear(x, s) \rightarrow$
    $On(x, table, Result(Move(x, table), s)) \wedge$
    $\neg On(x, z, Result(Move(x, table), s))$
  - $On(y, z, s) \wedge y \neq x \rightarrow On(y, z, Result(Move(x, table), s))$
  - The proliferation of frame axioms becomes very cumbersome in complex domains

The frame problem II

- **Successor-state axiom**: General statement that characterizes every way in which a particular predicate can become true:
  - Either it can be **made true**, or it can **already be true and not be changed**:
  - On (x, table, Result(a,s)) $\leftrightarrow$
    [On (x, z, s) $\wedge$ Clear (x, s) $\wedge$ a = Move(x, table)] $\wedge$
    [On (x, table, s) $\wedge$ a $\neq$ Move (x, z)]
- In complex worlds, where you want to reason about longer chains of action, even these types of axioms are too cumbersome
  - Planning systems use special-purpose inference methods to reason about the expected state of the world at any point in time during a multi-step plan

Qualification problem

- Qualification problem:
  - How can you possibly characterize every single effect of an action, or every single exception that might occur?
  - When I put my bread into the toaster, and push the button, it will become toasted after two minutes, unless…
    - The toaster is broken, or…
    - The power is out, or…
    - I blow a fuse, or…
    - A neutron bomb explodes nearby and fries all electrical components, or…
    - A meteor strikes the earth, and the world we know it ceases to exist, or…

Ramification problem

- Similarly, it's just about impossible to characterize every side effect of every action, at every possible level of detail:
  - When I put my bread into the toaster, and push the button, the bread will become toasted after two minutes, and…
    - The crumbs that fall off the bread onto the bottom of the toaster over tray will also become toasted, and…
    - Some of the aforementioned crumbs will become burnt, and…
    - The outside molecules of the bread will become "toasted," and…
    - The inside molecules of the bread will remain more "breadlike," and…
    - The toasting process will release a small amount of humidity into the air because of evaporation, and…
    - The heating elements will become a tiny fraction more likely to burn out the next time I use the toaster, and…
    - The electricity meter in the house will move up slightly, and…

Knowledge engineering!

- Modeling the “right” conditions and the “right” effects at the “right” level of abstraction is very difficult
- Knowledge engineering (creating and maintaining knowledge bases for intelligent reasoning) is an entire field of investigation
- Many researchers hope that automated knowledge acquisition and machine learning tools can fill the gap:
  - Our intelligent systems should be able to **learn** about the conditions and effects, just like we do!
  - Our intelligent systems should be able to learn when to pay attention to, or reason about, certain aspects of processes, depending on the context!

Preferences among actions

- A problem with the Wumpus world knowledge base that we have built so far is that it is difficult to decide which action is best among a number of possibilities.
- For example, to decide between a forward and a grab, axioms describing when it is OK to move to a square would have to mention glitter.
- This is not modular!
- We can solve this problem by **separating facts about actions from facts about goals**. This way our **agent can be reprogrammed just by asking it to achieve different goals**.

Preferences among actions

- The first step is to describe the desirability of actions independent of each other.
- In doing this we will use a simple scale: actions can be Great, Good, Medium, Risky, or Deadly.
- Obviously, the agent should always do the best action it can find:

$(\forall a,s)\ \mathrm{Great}(a,s) \rightarrow \mathrm{Action}(a,s)$

$(\forall a,s)\ \mathrm{Good}(a,s) \wedge \neg(\exists b)\ \mathrm{Great}(b,s) \rightarrow \mathrm{Action}(a,s)$

$(\forall a,s)\ \mathrm{Medium}(a,s) \wedge (\neg(\exists b)\ \mathrm{Great}(b,s) \vee \mathrm{Good}(b,s)) \rightarrow \mathrm{Action}(a,s)$

...

Preferences among actions

- We use this action quality scale in the following way.
- Until it finds the gold, the basic strategy for our agent is:
  – Great actions include picking up the gold when found and climbing out of the cave with the gold.
  – Good actions include moving to a square that's OK and hasn't been visited yet.
  – Medium actions include moving to a square that is OK and has already been visited.
  – Risky actions include moving to a square that is not known to be deadly or OK.
  – Deadly actions are moving into a square that is known to have a pit or a Wumpus.

Goal-based agents

- Once the gold is found, it is necessary to change strategies. So now we need a new set of action values.
- We could encode this as a rule:
  - $(\forall s)$ Holding(Gold,s) $\rightarrow$ GoalLocation([1,1]),s)
- We must now decide how the agent will work out a sequence of actions to accomplish the goal.
- Three possible approaches are:
  - **Inference**: good versus wasteful solutions
  - **Search**: make a problem with operators and set of states
  - **Planning**: to be discussed later

Thank You