Logical Representations

1 Motivation

We're looking at agents that think ahead. We've seen that this is a process of simulation, but also of search. And we've spent some time looking at *efficiency* issues. But we've done all of this while using *iconic representations* of the states. And the problem with these representations is that, for agents that live in 'rich' environments (e.g. physical buildings or complex virtual worlds such as the Internet), iconic representations are often not *expressive* enough. They do not lend themselves well to expressing the following kinds of knowledge about states of the world:

- General laws and other statements that apply across many objects. E.g.:
  - All trees are obstacles to lawn mowers.
  - All unsupported objects fall to the ground.
  - If you change the location of any container, then any objects in the container change location too.
- Indefinite statements (e.g. disjunctions, negative statements that say what isn't true without saying what is true, and existentially quantified statements that fail to identify the objects involved). E.g.:
  - The key is either on the desk or in the drawer.
  - There is no tree in front of the lawn mower.
  - Some of the books are on the table.

It's very hard to come up with iconic representations to handle these, and in those cases where iconic representations are possible, they are often far from concise.

By contrast, *logical representations* can be expressive enough. What's more, logical representations allow *inference* of properties of the world that have not been stated explicitly.

So, in future lectures, we're going to use logical representations in our agent. States of the world will be represented by sets of statements in logic. The preconditions and effects of operators will also be represented using logic. And the goal condition will be expressed in logic too.

But, additionally, we will be able to express *background knowledge* about the world in logic too (e.g. general laws, commonsense knowledge about objects and their properties, and so on.) We will store this knowledge in a *knowledge base*. Then, to see whether an operator's precondition is true in some state, or to see whether the goal condition is true, might no longer be simply a case of matching the two; it might be necessary to do some reasoning (inference) to see whether the precondition/goal is true.

But before we get on to any of that, we start with an introduction to the syntax and semantics of logic.

2 What is a Logic?

I use the words *logic* and *knowledge representation language* (KRL) interchangeably. Every logic (or KRL) has:

Syntax: The legitimate symbols of the language and the rules that define the legitimate configurations of these symbols.

Semantics: Rules that define the meanings of the symbols, the meanings of configurations of symbols and certain relationships that hold between different configurations (e.g. whether one contradicts another).

Proof theory: Rules that define the inferences that can be drawn. (These rules use *syntactic operations only*.)

There are numerous logics (KRLs), many of them invented by AI researchers. They may differ in their expressiveness but also in the efficiency of their proof theory. (Generally, the more expressive, the less efficient.) AI researchers will choose to use a logic that gives them the best trade-off of expressiveness and efficiency they need for the task in hand.

We're going to look at one particular logic, *first-order predicate logic*. (Other ways of referring to this logic are: FOPL, first-order logic, FOL, first-order predicate calculus and FOPC.) This is the main choice in AI. It is maximally expressive. However:

- Some concepts may not be expressed especially concisely with this logic. Hence, AI researchers and others have invented more exotic logics to express these concepts more concisely. But statements in these more exotic logics can always be 'reified' into FOPL.
- Reasoning with FOPL may not always be especially efficient so, wherever possible, AI researchers try to manage with sublanguages of FOPL, where the full expressiveness is not exploited.

3 The Syntax of FOPL

Statements in FOPL are called *well-formed formulae* (wffs). To define wffs, we must define *terms* and *atoms*.

3.1 Terms

1. Every *constant symbol* is a term, e.g. a, b, c, *clyde*, *gertie*
2. Every *variable* is a term, e.g. x, y, z, x_0, x_1, x_2
3. For any *function symbol* F of arity n and any n terms $T_1, T_2, \ldots, T_n$, then
$$F(T_1, T_2, \ldots, T_n)$$
is a term. E.g. if f is a function symbol of arity 1 and g is a function symbol of arity 2, then $f(a), g(b, c), g(f(a), c)$ and $g(f(f(a)), x)$ are terms.
4. Nothing else is a term.

3.2 Atoms

1. For any *predicate symbol* P of arity n and any n terms $T_1, T_2, \ldots, T_n$, then
$$P(T_1, T_2, \ldots, T_n)$$
is an atom. E.g. if p and *elephant* are predicate symbols of arity 1, and q and *likes* are predicate symbols of arity 2, then $p(a)$, *elephant*(*clyde*), $q(a, f(a))$ and *likes*(*clyde*, *gertie*) are atoms.
2. Nothing else is an atom.

Note that predicate symbols, function symbols, variables and constant symbols should be drawn from disjoint sets to avoid confusion.

3.3 Well-Formed Formulae (wffs)

1. Every atom is a wff.
2. For any variable X and any wff W, $(\forall XW)$ and $(\exists XW)$ are wffs. They are referred to as *quantified wffs*. $(\forall XW)$ is said to be universally quantified; $(\exists XW)$ is said to be existentially quantified.
3. For any wffs W, W_1 and W_2, the following are also wffs: $(\neg W)$ (negation), $(W_1 \wedge W_2)$ (conjunction), $(W_1 \vee W_2)$ (disjunction), $(W_1 \Rightarrow W_2)$ (conditional) and $(W_1 \Leftrightarrow W_2)$ (biconditional).
4. Nothing else is a wff.

E.g. the following are wffs: *elephant(clyde)*, $\neg$*likes*(*clyde*, *gertie*), *elephant*(*clyde*) $\vee$ *likes*(*clyde*, *gertie*) and $\forall x(p(x) \Rightarrow (q(b,c) \wedge q(f(a),x)))$.

While rules 2 and 3 above introduce parentheses around wffs, we will only write them when they are helpful for disambiguation purposes. In wffs where we can do without them, we won't write them.

Question.

a	is a constant symbol
s	is a unary function symbol
u	is a binary function symbol
e	is a unary predicate symbol
g	is a binary predicate symbol

Which of these are wffs?

1. $\forall x\, e(x)$
2. $\exists y(g(a,y) \wedge e(x))$
3. $\forall x \exists y(g(x,a))$
4. $\forall x \exists y(g(x,a) \vee g(y,x))$
5. $g(s(a,a),a)$
6. $e(u(a,s(a))) \Rightarrow g(a,a)$
7. $\exists x(u(a,x) \vee s(x))$
8. $\forall x(e(x) \vee e(s(x)))$
9. $\neg(e(x) \wedge e(s(x)))$
10. $\forall x\, e(s(a))$
11. $g(a,e(a))$
12. $\forall x(g(u(x,x),s(x)) \vee g(s(a),u(a,a))$

4 Quantifier Scope

The *scope* of a quantifier ($\forall, \exists$) is the subwff to which the quantifier applies. With plenty of parentheses, the scope will be apparent. An occurrence of a variable in a wff is *bound* if it falls within the scope of a quantifier for that variable. Otherwise, that occurrence of the variable is *free*.

For example:

- $\exists x(p(x)) \vee (q \Rightarrow \forall y(r(x,y)))$
  the scope of $\exists x$ is $p(x)$
  the scope of $\forall y$ is $r(x,y)$
  the first occurrence of x is bound; the second is free
  the only occurrence of y is bound.
- $\exists x(p(x) \vee (q \Rightarrow \forall y(r(x,y))))$
  the scope of $\exists x$ is $p(x) \vee (q \Rightarrow \forall y(r(x,y)))$
  the scope of $\forall y$ is $r(x,y)$
  all occurrences of x and y are bound.

What happens if there aren't enough parentheses? Different books use different conventions, so be careful. I'll try to put enough parentheses in, but if I don't, then go for the narrowest scope possible. E.g. in

$$\exists x\, p(x) \vee (q \Rightarrow \forall y\, r(x,y))$$

the scope of x is $p(x)$ and that of y is $r(x,y)$.

In this course, we will not write wffs that have free variables.

5 More Terminology

Unfortunately, some further terminology will be useful:

- In a conjunction, $W_1 \wedge W_2$, W_1 and W_2 are called the *conjuncts*.
- In a disjunction, $W_1 \vee W_2$, W_1 and W_2 are called the *disjuncts*.
- In a conditional, $W_1 \Rightarrow W_2$, W_1 is called the *antecedent* and W_2 is called the *consequent*.
- We know what an atom is: a predicate symbols with its arguments (each of which is a term). A *literal* is an atom or a negated atom, e.g. *likes*(*clyde*, *gertie*), $\neg$*likes*(*clyde*, *gertie*).
- We use the word *ground* to refer to an expression with no variables in it. So we talk of a *ground term* (e.g. $f(a,g(b))$), a *ground atom* (e.g. $p(a,f(a))$), a *ground literal* (e.g. $p(a,f(a))$ and $\neg p(a,f(a))$), and a *ground wff* (e.g. $p(a,f(a) \wedge q(b,c))$).

Exercise

In these questions, p and q are binary predicate symbols, f is a unary function symbol, g is a binary function symbol, and a and b are constant symbols. x, y, z and subscripted versions of these will be used as variables.

1. Indicate, by writing *Legal* or *Illegal*, which of the following are well-formed formulae of first-order predicate logic. Where you think they are *Illegal*, briefly explain why.

   (a) $\forall x(p(a, f(f(g(a, b)))) \Rightarrow \exists y\, p(a, f(f(g(b, a)))))$

   (b) $\forall x \forall y(p(a, b) \vee \forall x(q(f(a), f(b)), p(b, a)))$

   (c) $\forall x \forall y(p(a, b) \vee \forall x(q(f(a, b)) \wedge p(b, a)))$

2. Indicate, by writing *Yes* or *No* whether the following wffs of first-order predicate logic contain *free variables*. Where you think that they do, also indicate which variable occurrences within the wffs are free.

   (a) $\forall x(p(a, f(x)) \wedge p(x, b)) \Rightarrow p(x, f(g(b, x)))$

   (b) $\exists y(q(y, y) \Rightarrow ((\exists x\, q(y, x)) \vee \forall x\, q(x, y)))$

   (c) $\forall x(\exists x(\forall x(\exists x\, (p(x, x) \Rightarrow q(x, x)) \wedge p(x, x))) \vee q(x, x)) \Leftrightarrow \neg p(x, x)$