First Order Predicate Logic

CIS 32

Functionalia

Demos?

HW 3 is out on the web-page.

Today:

- Predicate Logic
- Constructing the Logical Agent

Predicate Logic

- *First-order predicate logic*
- More expressive than propositional logic.
- Consider the following argument:
  - *all monitors are ready;*
  - *X12 is a monitor;*
  - *therefore X12 is ready.*
- Sense of this argument *cannot* be captured in propositional logic.
- Propositional logic is too *coarse grained* to allow us to represent and reason about this kind of statement.

Syntax

- We shall now introduce a generalization of propositional logic called first-order logic (FOL). This new logic affords us much greater expressive power.

- **Definition**: The alphabet of FOPL contains:

  1. a set of ***constants***;
  2. a set of ***variables***;
  3. a set of ***function symbols***;
  4. a set of ***predicates symbols***;
  5. the ***connectives*** $\vee$, $\neg$;
  6. the ***quantifiers*** $\forall$, $\exists$, $\exists_1$;
  7. the punctuation symbols), (.

Terms : Constants

- The basic components of FOL are called *terms*.
- Essentially, a term is an object that denotes some object other than $\top$ or $\bot$.
- The simplest kind of term is a ***constant***.
- A value such as 8 is a constant.
- The ***denotation*** of this term is the number 8.
- Note that a constant and the number it denotes are different!
- Aliens don’t write “8” for the number 8, and nor did the Romans.

Terms : Variables

- The second simplest kind of term is a *variable*.
- A variable can stand for anything in the *domain of discourse*.
- The domain of discourse (usually abbreviated to domain) is the set of all objects under consideration.
- Sometimes, we assume the set contains "everything".
- Sometimes, we explicitly *give* the set, and *state* what variables/constants can stand for.

Terms : Functions

- We can now introduce a more complex class of terms — *functions*.

- The idea of functional terms in logic is similar to the idea of a function in programming: recall that in programming, a function is a procedure that takes some arguments, and *returns a value*.

In C:

```
T myfunction( T1 a1, ..., Tn an ) {
   ...
}
```

this function takes *n* arguments; the first is of type **`T1`**, the second is of type **`T2`**, and so on. The function returns a value of type **`T`**.

- In FOL, we have a set of *function symbols*; each symbol corresponds to a particular function. (It denotes some function.)

Function: *arity*

- Each function symbol is associated with a number called its *arity*. This is just the number of arguments it takes.

- A *functional term* is built up by *applying* a function symbol to the appropriate number of terms.

- Formally …

Definition: Let *f* be an arbitrary function symbol of arity *n*. Also, let $\tau_1, \ldots, \tau_n$ be terms. Then

$$f(\tau_1, \ldots, \tau_n)$$

is a functional term.

Function : *arity*

• All this sounds complicated, but isn't. Consider a function *plus*, which takes just two arguments, each of which is a number, and returns the first number added to the second.

Then:

– *plus*(2, 3) is an acceptable functional term;

– *plus*(0, 1) is acceptable;

– *plus*(*plus*(1, 2), 4) is acceptable;

– *plus*(*plus*(*plus*(0, 1), 2), 4) is acceptable;

Functions

- In maths, we have many functions; the obvious ones are

$$+ \quad - \quad / \quad * \quad \sqrt{\quad} \quad \sin \quad \cos \quad \ldots$$

- The fact that we write

$$2 + 3$$

instead of something like

$$\mathit{plus}(2, 3)$$

is just convention, and is not relevant from the point of view of logic; all these are functions in exactly the way we have defined.

Function

- Using functions, constants, and variables, we can build up *expressions*, e.g.:

$$(x + 3) * \sin 90$$

(which might just as well be written

$$\mathit{times}(\mathit{plus}(x, 3), \mathit{sin}(90))$$

for all it matters.)

Predicates

- In addition to having terms, FOL has *relational operators*, which capture *relationships* between objects.

- The language of FOL contains *predicate symbols*.

- These symbols stand for *relationships between objects*.

- Each predicate symbol has an associated *arity* (number of arguments).

- **Definition**: Let *P* be a predicate symbol of arity *n*, and $\tau_1, \ldots, \tau_n$ are terms.

Then

$$P(\tau_1, \ldots, \tau_n)$$

is a predicate, which will either be $\top$ or $\perp$ under some interpretation.

• **EXAMPLE**. Let *gt* be a predicate symbol with the intended interpretation '*greater than*'. It takes two arguments, each of which is a natural number.

Then:

– *gt*(4, 3) is a predicate, which evaluates to $\top$;

– *gt*(3, 4) is a predicate, which evaluates to $\perp$.

• The following are standard mathematical predicate symbols:

$$> \; < \; = \; \neq \; \geq \; \leq \; \ldots$$

• The fact that we are normally write $x > y$ instead of $gt(x, y)$ is just convention.

- We can build up more complex predicates using the connectives of propositional logic:

$$(2 > 3) \wedge (6 = 7) \vee (\sqrt{4} = 2)$$

- So a predicate just expresses a relationship between some values.
- What happens if a predicate contains *variables*: can we tell if it is true or false?

    Not usually; we need to know an *interpretation* for the variables.

- A predicate that contains no variables is a proposition.

Properties

- Predicates of *arity* 1 are called properties.
- **EXAMPLE**. The following are properties:

$$Man(x)$$

$$Mortal(x)$$

$$Malfunctioning(x).$$

- We interpret $P(x)$ as saying x is in the set P.
- Predicate that have arity 0 (i.e., take no arguments) are called *primitive propositions*.

These are identical to the primitive propositions we saw in propositional logic.

Quantifiers

- We now come to the central part of first order logic: *quantification*.
- Consider trying to represent the following statements:
  - *all men have a mother ;*
  - *every positive integer has a prime factor.*
- We can't represent these using the apparatus we've got so far; we need *quantifiers*.

Quantifiers

- We use three quantifers:

  $\forall$ — the *universal quantifier* ;

  - is read 'for all…'

  $\exists$ — the *existential quantifier* ;

  - is read 'there exists…'

  $\exists_1$ — the unique quantifier ;

  - is read 'there exists a unique…'

- The simplest form of quantified formula is as follows:

  quantifier variable $\cdot$ *predicate*

where

- *quantifier* is one of $\forall$, $\exists$, $\exists_1$;
- *variable* is a variable;
- and *predicate* is a predicate.

Examples

- $\forall x \cdot Man(x) \Rightarrow Mortal(x)$

  'For all *x*, if *x* is a man, then *x* is mortal.'

  (i.e. all men are mortal)

- $\forall x \cdot Man(x) \Rightarrow \exists_1 y \cdot Woman(y) \wedge MotherOf(x, y)$

  'For all *x*, if *x* is a man, then there exists exactly one *y* such that *y* is a woman and the mother of *x* is *y*.'

  (i.e., every man has exactly one mother).

Examples

- $\exists m \cdot Monitor(m) \wedge MonitorState(m, ready)$

  'There exists a monitor that is in a ready state.'

- $\forall r \cdot Reactor(r) \Rightarrow \exists_1 t \cdot (100 \leq t \leq 1000) \wedge temp(r) = t$

  'Every reactor will have a temperature in the range 100 to 1000.'

- $\exists n \cdot posInt(n) \wedge n = (n * n)$

  "Some positive integer is equal to its own square."

- $\exists c \cdot EUcountry(c) \wedge Borders(c, Albania)$

  "Some EU country borders Albania."

- $\forall m, n \cdot Person(m) \wedge Person(n) \Rightarrow \neg Superior(m, n)$

  "No person is superior to another."

- $\forall m \cdot Person(m) \Rightarrow \neg \exists n \cdot Person(n) \wedge Superior(m, n)$

  (same as previous)

Domains & Interpretations

- Suppose we have a formula $\forall x \cdot P(x)$.

  What does x *range* over?

  Physical objects, numbers, people, times, . . . ?

- Depends on the *domain* that we intend.
- Often, we *name* a domain to make our intended interpretation clear.

Example of Domains

- Suppose our intended interpretation is the positive integers.
Suppose $>, +, *, \ldots$ have the usual mathematical interpretation.

- Is this formula satisfiable under this interpretation?

$$\exists n \cdot n = (n * n)$$

- Now suppose that our domain is all living people,
and that $*$ means "is the child of".

- Is the formula satisfiable under this interpretation?

Conjunctions

- Note that universal quantification is similar to *conjunction*.

  Suppose the domain is the numbers {2, 4, 6}. Then

$$\forall n \;\cdot\; Even(n)$$

is the same as

$$Even(2) \wedge Even(4) \wedge Even(6).$$

- Existential quantification is similar to disjunction. Thus with the same domain,

$$\exists n \;\cdot\; Even(n)$$

is the same as

$$Even(2) \vee Even(4) \vee Even(6)$$

- The universal and existential quantifiers are in fact *duals* of each other:

$$\forall x \cdot P(x) \Leftrightarrow \neg \exists x \cdot \neg P(x)$$

Saying that everything has some property is the same as saying that there is nothing that does not have the property.

$$\exists x \cdot P(x) \Leftrightarrow \neg \forall x \cdot \neg P(x)$$

Saying that there is something that has the property is the same as saying that its not the case that everything doesn't have the property.

Validity

- In propositional logic, we saw that some formulae were tautologies — they had the property of being true under all interpretations.
- We also saw that there was a procedure which could be used to tell whether any formula was a tautology — this procedure was the truth-table method.
- A formula of FOL that is true under all interpretations is said to be *valid*.
- So in theory we could check for validity by writing down all the possible interpretations and looking to see whether the formula is true or not.

Decidability and Undecidability

- Unfortuately in general we can't use this method.
- Consider the formula:

$$\forall n \cdot Even(n) \Rightarrow \neg Odd(n)$$

- There are an infinite number of interpretations.
- Is there any other procedure that we can use, that will be guaranteed to tell us, in a finite amount of time, whether a FOL formula is, or is not, valid?
- The answer is *no*.
- FOL is for this reason said to be *undecidable*.

Proof in FOL

- Proof in FOL is similar to propositional logic (PL); we just need an extra set of rules, to deal with the quantifiers.
- FOL inherits all the rules of PL.
- To understand FOL proof rules, need to understand *substitution*.
- The most obvious rule, for ∀-E.

Tells us that if everything in the domain has some property, then we can infer that any particular individual has the property.

$$\frac{\vdash \forall x \cdot \phi(x);}{\vdash \phi(a)} \;\forall\text{-E} \qquad \text{for any } a \text{ in the domain}$$

Going from *general* to *specific*

Example 1

Let's use ∀-E to get the Socrates example out of the way.

$$Man(s); \forall x \cdot Man(x) \Rightarrow Mortal(x)$$
$$\vdash Mortal(s)$$

1.	$Man(s)$	Given
2.	$\forall x \cdot Man(x) \Rightarrow Mortal(x)$	Given
3.	$Man(s) \Rightarrow Mortal(s)$	2, ∀-E
4.	$Mortal(s)$	1, 3, ⇒-E

- Existential Introduction Rule 1 (∃-I(1)).
- We can also go from the general to the slightly less specific!

$$\frac{\vdash \forall x \cdot \phi(x);}{\vdash \exists x \cdot \phi(x)} \; \exists\text{-I}(1) \qquad \text{if domain not empty}$$

Note the *side condition*.

The ∃ quantifier *asserts the existence* of at least one object.

The ∀ quantifier does not.

- Existential Introduction Rule 2 (∃-I(2)).

- We can also go from the very specific to less specific.

$$\frac{\vdash \phi(a);}{\vdash \exists x \cdot \phi(x)}\ \exists\text{-I}(2)$$

- In other words once we have a concrete example, we can infer there exists something with the property of that example.

- We often informally make use of arguments along the lines…

1. We know somebody is the murderer.
2. Call this person *a*.
3. …

(Here, *a* is called a *Skolem constant*.)

- We have a rule which allows this, but we have to be careful how we use it!

$$\frac{\vdash \exists x \cdot \phi(x);}{\vdash \phi(a)} \ \exists\text{-E} \quad a \text{ doesn't occur elsewhere}$$

- Here is an invalid use of this rule:

1.	$\exists x \cdot Boring(x)$	Given
2.	$Lecture(AI)$	Given
3.	$Boring(AI)$	1, $\exists$-E

- (The conclusion may be true, the argument isn't sound.)

Example 2

1. Everybody is either happy or rich.

2. Simon is not rich.

3. Therefore, Simon is happy.

Predicates:

– $H(x)$ means x is happy;

– $R(x)$ means x is rich.

• *Formalisation*:

$\forall x \; H(x) \vee R(x); \neg R(Simon) \vdash H(Simon)$

Proof

1.	$\forall x. H(x) \vee R(x)$	Given
2.	$\neg R(Simon)$	Given
3.	$H(Simon) \vee R(Simon)$	1, $\forall$-E
4.	$\neg H(Simon) \Rightarrow R(Simon)$	3, defn $\Rightarrow$
5.	$\neg H(Simon)$	Assumption |
6.	$R(Simon)$	4, 5, $\Rightarrow$-E |
7.	$R(Simon) \wedge \neg R(Simon)$	2, 6, $\wedge$-I |
8.	$\neg\neg H(Simon)$	5, 7, $\neg$-I
9.	$H(Simon) \Leftrightarrow \neg\neg H(Simon)$	PL axiom
10.	$(H(Simon) \Rightarrow \neg\neg H(Simon))$ $\wedge(\neg\neg H(Simon) \Rightarrow H(Simon))$	9, defn $\Leftrightarrow$
11.	$\neg\neg H(Simon) \Rightarrow H(Simon)$	10,$\wedge$-E
12.	$H(Simon)$	8, 11, $\Rightarrow$-E