First Order Predicate Logic

Predicate logic

We began the course by considering *classical logic*, which allowed us to evaluate the truth values of simple statements. *Predicate logic* extends this by allowing us to consider statements containing variables.

Examples:

green (x)	x is green
happy (x)	x is happy

These have Boolean values (true or false), so can be combined with logic connectives:

$\neg$happy (x)

rich (x) $\wedge$ famous (x)

First Order Predicate Logic

Can also define predicates in terms of other predicates:

$$\text{successful}(x) \equiv \text{rich}(x) \vee \text{famous}(x)$$
$$\text{girl}(x) \equiv \text{child}(x) \wedge \text{female}(x)$$

Predicates can have more than one variable -- e.g. binary predicates:

father (x,y)	x is the father of y
loves (x,y)	x loves y

More generally, *n*-ary predicates:

better-player (x,y,z)
x is a better player than y at z

Mostly, only use binary or unary predicates.

Evaluating predicates

To evaluate the truth value of a predicate, must assign values to (*instantiate*) all of its variables:

loves (x,y) unknown

Can't evaluate because it contains *free variables*. However:

loves (Helen, Malcolm) true
loves (William H, Tony B) false

Can also partly assign variables:
loves (x, Julie)

This can be regarded as a predicate itself:
loves_Julie (x) ≡ loves (x, Julie)

that is, "x loves Julie."

First Order Predicate Logic

In fact, statements of classical logic can be considered as predicates with fully assigned variables:

The sky is blue. blue (sky)

Puff is a green dragon.

Giles loves Anna.

Either Alastair or Kath is right.

If Puff is a dragon,
then he can fly.

[Can even regard the atoms *true* and *false* as predicates:

true ()

false ()

containing no variables at all.]

Predicates as functions

Formally, we can define a predicate as a particular kind of function.

By analogy:

5	true
2+3	true $\lor$ false
x–y	$p \Rightarrow q$
f (x) + g (x)	green (x) $\land$ dragon (x)

Can regard predicate as a function from a set S to the set {true, false}.

The values for x are drawn from S -- as with relations, usually must specify the set of interest when defining the predicate.

In this view, a predicate is a kind of *test*, or condition on the members of S.

Predicates as relations

Another way of looking at binary predicates is as relations:

$$P(x,y) \quad \underline{\text{iff}} \quad (x,y) \in R$$

For example:

[predicates]

father (Philip, Charles)
father (Charles, William)
father (Charles, Harry)

[relation]

R = {(Philip, Charles), (Charles, William), (Charles, Harry)}

Quantifiers

So far, we can only make statements about concrete subjects. Would like to talk about things like *all*, *some*, *none*, *any*...
Need quantifiers.

The *universal* quantifier:

$\forall$ for all

expresses a statement about all members of the set.

Examples:

All men are mortal. $\forall x \in \{men\}: mortal(x)$

Not everyone is lucky. $\neg\forall x: lucky(x)$

Must be careful about negation!

No one is perfect.

First Order Predicate Logic

The *existential* quantifier:

$\exists$ there exists
(… such that)

expresses a statement about at least one member of the set.

Examples:

Someone is a winner. $\exists$x: winner (x)
… and doesn't know it.

With negation:

Some people are unlucky.
There does not exist anyone
who is perfect.

Quantifiers are needed to properly evaluate statements like "green (x)", which contain free variables.

First Order Predicate Logic

Duality

We see that $\forall$ and $\exists$ can be converted into each other using $\neg$.

$\neg\forall x: P(x)$ Not all x are P.	$\neg\exists x: P(x)$ No x is P.
$\exists x: \neg P(x)$ Some x is not P.	$\forall x: \neg P(x)$ All x are not P.

Which of these sets of statements is stronger?

First Order Predicate Logic

Quantifiers can be formally defined just in terms of the $\wedge$ and $\vee$ operators that we already know.

The universal quantification

$$\forall x \in \{s_1, s_2, \ldots, s_n\}: P(x)$$

is equivalent to:

$$P(s_1) \wedge P(s_2) \wedge \ldots \wedge P(s_n)$$

while the existential quantification:

$$\exists x \in \{s_1, s_2, \ldots, s_n\}: P(x)$$

is equivalent to:

$$P(s_1) \vee P(s_2) \vee \ldots \vee P(s_n)$$

So the distributive law of $\neg$ over $\forall$ and $\exists$ follows directly from its application to $\wedge$ and $\vee$.

Quantifiers and implication

A particularly important class of statements involve quantifiers and the implication operator.

For example:

$\forall x$: rises (x) $\Rightarrow$ converges (x)

[due to Flannery O'Connor]

Can also rewrite the Greek syllogism:

$\forall x$: man (x) $\Rightarrow$ mortal (x)

In general, a statement of the form:

$\forall x \in S$: P(x)

can be rewritten:

$\forall x$: in_S (x) $\Rightarrow$ P (x)

by introducing a new predicate for set membership, in_S.

First Order Predicate Logic

More examples:

All green dragons can fly.
The child of a dragon is always a dragon.
Some red dragons can't fly.

$\neg\exists x$: man (x) $\wedge$ island (x)
$\forall x$: good (x) $\Rightarrow$ ends (x)
$\exists x$: watching (x, me)

What are the negations of these?

What about:

All dragons are friends with each other.
Every dragon has a red child.
All dragons with children are happy. (tricky)

Need multiple quantifiers.

Multiple quantifiers

Since quantified statements are themselves statements (albeit with fewer free variables), we can nest quantifiers.

$$\forall x: \forall y: \text{friend}(x,y)$$

IMPORTANT: Order matters.

$$\forall x: \exists y: \text{needs}(x,y)$$

vs: $\exists y: \forall x: \text{needs}(x,y)$

No man is good enough for a father's daughter.

Quantifiers and connectives

We can even take quantified statements and combine them with our usual logical connectives:

$[\exists x\colon \text{win}(x)] \wedge [\exists x\colon \text{lose}(x)]$

$[\forall x\colon \text{ready}(x)] \Rightarrow \text{can\_fly}(\text{rocket})$

Be careful; consider:

$\forall x\colon \text{male}(x) \vee \text{female}(x)$

vs: $[\forall x\colon \text{male}(x)] \vee [\forall x\colon \text{female}(x)]$

Can even have:

$\exists y\colon \text{child}(y,x)$

$\forall x\colon [\exists y\colon \text{child}(y,x)] \Rightarrow \text{happy}(x)$