Predicate Logic - Introduction

Outline

- Motivation
- Predicates & Functions
- Quantifiers $\forall, \exists$
- Coming to Terms with Formulas
- Quantifier Scope & Bound Variables
- Free Variables & Sentences

Motivation:

Specification of programs

Make requirements unambiguous.
E.g. For table meant to define a function $f(x)$:

$$f(x) = \begin{array}{|c|c|c|} \hline C_1(x) & C_2(x) & C_3(x) \\ \hline f_1(x) & f_2(x) & f_3(x) \\ \hline \end{array} \begin{array}{l} \leadsto \text{condition} \\ \leadsto \text{action} \end{array}$$

i) The table is consistent (i.e. not contradictory) - a sufficient condition is that there is no "overlap" between column conditions:

$$\forall x \neg((C_1(x) \wedge C_2(x)) \vee (C_1(x) \wedge C_3(x)) \vee (C_2(x) \wedge C_3(x)))$$

ii) Table is complete - for all possible inputs, an output is specified

$$\forall x (C_1(x) \vee C_2(x) \vee C_3(x))$$

Motivation:

Verification of programs:

E.g. How do you know you got 2A04 lab 2 right? When every input to program gives same answer as table.

For all a, b, c $prog(a, b, c) = table(a, b, c)$

$$\forall x_a \forall x_b \forall x_c (prog(x_a, x_b, x_c) = table(x_a, x_b, x_c))$$

E.g. How do you show someone got 2A04 Lab 2 wrong? Show that there is at least one case when program gives wrong (different from table) answer.

$$\exists x_a \exists x_b \exists x_c (prog(x_a, x_b, x_c) \neq table(x_a, x_b, x_c))$$

or equivalently

$$\neg \forall x_a \forall x_b \forall x_c (prog(x_a, x_b, x_c) = table(x_a, x_b, x_c))$$

Predicates & Functions

We will use the notation $(u_1, u_2, \ldots, u_n)$ for an ordered n-tuple.

Def: Let A be a set. An n *place predicate* or *relation* (over A) is a subset of A^n.

An n place predicate P is said to have an *arity* of n and is also called an n*-ary predicate*.

n-ary predicate P can also be considered to define a *characteristic function*:

$$P : A^n \to \{T, F\}$$

$$P(u_1, u_2, \ldots u_n) := \begin{cases} T, & \text{if } (u_1, u_2, \ldots, u_n) \in P \\ F, & \text{if } (u_1, u_2, \ldots, u_n) \notin P \end{cases}$$

E.g. If $A := \mathbb{R}$ then $\leq := \{(x, y) | x \leq y\} \subset \mathbb{R}^2$ and $\leq (1, 2) = T$ while $\leq (2, 1) = F$

Many mathematical predicates such as $\leq$ are written using *infix notation* as $1 \leq 2$.

Predicates and Functions

Subset and characteristic function representations are often used interchangeably. Typically a given predicate symbol P has fixed arity (number of arguments) n. To make this explicit formally the notation P^n is often used.

We will assume that the arity of a predicate is obvious from how it is used or the *context*. E.g. $P(x, y)$ is a binary predicate while $Q(u, v, x, y)$ is a 4-ary predicate.

Some logics (PVS) allow *overloading* of predicate symbols:

$P(x, y)$ might denote $x \leq y$ while
$P(x, y, z)$ might denote $x + y = z$.

The intended *interpretation* is clear from the context.

Predicates and Functions

Def: f is *a function of* n *variables* or an n*-ary function* if f is a subset of A^{n+1} (f is $(n + 1) - ary$ relation over A) such that if $(u_1, \ldots, u_n, v_1) \in f$ and $(u_1, \ldots, u_n, v_2) \in f$ then $v_1 = v_2$. We denote this $f : A^n \to A$.

Formally:

$$\forall u_1 \ldots \forall u_n \forall v_1 \forall v_2$$
$$(f(u_1, \ldots, u_n) = v_1 \wedge f(u_1, \ldots, u_n) = v_2 \to v_1 = v_2)$$

PVS similarly allows one to overload function symbols:

```
x,y,z:VAR nat
f(x,y):nat = x + y
f(x,y,z):nat = x * y * z
```


Quantifiers

$\forall$ (FORALL) - Universal Quantifier

$\forall x P(x)$ - "For all x, $P(x)$ holds (is true).
Also read as "For every $x \ldots$" "For each $x \ldots$"

$\exists$ (EXISTS) - Existential Quantifier

$\exists x P(x)$ - "There exists an x such that $P(x)$ holds."

Also read as "There is at least one $x \ldots$"
"There is an x satisfying P."

Note: Order counts when you mix quantifiers!

"In every class there is a student with the highest mark."

$$\forall x \exists y (C(x) \wedge S(y) \to H(x, y))$$

"There is a student such that in every class she has the highest mark."

$$\exists y \forall x (C(x) \wedge S(y) \to H(x, y))$$

Consider the following statement:
No student who likes math also likes Oscar.
This could be interpreted as:

> *For every x, if x is a student and x likes math, then x doesn't like Oscar.*

$$\forall x(S(x) \wedge M(x) \rightarrow \neg L(x,o))$$

A seemingly equivalent statement would be:

> *For every x, if x is a student then it is not the case that x likes math and likes Oscar.*

$$\forall x(S(x) \rightarrow \neg(M(x) \wedge L(x,o)))$$

Are these statements really saying the same thing?

Restriction of Quantifiers

Often want to restrict ourselves to considering x's of certain *type*.

$$\forall x(P(x) \rightarrow Q(x))$$

$$\exists x(P(x) \wedge Q(x))$$

E.g. In Dilbert $\forall x(Manager(x) \rightarrow Idiot(x))$
$\exists x(Animal(x) \wedge \neg Glasses(x))$

What is the relationship between these two forms?

$$\neg\forall x(P(x) \rightarrow Q(x)) \text{ iff } \exists x(P(x) \wedge \neg Q(x))$$

Why?

Note: Other styles of quantification

$$(\forall x \in P)Q(x) \text{ or } \forall x \in P : Q(x)$$

mean same as $\forall x(P(x) \rightarrow Q(x))$

$\exists x(P(x) \wedge Q(x))$ is also written:

$$(\exists x \in P)Q(x) \text{ or } \exists x \in P : Q(x)$$

read "There exists an x in P such that Q(x) holds."

This starts to lead into Type Theory.

Language of Predicate Calculus

A *predicate vocabulary* consists of three sets $(\mathcal{C}, \mathcal{F}, \mathcal{P})$ where each denotes respectively:

$\mathcal{C}$ - set of constant symbols

$\mathcal{F}$ - set of functions symbols

$\mathcal{P}$ - set of predicate symbols

We also have an arity associated with each function and predicate symbol which we can think of as a mapping:

$$arity : \mathcal{F} \cup \mathcal{P} \rightarrow \mathbb{N}$$

where $\mathbb{N}$ denotes natural numbers $\{0, 1, 2, \ldots\}$.

For our Oscar example: $\mathcal{C} = \{o\}, \mathcal{F} = \emptyset$,
$\mathcal{P} = \{L, M, S\}$ and $arity(L) = 2$.

Language of Predicate Calculus (cont)

In addition to constants, function symbols and predicate symbols our language will make use of

Variables: e.g., u, v, w, x, y, z or u_1, x_4, etc.

Connectives: $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$

Quantifiers: $\forall, \exists$

as well as parentheses (,) and we'll also usually include the two special 0-ary predicate symbols $\top, \bot$.

Note: In PVS most strings of letters, numbers and underscore can be defined to be a variable, constant, function symbol or predicate. In fact a string can even be several of these things at once!
PVS translates $\forall$ as FORALL and $\exists$ as EXISTS

Language of Predicate Calculus (cont)

There are now two *sorts* of objects we are dealing with:

Terms: Variables such as x, constants such as o and functions applied to these such as $f(x, o)$. All denote objects of our universe.

Formulas: Predicates $P(x)$ and logical connectives such as $M(x) \wedge L(x, o)$ and quantifiers over a variable applied to a formula such as $\forall x P(x)$. Once values are substituted for constants and free variables, these formulas all denote truth values.

We now formally define terms and formulas.

Terms

Def: A *term* is defined as follows:

1. Any constant $c \in \mathcal{C}$ or variable is a term.
2. If $t_1, \ldots, t_n$ are terms and $f \in \mathcal{F}$ is an n-ary function symbol (i.e. $arity(f) = n$) then $f(t_1, \ldots, t_n)$ is a term.

In BNF form a term t is defined as:

$$t ::= x | c | f(t, \ldots, t)$$

where x is a variable, $c \in \mathcal{C}$ and $f \in \mathcal{F}$ has $arity(f) = n$.

Constants can be thought of as 0-ary functions - they take no arguments so we drop the $(\cdot)$ and eliminate the set $\mathcal{C}$. (e.g., for the Oscar example then $\mathcal{F} = \{o\}$ and $arity(o) = 0$).

Formulas

Def: The *set of formulas over* $(\mathcal{F}, \mathcal{P})$ is defined as follows:

1. If $t_1, \ldots, t_n$ are terms and $P \in \mathcal{P}$ is an n-ary predicate symbol, then $P(t_1, \ldots, t_n)$ is a formula.
2. If ϕ and ψ are formulas, so are:
   $$(\neg\phi), (\phi \wedge \psi), (\phi \vee \psi), (\phi \rightarrow \psi), (\phi \leftrightarrow \psi)$$
   $\top$ and $\bot$ are also formulas.
3. If x is a variable and ϕ is a formula, then so are $(\forall x \phi)$ and $(\exists x \phi)$.

In BNF form formulas are defined as:

$$\begin{aligned} \phi \ ::= \ & P(t_1 \ldots t_n) | (\neg\phi) | (\phi \wedge \phi) | (\phi \vee \phi) | \\ & (\phi \rightarrow \phi) | (\phi \leftrightarrow \phi) | (\forall x \phi) | (\exists x \phi) \end{aligned}$$

where x is a variable, t_i are terms (over $\mathcal{F}$), and $P \in \mathcal{P}$ has $arity(P) = n$.

Order of Precedence & Parenthesis

Recall: We use precedence of logical operators and associativity of $\wedge, \vee, \leftrightarrow$ to drop parentheses. It is understood that this is shorthand for the fully parenthesized expressions.

Huth+Ryan uses order of precedence:

$$\begin{matrix} \neg \\ \forall \\ \exists \end{matrix}, \begin{matrix} \wedge \\ \vee \end{matrix}, \begin{matrix} \rightarrow \\ \leftrightarrow \end{matrix}$$

PVS uses order of precedence:

$$\neg, \wedge, \vee, \rightarrow, \leftrightarrow, \begin{matrix} \forall \\ \exists \end{matrix}$$

$(\forall x)P(x) \rightarrow (\exists y)Q(x, y) \wedge P(y)$ becomes:
In Huth+Ryan:

$$(\forall x P(x)) \rightarrow ((\exists y Q(x, y)) \wedge P(y))$$

In PVS:

$$\forall x (P(x) \rightarrow (\exists y (Q(x, y) \wedge P(y))))$$

Parse Tree

We can apply this inductive definition in reverse to construct a formula's *parse tree*. A parse tree represents a WFF ϕ if

i) The root node is P and if $arity(P) = n$ then there are n well formed term subtrees,

ii) the root is $\forall x$ or $\exists x$ and there is only one well formed subtree

iii) the root is $\neg$ and there is only one well formed subtree, or

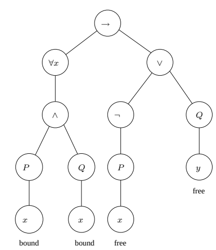
iv) the root is $\wedge, \vee, \rightarrow$ or $\leftrightarrow$ and there are two well formed subtrees or

Note: All leaf nodes will be variables or constants (or $\perp$ or $\top$).

Parse Tree (cont)

Example 1: Draw the parse tree for the formula

$$\forall x(P(x) \wedge Q(x)) \rightarrow \neg P(x) \vee Q(y)$$



Example 2: Draw the parse trees for the two formulas on slide 14.

Quantifier Scope & Bound Variables

Scope of quantifiers: The *scope of a quantifier* in a formula ϕ is the subformula to which the quantifier was applied in the inductive construction of ϕ.

In the fully parenthesized formulas the scope is the quantifier itself and the matching parentheses immediately following. E.g

$$P(x,y) \rightarrow \overbrace{\forall x(Q(x) \wedge P(f(y,x),x))}^{\text{scope}} \vee \forall z(Q(f(x,z))$$

$$(P(x,y) \rightarrow (\underbrace{(\forall x(Q(x) \wedge P(f(y,x),x)))}_{\text{scope}} \vee (\forall z(Q(f(x,z)))$$

An occurrence of a variable x in a formula ϕ is *bound* if it falls within the scope of $\forall x$ or $\exists x$.

Alternatively we can consider the parse tree. Then an occurrence of x is bound if it occurs under a $\forall x$ or $\exists x$, otherwise it is free.

Free Variables & Sentences

Def: The *free variables* of a formula ϕ, denoted $FV(\phi)$ can be defined inductively as follows:

1. For constants (e.g. k): $FV(k) = \emptyset$
2. For variables: $FV(x) = \{x\}$
3. For terms:

   $$FV(f(t_1, \ldots, t_n)) = FV(t_1) \cup \ldots \cup FV(t_n)$$

4. For atomic formulas:

   $$FV(P(t_1, \ldots, t_n)) = FV(t_1) \cup \ldots \cup FV(t_n)$$

5. For formulas ϕ, ψ:

   $$FV(\neg\phi) = FV(\phi)$$
   $$FV(\phi \wedge \psi) = FV(\phi) \cup FV(\psi)$$
   $$FV(\forall x\phi) = FV(\phi) - \{x\}$$
   $$FV(\exists x\phi) = FV(\phi) - \{x\}$$

   Also, $FV(\top) = FV(\perp) = \emptyset$

Def: A predicate logic formula ϕ is a *sentence* if $FV(\phi) = \emptyset$, otherwise ϕ is a *sentence form*.

Valid Substitutions

Def: For formula ϕ, term t and x is a variable, replace each free occurrence of x with t to obtain $\phi[t/x]$, the *substitution* of t for x. It is a *valid substitution* provided no occurrence of a (free) variable in t is bound in $\phi[t/x]$.

Substitution is valid if:

1. Each free occurrence of x in ϕ is replaced by t.
2. For each $y \in FV(t)$, every occurrence y in a substituted t is free in $\phi[t/x]$.

Example: Let ϕ be $Ix \to \exists y(Iy \wedge y > x)$

$\phi[u/x]$ Valid: $Iu \to \exists y(Iy \wedge y > u)$

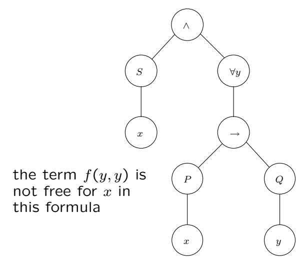
$\phi[y/x]$ Invalid: $Iy \to \exists y(Iy \wedge y > y)$

Parse Tree (cont)

Example: Consider the formula

$$S(x) \wedge \forall y(P(x) \to Q(y))$$

Q: Is $\phi[f(y,y)/x]$ valid?



A: No. y's in $f(y,y)$ become bound by $\forall y$ in when substituting for 2nd occurrence of x.