Syntax of Predicate Logic

The Language of Predicate Logic

- Domain: a non-empty set of objects
- Constants: concrete objects in the domain
- Variables: placeholders for concrete objects in the domain
- Functions: takes objects in the domain as arguments and returns an object of the domain.
- Predicates: takes objects in the domain as arguments and returns true or false. They describe properties of objects or relationships between objects.
- Quantifiers: for how many objects in the domain is the statement true?

A question on functions

Consider two translations of the sentence "every child is younger than its mother."

1. $(\forall x\,(\forall y\,((Child(x) \wedge Mother(y,x)) \rightarrow Younger(x,y))))$
2. $(\forall x\,(Child(x) \rightarrow Younger(x, mother(x))))$

Which of the following is the best answer?

a. Both are wrong.

b. 1 is correct and 2 is wrong.

c. 2 is correct and 1 is wrong.

d. Both are correct. 1 is better.

e. Both are correct. 2 is better.

The domain is the set of people. $Child(x)$ means x is a child. $Mother(x,y)$ means x is y's mother. $Younger(x,y)$ means x is younger than y. $mother(x)$ returns x's mother.

Functions

Using functions allows us to avoid ugly/inelegant predicate logic formulas.

Try translating the following sentence with and without functions.

“Andy and Paul have the same maternal grandmother.”

The Language of Predicate Logic

The seven kinds of symbols:

1.	Constant symbols.	Usually $c, d, c_1, c_2, \dots, d_1, d_2 \dots$
2.	Variables.	Usually $x, y, z, \dots x_1, x_2, \dots, y_1, y_2 \dots$
3.	Function symbols.	Usually $f, g, h, \dots f_1, f_2, \dots, g_1, g_2$, ...
4.	Predicate symbols.	P, Q, ...P_1, P_2, ..., Q_1, Q_2, ...
5.	Connectives:	$\neg$, $\wedge$, $\vee$, $\rightarrow$, and $\leftrightarrow$
6.	Quantifiers:	$\forall$ and $\exists$
7.	Punctuation:	'(', ')', and ','

Function symbols and predicate symbols have an assigned *arity*—the number of arguments required. For example,

- $f^{(1)}$: f is a unary function.
- $P^{(2)}$: P is a binary predicate.

Two kinds of expressions

In predicate logic, we need to consider two kinds of expressions:

- those that refer to an object of the domain, called *terms*, and
- those that can have a truth value, called *formulas*.

Terms

Each term refers to an object of the domain.

We define the set of terms inductively as follows.

1. Each constant symbol is an atomic term.
2. Each variable is an atomic term.
3. $f(t_1, \dots, t_n)$ is a term if $t_1, \dots, t_n$ are terms and f is an n-ary function symbol. (If f is a binary function symbol, then we may write $(t_1 \ f \ t_2)$ instead of $f(t_1, t_2)$.)
4. Nothing else is a term.

Which expressions are terms?

A term refers to an object of the domain.

Which of the following expressions is a term?

a. $g(d, d)$
b. $P(f(x, y), d)$
c. $f(x, g(y, z), d)$
d. $g(x, f(y, z), d)$

Let d be a constant symbol. Let P be a predicate symbol with 2 arguments. Let f be a function symbol with 2 arguments and g be a function symbol with 3 arguments. Let x, y, and z be variable symbols.

Is this a term?

True or False: The expression $(2 - f(x)) + (y * x)$ is a term.

a. True

b. False

c. Not enough information to tell

The domain is the set of integers. $+$, $-$ and $*$ are binary functions. f is a unary function. x and y are variables and 2 is a constant.

Well-Formed Predicate Logic Formulas

We define the set of well-formed formulas of predicate logic inductively as follows.

1. $P(t_1, \dots, t_n)$ is an atomic formula if P is an n-ary predicate symbol and each t_i is a term $(1 \leq i \leq n)$.
2. $(\neg\alpha)$ is a formula if α is a formula.
3. $(\alpha \star \beta)$ is a formula if α and β are formulas and $\star$ is a binary connective symbol.
4. Each of $(\forall x\ \alpha)$ and $(\exists x\ \alpha)$ is a formula if α is a formula and x is a variable.
5. Nothing else is a formula.

Determine whether a formula is well-formed

m is a constant and x and y are variables. $P^{(2)}$ and $Q^{(2)}$ are binary predicates. $f^{(1)}$ is a unary function.

Which of the following is a well-formed predicate logic formula?

a. $(f(x) \to P(x, y))$
b. $\forall y\ P(m, f(y))$
c. $P(x, y) \to Q(Q(x))$
d. $Q(m, f(m))$
e. $P(m, f(Q(x, y)))$

Determine whether a formula is well-formed

Things to consider:

- Are there *enough brackets*? Are the brackets *in the right places*?
- Is each unary/binary *connective* applied to the right *number of predicates*?
- Does every function have the right *number* and *type* of arguments?
- Does every predicate have the right *number* and *type* of arguments?

Comparing the Definitions of WFFs

Let's compare and contrast the definitions of WFFs for propositional and predicate logic.

- Which parts of the two definitions are *the same*?
- The definition of WFF for propositional logic says that *a propositional variable is an atomic WFF.* Is this still the case for predicate logic?
- What is *new in the definition of WFF for predicate logic* compared to the definition of WFF for propositional logic?
- If we were to prove that *every WFF for predicate logic has a property P* by *structural induction*, what does the proof look like? What are the *base case(s)* and what are the *cases in the induction step*?

Parse Trees of Predicate Logic Formulas

New elements in the parse tree:

- Quantifiers $\forall x$ and $\exists y$ has one subtree, similar to the unary connective negation.
- A predicate $P(t_1, t_2, \ldots, t_n)$ has a node labelled P with a sub-tree for each of the terms $t_1, t_2, \ldots, t_n$.
- A function $f(t_1, t_2, \ldots, t_n)$ has a node labelled f with a sub-tree for each of the terms $t_1, t_2, \ldots, t_n$.

Example 1: $(\forall x\,(P(x) \wedge Q(x))) \to (\neg(P(f(x,y)) \vee Q(y)))$

Example 2: $(\forall x\,((P(x) \wedge Q(x)) \to (\neg(P(f(x,y)) \vee Q(y)))))$

Evaluating a Formula

To evaluate the truth value of a formula, we need to replace the variables by concrete objects in the domain. However, we don't necessarily have to perform this substitution for every variable.

There are two types of variables in a formula:

- A variable may be *free*. To evaluate the formula, we need to replace a free variable by an object in the domain.
- A variable may be *bound by a quantifier*. The quantifier tells us how to evaluate the formula.

We need to understand *how to determine whether a variable is free/bound* and *how to replace a free variable with an object in the domain*.

Free and Bound Variables

In a formula $(\forall x\ \alpha)$ or $(\exists x\ \alpha)$, the *scope* of a quantifier is the formula α. A quantifier *binds* its variable within its scope.

An occurrence of a variable in a formula is *bound* if it lies in the scope of some quantifier of the same variable. Otherwise the occurrence of this variable is *free*.

- If a variable occurs multiple times, we need to consider each occurrence of the variable separately.
- The variable symbol immediately after $\exists$ or $\forall$ is neither free nor bound.

A formula with no free variables is called a *closed formula* or *sentence*.

Determine whether a variable is free or bound

Determine whether a variable is free or bound using a parse tree.

1. Draw the parse tree for the formula.
2. Choose the leaf node for an occurrence of a variable.
3. Walk up the tree. Stop as soon as we encounter a quantifier for this variable or we reach the root of the tree.
4. If we *encountered a quantifier for the variable*, this occurrence of the variable is *bound*.
5. If we *reached the root of the tree which is not a quantifier for the variable*, this occurrence of the variable is *free*.

Example 1: $(\forall x\,(P(x) \wedge Q(x))) \rightarrow (\neg(P(f(x,y)) \vee Q(y)))$

Example 2: $(\forall x\,((P(x) \wedge Q(x)) \rightarrow (\neg(P(f(x,y)) \vee Q(y)))))$

Substitution

When writing natural deduction proofs in predicate logic, it is often useful to replace a variable in a formula with a term.

Suppose that the following sentences are true:

$$(\forall x\, (\mathit{Fish}(x) \rightarrow \mathit{Swim}(x)))\, (1)$$

$$\mathit{Fish}(\mathit{Nemo})\, (2)$$

To conclude that Nemo can swim, we need to replace every occurence of the variable x in the implication $\mathit{Fish}(x) \rightarrow \mathit{Swim}(x)$ by the term *Nemo*. This gives us

$$(\mathit{Fish}(\mathit{Nemo}) \rightarrow \mathit{Swim}(\mathit{Nemo}))\, (3)$$

By modus ponens on (2) and (3), we conclude that $\mathit{Swim}(\mathit{Nemo})$.

Formally, we use *substitution* to refer to the process of replacing x by *Nemo* in the formula $\forall x\ \mathit{Fish}(x) \rightarrow \mathit{Swim}(x)$.

Substitution

Intuitively, $\alpha[t/x]$ answers the question,

"What happens to α if x has the value specified by term t?"

For a variable x, a term t, and a formula α, $\alpha[t/x]$ denotes the resulting formula by replacing each *free* occurrence of x in α with t. In other words, substitution *does NOT* affect *bound* occurrences of the variable.

Examles of Substitution

Examples.

- If α is the formula $E(f(x))$, then $\alpha[(y+y)/x]$ is $E(f(y+y))$.
- $\alpha[f(x)/x]$ is $E\big(f(f(x))\big)$.
- $E(f(x+y))[y/x]$ is $E(f(y+y))$.

- If β is $\big(\forall x\,(E(f(x)) \wedge S(x,y))\big)$, then $\beta[g(x,y)/x]$ is β, because β has no free occurrence of x.

Examples: Substitution

Example. Let β be $\big(P(x) \wedge (\exists x\ Q(x))\big)$. What is $\beta[y/x]$?

$\beta[y/x]$ is $\big(P(y) \wedge (\exists x\ Q(x))\big)$. Only the free x gets substituted.

Example. What about $\beta[(y-1)/z]$, where β is $\Big(\forall x \left(\exists y \left((x+y) = z\right)\right)\Big)$?

At first thought, we might say $\Big(\forall x \left(\exists y \left((x+y) = (y-1)\right)\right)\Big)$. But there's a problem—the free variable y in the term $(y-1)$ got "captured" by the quantifier $\exists y$.

We want to avoid this capture.

Preventing Capture

Example. Formula $\alpha = S(x) \wedge \forall y\,(P(x) \to Q(y))$; term $t = f(y, y)$.

The leftmost x can be substituted by t since it is not in the scope of any quantifier, but substituting in $P(x)$ puts the variable y into the scope of $\forall y$.

We can prevent capture of variables by a different choice of variable. (Above, we might be able to substitute $f(z, z)$ instead of $f(y, y)$. Or alter α to quantify some other variable.)

Substitution—Formal Definition

Let x be a variable and t a term.

For a term u, the term $u[t/x]$ is u with each occurrence of the variable x replaced by the term t.

For a formula α,

1. If α is $P(t_1, \dots, t_k)$, then $\alpha[t/x]$ is $P\Big(t_1[t/x], \dots, t_k[t/x]\Big)$.
2. If α is $(\neg\beta)$, then $\alpha[t/x]$ is $(\neg\beta[t/x])$.
3. If α is $(\beta \star \gamma)$, then $\alpha[t/x]$ is $(\beta[t/x] \star \gamma[t/x])$.
4. …

(Continued next page...)

Substitution—Formal Definition (2)

For variable x, term t and formula α:

$\vdots$

4. If α is $(Qx\ \beta)$, then $\alpha[t/x]$ is α.
5. If α is $(Qy\ \beta)$ for some other variable y, then
   (a) If y does not occur in t, then $\alpha[t/x]$ is $(Qy\ \beta[t/x])$.
   (b) Otherwise, select a variable z that occurs in neither α nor t; then $\alpha[t/x]$ is $(Qz\ (\beta[z/y])[t/x])$.

The last case prevents capture by renaming the quantified variable to something harmless.

Example, Revisited

Example. If α is $\big(\forall x\,(\exists y\,(x+y=z))\big)$, what is $\alpha[(y-1)/z]$?

This falls under case 5(b): the term to be substituted, namely $y-1$, contains a variable y quantified in formula α.

Let β be $(x+y=z)$; thus α is $\big(\forall x\,(\exists y\ \beta)\big)$.

Select a new variable, say w. Then

$$\beta[w/y] \quad \text{is} \quad x+w=z,$$

and

$$\beta[w/y][(y-1)/z] \quad \text{is} \quad (x+w)=(y-1)\ \ .$$

Thus the required formula $\alpha[(y-1)/z]$ is

$$\big(\forall x\ \exists w((x+w)=(y-1))\big)\ \ .$$