# Syntax of Predicate Calculus

The predicate calculus uses the following types of symbols:

**Constants:** A constant symbol denotes a particular entity. E.g. `John`, `Muriel`, `1`.

**Functions:** A function symbol denotes a mapping from a number of entities to a single entities: E.g. `FatherOf` is a function with one argument. `Plus` is a function with two arguments. `FatherOf(John)` is some person. `Plus(2,7)` is some number.

**Predicates:** A predicate denotes a relation on a number of entities. e.g. `Married` is a predicate with two arguments. `Odd` is a predicate with one argument. `Married(John, Sue)` is a sentence that is true if the relation of marriage holds between the people John and Sue. `Odd(Plus(2,7))` is a true sentence.

**Variables:** These represent some undetermined entity. Examples: `x`, `s1`, etc.

**Boolean operators:** $\neg$, $\vee$, $\wedge$, $\Rightarrow$, $\Leftrightarrow$.

**Quantifiers:** The symbols $\forall$ (for all) and $\exists$ (there exists).

**Grouping symbols:** The open and close parentheses and the comma.

A *term* is either

1. A constant symbol; or
2. A variable symbol; or
3. A function symbol applied to terms.

Examples: `John`, `x`, `FatherOf(John)`, `Plus(x,Plus(1,3))`.

An *atomic formula* is a predicate symbol applied to terms.
Examples: `Odd(x)`. `Odd(plus(2,2))`. `Married(Sue,FatherOf(John))`.

A *formula* is either

1. An atomic formula; or
2. The application of a Boolean operator to formulas; or
3. A quantifier followed by a variable followed by a formula.

Examples: `Odd(x).` `Odd(x)` $\vee$ $\neg$`Odd(Plus(x,x)).` $\exists_{\texttt{x}}$ `Odd(Plus(x,y)).`
$\forall_{\texttt{x}}$ `Odd(x)` $\Rightarrow$ $\neg$`Odd(Plus(x,3)).`

A *sentence* is a formula with no free variables. (That is, every occurrence of every variable is associated with some quantifier.)

# Clausal Form

A *literal* is either an atomic formula or the negation of an atomic formula.

Examples: `Odd(3).` $\neg$`Odd(Plus(x,3)).` `Married(Sue,y).`

A *clause* is the disjunction of literals. Variables in a clause are interpreted as universally quantified with the largest possible scope.

Example: `Odd(x)` $\vee$ `Odd(y)` $\vee$ $\neg$`Odd(Plus(x,y))` is interpreted as
$\forall_{\texttt{x,y}}$ `Odd(x)` $\vee$ `Odd(y)` $\vee$ $\neg$`Odd(Plus(X,Y)).`

## Converting a sentence to clausal form

1. Replace every occurrence of $\alpha \Leftrightarrow \beta$ by
   $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$. When this is complete, the sentence will have no occurrence of $\Leftrightarrow$.

2. Replace every occurrence of $\alpha \Rightarrow \beta$ by $\neg\alpha \vee \beta$. When this is complete, the only Boolean operators will be $\vee$, $\neg$, and $\wedge$.

3. Replace every occurrence of $\neg(\alpha \vee \beta)$ by $\neg\alpha \wedge \neg\beta$; every occurrence of $\neg(\alpha \wedge \beta)$ by $\neg\alpha \vee \neg\beta$; and every occurrence of $\neg\neg\alpha$ by $\alpha$.

   New step: Also, replace every occurrence of $\neg\exists_\mu \alpha$ by $\forall_\mu \neg\alpha$ and every occurrence of $\neg\forall_\mu \alpha$ by $\exists_\mu \neg\alpha$.

   Repeat as long as applicable. When this is done, all negations will be next to an atomic sentence.

4. (New Step: Skolemization). For every existential quantifier $\exists_\mu$ in the formula, do the following: If the existential quantifier is not inside the scope of any universal quantifiers, then

   i. Create a new constant symbol $\gamma$.

   ii. Replace every occurrence of the variable $\mu$ by $\gamma$.

   iii. Drop the existential quantifier.

   If the existential quantifier is inside the scope of universal quantifiers with variables $\Delta_1 \ldots \Delta_k$,then

   i. Create a new function symbol $\gamma$.

   ii. Replace every occurrence of the variable $\mu$ by the term $\gamma(\Delta_1 \ldots \Delta_k)$

   iii. Drop the existential quantifier.

   Example. Change $\exists_{\texttt{x}}$ `Blue(x)` to `Blue(Sk1)`.
   Change $\forall_{\texttt{x}}\exists_{\texttt{y}}$ `Odd(Plus(x,y))` to $\forall_{\texttt{x}}$ `Odd(Plus(x,Sk2(x))`.
   Change $\forall_{\texttt{x,y}}\exists_{\texttt{z}}\forall_{\texttt{a}}\exists_{\texttt{b}}$ `P(x,y,z,a,b)` to `P(x,y,Sk3(x,y),a,Sk4(x,y,a))`.

5. New step: Elimination of universal quantifiers:
   Part 1. Make sure that each universal quantifier in the formula uses a variable with a different name, by changing variable names if necessary.
   Part 2. Drop all univeral quantifiers.

   Example. Change `[`$\forall_{\texttt{x}}$ `P(x)]` $\vee$ `[`$\forall_{\texttt{x}}$ `Q(x)]` to `P(x)` $\vee$ `Q(x1)`.

6. (Same as step 4 of CNF conversion.) Replace every occurrence of $(\alpha \wedge \beta) \vee \gamma$
   by $(\alpha \vee \gamma) \wedge (\beta \vee \gamma)$, and every occurrence of $\alpha \vee (\beta \wedge \gamma)$ by $(\alpha \vee \beta) \wedge (\alpha \vee \gamma)$. Repeat as long as applicable. When this is done, all conjunctions will be at

   top level.

7. (Same as step 5 of CNF conversion.) Break up the top-level conjunctions into separate sentences. That is, replace $\alpha \wedge \beta$ by the two sentences $\alpha$ and $\beta$. When this is done, the set will be in CNF.

**Example:**

Start. $\forall_x$ [Even(x) $\Leftrightarrow$ [$\forall_y$ Even(Times(x,y))]]

After Step 1: $\forall_x$ [[Even(x) $\Rightarrow$ [$\forall_y$ Even(Times(x,y))]] $\wedge$
[[$\forall_y$ Even(Times(x,y))] $\Rightarrow$ Even(x)]].

After step 2: $\forall_x$ [[$\neg$Even(x) $\vee$ [$\forall_y$ Even(Times(x,y))]] $\wedge$
[$\neg$[$\forall_y$ Even(Times(x,y))] $\vee$ Even(x)]].

After step 3: $\forall_x$ [[$\neg$Even(x) $\vee$ [$\forall_y$ Even(Times(x,y))]] $\wedge$
[[$\exists_y$ $\neg$Even(Times(x,y))] $\vee$ Even(x)]].

After step 4: $\forall_x$ [[$\neg$Even(x) $\vee$ [$\forall_y$ Even(Times(x,y))]] $\wedge$
[$\neg$Even(Times(x,Sk1(x))) $\vee$ Even(x)]].

After step 5: [$\neg$Even(x) $\vee$ Even(Times(x,y))] $\wedge$
[$\neg$Even(Times(x,Sk1(x))) $\vee$ Even(x)].

Step 6 has no effect.

After step 7: $\neg$Even(x) $\vee$ Even(Times(x,y)).
$\neg$Even(Times(x,Sk1(x))) $\vee$ Even(x).

# Resolution

A *substitution* is an association of variables with terms;

Example: $\sigma$ = { x $\rightarrow$ A, y $\rightarrow$ F(z) } is a substitution.

The *application* of a substitution $\sigma$ to a clause $\phi$, written $\phi\sigma$, is the clause that is obtained when each occurrence in $\phi$ of a variable in $\sigma$ is replaced by the associated term.

Example: If $\phi$ is the clause P(x,y) $\vee$ $\neg$Q(y,z), and $\sigma$ is the substution above, then $\phi\sigma$ is P(A,F(z)) $\vee$ Q(F(z),z).

Fact: If $\phi$ is true, then $\phi\sigma$ is true.

Let $\alpha$ and $\beta$ be atomic formulas. $\alpha$ and $\beta$ are *unifiable* if there are substutions $\sigma_A$ and $\sigma_B$ such that $\alpha\sigma_A = \beta\sigma_B$.

Examples. P(A,B) is unifiable with P(x,y) under the substitution $\sigma_B$ = { x $\rightarrow$A, y $\rightarrow$B }

P(A,B) is not unifiable with P(x,x).

P(A,z) is unifiable with P(z,B) under the substitutions $\sigma_A$ = {z $\rightarrow$ B }, $\sigma_B$ = {z $\rightarrow$ A }.

P(F(x),w) is unifiable with P(z,z) under the substitutions $\sigma_A$ = { w $\rightarrow$ F(x) }, $\sigma_B$ = { z $\rightarrow$ F(x) }.

P(F(x),x) is not unifiable with P(z,z).

There may be more than one set of substitutions that unifies two formulas. For example P(A,F(A),x) can be unified with P(A,F(A),y) by substituting x for y, or by substituting A for both X and y, or by substituting F(A) for both x and y, or by substituting F(w) for both x and y etc. However, the *best* way to unify them is to substitute x for y (or vice versa), because all the other substitutions can be derived by further substitutions from it. It is called the *most general unifier*

(mgu).

## Resolution: Rules of Inference

1. (Factoring) Let $\phi$ be the clause $\alpha_1 \vee \alpha_2 \vee ... \vee \alpha_k$. Let $\alpha_i$ and $\alpha_j$ be two literals that are either both positive or both negative, and let $\sigma$ be a single substitution that unifies $\alpha_i$ and $\alpha_j$. Then infer $(\phi - \alpha_j)\sigma$.

Example: From `P(A,x)` $\vee$ `P(y,B)` $\vee$ `Q(x,y,C)` infer `P(A,B)` $\vee$ `Q(B,A,C)`.

2. (Resolution) Let $\phi$ be the clause $\alpha_1 \vee \alpha_2 \vee ... \vee \alpha_k$, and let $\psi$ be the clause $\beta_1 \vee \beta_2 \vee ... \vee \beta_m$. Suppose that $\alpha_i = \gamma$ and $\beta_j = \neg\delta$, where $\gamma$ and $\delta$ are atomic and where $\gamma$ unifies with $\delta$ under the substitutions $\sigma_A$ and $\sigma_B$ Then infer $(\phi - \alpha_i)\sigma_A \vee (\psi - \beta_j)\sigma_B$.

Examples: From `P(A,B)` $\vee$ `Q(B,C)` and $\neg$`P(x,y)` $\vee$ `R(x,y)` infer `Q(B,C)` $\vee$ `R(A,B)`.

From `Man(Socrates)` and $\neg$`Man(x)` $\vee$ `Mortal(x)`, infer `Mortal(Socrates)`.

From `Man(Socrates)` and $\neg$`Man(x)` infer the empty clause.

Fact: $\Delta$ is an inconsistent set of clauses if and only if there is a derivation of the empty clause from $\Delta$ using the rules of resolution and of factoring.

## Resolution: Proof Technique

To prove sentence $\phi$ from a set of axioms $\Gamma$:

Step 1. Set $\Delta = \Gamma \cup \{\neg\phi\}$;

Step 2. Convert $\Delta$ to clausal form.

Step 3. Keep applying rules 1 and 2 to derive new sentences. If you succeed in deriving the empty clause, then $\phi$ is provable from $\Gamma$. If there is no way to derive the empty clause, then $\phi$ is not provable.

**Example:**

Given: 1. $\forall_{\texttt{s1,s2}}$ `Subset(s1,s2)` $\Leftrightarrow$ [$\forall_{\texttt{x}}$ `Member(x,s1)` $\Rightarrow$ `Member(x,s2)`].
Prove: H. $\forall_{\texttt{s1,s2,s3}}$ [`Subset(s1,s2)` $\wedge$ `Subset(s2,s3)`] $\Rightarrow$ `Subset(s1,s3)`.

Negation of H: 2. $\neg$[$\forall_{\texttt{s1,s2,s3}}$ [`Subset(s1,s2)` $\wedge$ `Subset(s2,s3)`] $\Rightarrow$ `Subset(s1,s3)`].

Converted to clausal form:
1a. $\neg$`Subset(s1,s2)` $\vee$ $\neg$`Member(x,s1)` $\vee$ `Member(x,s2)`.
1b. `Member(Sk0(s1,s2),s1)` $\vee$ `Subset(s1,s2)`.
1c. $\neg$`Member(Sk0(s1,s2),s2)` $\vee$ `Subset(s1,s2)`.
2a. `Subset(Sk1,Sk2)`.
2b. `Subset(Sk2,Sk3)`.
2c. $\neg$`Subset(Sk1,Sk3)`.

| | |
|---|---|
| From 2a and 1a, infer | 3. $\neg$`Member(x,Sk1)` $\vee$ `Member(x,Sk2)`. |
| From 2b and 1a, infer | 4. $\neg$`Member(x,Sk2)` $\vee$ `Member(x,Sk3)`. |
| From 3 and 4, infer | 5. $\neg$`Member(x,Sk1)` $\vee$ `Member(x,Sk3)`. |

| | |
|---|---|
| From 2c and 1b infer | 6. `Member(Sk0(Sk1,Sk3),Sk1).` |
| From 2c and 1c infer | 7. ¬`Member(Sk0(Sk1,Sk3),Sk3).` |
| From 6 and 5 infer | 8. `Member(Sk0(Sk1,Sk3), Sk3).` |
| From 7 and 8 infer | 9. The empty clause. |