Predicate Calculus

1 Syntax

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 entity. E.g. “father_of” is a function with one argument. “plus” is a function with two arguments. “father_of(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” …

**Boolean operators:** \( \neg, \lor, \land, \Rightarrow, \Leftrightarrow \).

**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”, “father_of(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,father_of(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) \lor \neg odd(plus(X,X)),” “\( \exists X \) odd(plus(X,Y)),” “\( \forall X \) 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.)
2 Clausal Form

A literal is either an atomic formula or the negation of an atomic formula.
Examples: odd(3). ¬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) ∨ odd(Y) ∨ ¬odd(plus(X,Y)) is interpreted as ∀X,Y odd(X) ∨ odd(Y) ∨ ¬odd(plus(X,Y)).

3 Converting a sentence to clausal form

1. Replace every occurrence of α ↔ β by (α → β) ∧ (β → α).
   When this is complete, the sentence will have no occurrence of ↔.

2. Replace every occurrence of α → β by ¬α ∨ β. When this is complete, the only Boolean operators will be ∨, ¬, and ∧.

3. Replace every occurrence of ¬(α ∨ β) by ¬α ∧ ¬β; every occurrence of ¬(α ∧ β) by ¬α ∨ ¬β; and every occurrence of ¬¬α by α.
   New step: Also, replace every occurrence of ¬∃μα by ∀μ¬α and every occurrence of ¬∀μα by ∃μ¬α.
   Repeat as long as applicable. When this is done, all negations will be next to an atomic formula.

4. (New Step: Skolemization). For every existential quantifier ∃μ 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 γ.
   ii. Replace every occurrence of the variable μ by γ.
   iii. Drop the existential quantifier.

   If the existential quantifier is inside the scope of universal quantifiers with variables Δ₁...Δₖ, then
   i. Create a new function symbol γ.
   ii. Replace every occurrence of the variable μ by the term γ(Δ₁...Δₖ)
   iii. Drop the existential quantifier.
Example. Change "\(\exists X \text{ blue}(X)\)" to "\(\text{blue}(\text{sk1})\)"
Change \(\forall X \exists Y \text{ odd}(\text{plus}(X, Y))\) to \(\forall X \text{ odd}(\text{plus}(X, \text{sk2}(X)))\).
Change \(\forall X, Y \exists Z, A, B \text{ p}(X, Y, Z, A, B)\) to \(\text{p}(X, Y, \text{sk3}(X, Y), A, \text{sk4}(X, Y, A))\).

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 universal quantifiers.
   Example. Change \([\forall X \text{ p}(X)] \lor [\forall X \text{ q}(X)]\) to \(\text{p}(X) \lor \text{q}(X1)\).

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

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

3.1 Example

Start. \(\forall X [\text{even}(X) \iff [\forall Y \text{ even}(\text{times}(X, Y))]]\)

After Step 1: \(\forall X [[\text{even}(X) \implies [\forall Y \text{ even}(\text{times}(X, Y))]] \land
   [[\forall Y \text{ even}(\text{times}(X, Y))] \implies \text{even}(X)].\]

After step 2: \(\forall X [[\neg \text{even}(X) \lor [\forall Y \text{ even}(\text{times}(X, Y))]] \land
   [\neg [\forall Y \text{ even}(\text{times}(X, Y))] \lor \text{even}(X)].\]

After step 3: \(\forall X [[\neg \text{even}(X) \lor [\forall Y \text{ even}(\text{times}(X, Y))]] \land
   [[\exists Y \neg \text{even}(\text{times}(X, Y))] \lor \text{even}(X)].\]

After step 4: \(\forall X [[\neg \text{even}(X) \lor [\forall Y \text{ even}(\text{times}(X, Y))]] \land
   [\neg \text{even}(\text{times}(X, \text{sk1}(X)))] \lor \text{even}(X)].\]

After step 5: \([\neg \text{even}(X) \lor \text{even}(\text{times}(X, Y))] \land
   [\neg \text{even}(\text{times}(X, \text{sk1}(X)))] \lor \text{even}(X)].\)

Step 6 has no effect.

After step 7: \(\neg \text{even}(X) \lor \text{even}(\text{times}(X, Y)).\)
\(\neg \text{even}(\text{times}(X, \text{sk1}(X))) \lor \text{even}(X).\)
4 Rule of 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 $p(X, Y) \lor \neg q(Y, Z)$, and $\sigma$ is the substitution above, then $\phi\sigma$ is $p(a,f(Z)) \lor q(f(Z),Z)$.

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 substitutions $\sigma_A$ 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 $Y$ ... 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).

4.1 Resolution Rule

Let $\phi$ be the clause $\alpha_1 \lor \alpha_2 \lor \ldots \lor \alpha_k$, and let $\psi$ be the clause $\beta_1 \lor \beta_2 \lor \ldots \lor \beta_m$. 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 $(\phi - \alpha_i)\sigma_A \lor (\psi - \beta_j)\sigma_B$.

From “man(socrates)” and “$\neg \text{man}(X) \lor \text{mortal}(X)$”, infer “mortal(socrates)”.

From “man(socrates)” and “$\neg \text{man}(X)$” infer the empty clause.
4.1.1 Comments on resolution

**Comment 1:** Resolution is much easier to carry out correctly, both manually and in implementation, if one uses separation of variables. That is if you wish to resolve together two clauses that use the same variable name, first change the name(s) in one clause to avoid the conflict, then resolve.

For example, to resolve clause CA: \(\neg\text{loves}(\text{sam},X) \lor \text{reads}(X,\text{nytimes})\) (Sam owns only people who read the NY Times) with Clause CB: \(\neg\text{read}(\text{mary},X) \lor \text{regencyromance}(X)\) (Mary reads only Regency Romances), you should:

1. Change \(X\) in CA to \(X_1\) giving \(\neg\text{read}(\text{mary},X_1) \lor \text{regencyromance}(X_1)\)

2. Applying the substitutions \(X\rightarrow\text{mary}\) and \(X_1\rightarrow\text{nytimes}\), resolve the two clauses, giving
   \(\neg\text{loves}(\text{sam},\text{mary}) \lor \text{regencyromance}(\text{nytimes})\).
   (If Sam loves Mary, then the NY Times is a Regency Romance.)

**Comment 2:** There may be two or more different ways to resolve two clauses together.

Let A be the clause \(\neg\text{gt}(X,Y) \lor \text{gt}(\text{plus}(X,1),Y)\).
Let B be the clause \(\neg\text{gt}(X_1,Y_1) \lor \text{gt}(\text{plus}(X_1,2),Y_1)\).

There are two possible resolutions:

1. The second literal of A can be resolved against the first literal of B, under the substitution \(X_1\rightarrow\text{plus}(X,1), Y_1\rightarrow Y\), giving the new clause \(\neg\text{gt}(X,Y) \lor \text{gt}(\text{plus}(\text{plus}(X,1),2),Y)\)

2. The first literal of A can be resolved against the second literal of B, under the substitution \(X\rightarrow\text{plus}(X_1,2), Y\rightarrow Y_1\), giving the new clause \(\neg\text{gt}(X_1,Y_1) \lor \text{gt}(\text{plus}(\text{plus}(X_1,2),1),Y_1)\)

**Comment 3:** It may be possible to resolve a clause against itself.

Example: Let A be the clause \(\neg\text{gt}(X,Y) \lor \text{gt}(\text{plus}(X,1),Y)\).
Make a copy of A, separating variables:
A1 is the clause \(\neg\text{gt}(X_1,Y_1) \lor \text{gt}(\text{plus}(X_1,1),Y_1)\).
Resolve the second literal of A with the first literal of A1 using the substitution \(X_1\rightarrow\text{plus}(X,1), Y_1\rightarrow Y\), giving the new clause \(\neg\text{gt}(X,Y) \lor \text{gt}(\text{plus}(\text{plus}(X,1),1),Y)\)

**Comment 4:** It is NOT ALLOWED to resolve on more than one literal in each clause at a time.

Example: Let A be the clause \(\neg\text{cow}(X) \lor \text{bovine}(X)\) (All cows are bovines) and let B the clause \(\neg\text{bovine}(X) \lor \text{cow}(X) \lor \text{bull}(X)\) (All bovines are cows or bulls.) It is NOT CORRECT to simultaneously resolve the first literal of A against the second literal of B and resolve the second literal of A against the first literal of B and derive a new clause “bull(X)” (Everything is a bull.)
4.2 Factoring Rule

Let $\phi$ be the clause $\alpha_1 \lor \alpha_2 \lor \ldots \lor \alpha_k$. 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: Given the clause “$p(a,X) \lor p(Y,b) \lor q(X,Y,c)$”, apply the substitution $X \rightarrow b$, $Y \rightarrow a$ to derive the new clause. “$p(a,b) \lor q(b,a,c)$”.

4.2.1 Comments on factoring rule

Comment 1: There are cases where a clause can be factored in more than one way. For example, the clause “$p(a,X) \lor p(b,X) \lor p(b,c) \lor p(a,d)$” can be factored, either by using the substitution $X \rightarrow c$ giving “$p(a,c) \lor p(b,c) \lor p(a,d)$” or using the substitution $X \rightarrow d$ giving “$p(a,d) \lor p(b,d) \lor p(b,c)$.”

Comment 2: Separation of variables is not possible for factoring.

Comment 3: Factoring is rarely necessary, and there are important classes of resolution problems (e.g. Horn clauses) where it is entirely unnecessary, but there are cases where the proof cannot be completed without it.

5 Resolution Proof Algorithm

resolution(in GAMMA : set of sentences; PHI : sentence) return boolean /* TRUE if PHI is a consequence of GAMMA; FALSE otherwise */

{ DELTA := GAMMA union {not PHI};
  DELTA := Convert DELTA to clausal form;
  for every clause C in DELTA
    for every factoring F of C
      add F to DELTA;
  repeat { pick clauses A and B from DELTA  /* PICKING STEP */
    if A and B can be resolved together then {
      NEWCS := set of resolvents of A with B;
      NEWCS := NEWCS cup all factorings of NEWCS;
      if (empty clause in NEWCS) then return(TRUE)
      DELTA := DELTA union NEWCS
    }
  } until (every pair of clauses in DELTA has been considered at PICKING STEP);
  return(FALSE)
Facts:
If resolution returns TRUE then PHI is a consequence of GAMMA.
If resolution returns FALSE then PHI is a consequence of GAMMA.
Suppose that resolution is implemented so that each pair A,B in DELTA is eventually considered for resolving (even if the “until” condition is never met.) If PHI is a consequence of GAMMA, then resolution will eventually return TRUE.
Regardless of how resolution is implemented, there are cases where PHI is not a consequence of GAMMA, but the algorithm never terminates. This is not unique to this algorithm; it is true of any algorithm for inference in first-order logic.

5.1 Example

Given: 1. \( \forall_{S_1,S_2} \text{subset}(S_1,S_2) \Leftrightarrow [\forall_X \text{member}(X,S_1) \Rightarrow \text{member}(X,S_2)] \).
Prove: H. \( \forall_{S_1,S_2,S_3} [\text{subset}(S_1,S_2) \land \text{subset}(S_2,S_3)] \Rightarrow \text{subset}(S_1,S_3) \).

Negation of H: 2. \( \neg[\forall_{S_1,S_2,S_3} [\text{subset}(S_1,S_2) \land \text{subset}(S_2,S_3)] \Rightarrow \text{subset}(S_1,S_3) \].

Converted to clausal form:
1a. \( \neg\text{subset}(S_1,S_2) \lor \neg\text{member}(X,S_1) \lor \text{member}(X,S_2) \).
1b. \( \text{member}(sk0(S1,S2),S1) \lor \text{subset}(S1,S2) \).
1c. \( \neg\text{member}(sk0(S1,S2),S2) \lor \text{subset}(S1,S2) \).
2a. \( \text{subset}(sk1,sk2) \).
2b. \( \text{subset}(sk2,sk3) \).
2c. \( \neg\text{subset}(sk1,sk3) \).

From 2a and 1a, infer
3. \( \neg\text{member}(X,sk1) \lor \text{member}(X,sk2) \).
From 2b and 1a, infer
4. \( \neg\text{member}(X,sk2) \lor \text{member}(X,sk3) \).
From 3 and 4, infer
5. \( \neg\text{member}(X,sk1) \lor \text{member}(X,sk3) \).
From 2c and 1b infer
6. \( \text{member}(sk0(sk1,sk3),sk1) \).
From 2c and 1c infer
7. \( \neg\text{member}(sk0(sk1,sk3),sk3) \).
From 6 and 5 infer
8. \( \text{member}(sk0(sk1,sk3),sk3) \).
From 7 and 8 infer
9. The empty clause.
5.2 Efficient implementation

The following comparatively efficient implementation uses the “set of support” strategy:

resolution(in GAMMA : set of sentences; PHI : sentence)
  return boolean /* TRUE if PHI is a consequence of GAMMA; FALSE otherwise */

{ OLD := Convert GAMMA to clausal form;
  OLD := OLD union all factorings of all sentences in OLD;
  NEW := Convert "not PHI" to clausal form;
  NEW := NEW union all factorings of all sentences in NEW;
  NEWQ := FIFO queue containing clauses in NEWQ
  repeat {
    A := pop(NEWQ);
    OLD := add A to OLD;
    for (B in OLD) do {
      for every resolvent R of A and B do {
        if (R == empty clause) return(TRUE);
        if (R is not in OLD and R is not in NEWQ) then {
          add R to the end of NEWQ;
          for every factoring F of R do
            if (F is not in OLD and F is not in NEWQ) then
              add F to the end of NEWQ;
        } /* endif */
      } /* endfor */
    } /* endfor */
  } /* end body of repeat */
  until NEWQ is empty;
  return(FALSE)