Resolution for Predicate Calculus

Lila Kari

University of Waterloo
Sherlock Holmes and Dr. Watson go on a camping trip. After a good dinner and a bottle of wine, they retire for the night, and go to sleep. Some hours later, Holmes wakes up and nudges his faithful friend. "Watson, look up at the sky and tell me what you see."

"I see millions and millions of stars, Holmes," replies Watson.

"And what do you deduce from that?", says Holmes.
Watson ponders for a minute: “Well, astronomically, it tells me that there are millions of galaxies and potentially billions of planets. Astrologically, I observe that Saturn is in Leo. Horologically, I deduce that the time is approximately a quarter past three. Meteorologically, I suspect that we will have a beautiful day tomorrow. Theologically, I can see that God is all powerful, and that we are a small and insignificant part of the universe. What does it tell you, Holmes?” Holmes is silent for a moment.
Resolution for Predicate Calculus

For resolution in predicate calculus, and for other purposes, it is often more convenient to deal with formulas in which all quantifiers have been moved to the front of the expression. These types of formulas are said to be in prenex normal form.

Definition A formula is in prenex normal form if it is of the form

\[ Q_1 x_1 \ Q_2 x_2 \ldots \ Q_n x_n B \]

where \( Q_i(i = 1, \ldots n) \) is \( \forall \) or \( \exists \) and the formula \( B \) is quantifier free.

The string \( Q_1 x_1 \ Q_2 x_2 \ldots Q_n x_n \) is called the prefix and \( B \) is called the matrix.

A formula with no quantifiers is regarded as a trivial case of a prenex normal form.
Example

Which of the following expressions are in prenex normal form?

\[ \forall x P(x) \lor \forall x Q(x) \]
\[ \forall x \forall y \neg(P(x) \rightarrow Q(y)) \]
\[ \forall x \exists y R(x, y) \]
\[ R(x, y) \]
\[ \neg \forall x R(x, y) \]
Algorithm for converting a formula into prenex normal form

Any formula can be converted into prenex normal form. To do this, the following steps are needed:

1. Eliminate all occurrences of $\rightarrow$ and $\leftrightarrow$ from the formula in question.

2. Move all negations inward such that, in the end, negations only appear as part of literals.

3. Standardize the variables apart (when necessary).

4. The prenex normal form can now be obtained by moving all quantifiers to the front of the formula.
To accomplish **Step 1** (eliminate →, ↔), make use of the following logical equivalences:

- \( A \rightarrow B \models \neg A \lor B \).
- \( A \leftrightarrow B \models (\neg A \lor B) \land (A \lor \neg B) \).
- \( A \leftrightarrow B \models (A \land B) \lor (\neg A \land \neg B) \).

To accomplish **Step 2** (move all negations inward, such that negations only appear as parts of literals), use the logical equivalences:

- De Morgan’s Laws.
- \( \neg \neg A \models A \).
- \( \neg \exists x A(x) \models \forall x \neg A(x) \).
- \( \neg \forall x A(x) \models \exists x \neg A(x) \).
To accomplish Step 3 (standardize variables apart), make use of the following theorem which allows one to rename the variables in order to make them distinct. Renaming the variables in a formula such that distinct variables have distinct names is called standardizing the variables apart.

**Theorem (Replaceability of bound variable symbols)**

Suppose $A$ results from $A'$ by replacing in $A$ some (not necessarily all) occurrences of $Q(x)B(x)$ by $Q(y)B(y)$. Then $A \models A'$ and $A \vdash A'$.

**Note:** As before, $Q$ denotes either the existential or the universal quantifier.
Example - Standardize variables apart

Standardize all variables apart in the following formula:

\[ \forall x(P(x) \rightarrow Q(x)) \land \exists xQ(x) \land \exists zP(z) \land \exists z(Q(z) \rightarrow R(z)) \]

Solution: Use \( y \) for \( x \) in \( \forall x \), \( u \) for \( x \) in \( \exists xQ(x) \), and \( w \) for \( z \) in \( \exists zP(z) \) to obtain:

\[ \forall y(P(y) \rightarrow Q(y)) \land \exists uQ(u) \land \exists wP(w) \land \exists z(Q(z) \rightarrow R(z)) \]
To accomplish Step 4 (move all quantifiers in front of the formula) make use of the following logical equivalences:

- $A \land \exists x B(x) \iff \exists x (A \land B(x))$, $x$ not occurring in $A$.
- $A \land \forall x B(x) \iff \forall x (A \land B(x))$, $x$ not occurring in $A$.
- $A \lor \exists x B(x) \iff \exists x (A \lor B(x))$, $x$ not occurring in $A$.
- $A \lor \forall x B(x) \iff \forall x (A \lor B(x))$, $x$ not occurring in $A$.

(These equivalences essentially show that if a formula $A$ has a truth value that does not depend on $x$, then one is allowed to quantify over $x$. )
More logical equivalences for Step 4

- \( \forall x A(x) \land \forall x B(x) \models \forall x (A(x) \land B(x)) \).
- \( \exists x A(x) \lor \exists x B(x) \models \exists x (A(x) \lor B(x)) \).
- \( \forall x \forall y A(x, y) \models \forall y \forall x A(x, y) \).
- \( \exists x \exists y A(x, y) \models \exists y \exists x A(x, y) \).
- \( Q_1 x A(x) \land Q_2 y B(y) \models Q_1 x Q_2 y (A(x) \land B(y)) \).
- \( Q_1 x A(x) \lor Q_2 y B(y) \models Q_1 x Q_2 y (A(x) \lor B(y)) \).

Where \( Q_1, Q_2 \in \{\forall, \exists\} \).

For example, if \( Q_1 = \forall \) and \( Q_2 = \exists \), we have
\( \forall x A(x) \land \exists y B(y) \models \forall x \exists y (A(x) \land B(y)) \).
Example - Prenex normal form

Find the prenex normal form of

$$\forall x (\exists y R(x, y) \land \forall y \neg S(x, y) \rightarrow \neg \exists y R(x, y))$$

Solution:

- According to Step 1, we must eliminate $\rightarrow$, which yields

$$\forall x (\neg (\exists y R(x, y) \land \forall y \neg S(x, y)) \lor \neg \exists y R(x, y))$$

- We move all negations inwards, which yields:

$$\forall x (\forall y \neg R(x, y) \lor \exists y S(x, y) \lor \forall y \neg R(x, y))$$

- Next, all variables are standardized apart:

$$\forall x (\forall y_1 \neg R(x, y_1) \lor \exists y_2 S(x, y_2) \lor \forall y_3 \neg R(x, y_3))$$

- We can now move all quantifiers in front, which yields

$$\forall x \forall y_1 \exists y_2 \forall y_3 (\neg R(x, y_1) \lor S(x, y_2) \lor \neg R(x, y_3))$$
Exercise

Transform the following formula into prenex normal form:

\[ \neg[\forall x \exists y F(u, x, y) \rightarrow \exists x (\neg\forall y G(y, v) \rightarrow H(x))] \]

Solution:

\[ \forall x \exists y \exists z [F(u, x, y) \land \neg G(z, v) \land \neg H(x)] \]
Recall that a sentence is a formula without free variables.

**Definition** A sentence $A \in L^{pred}$ is said to be in $\exists$-free prenex normal form if it is in prenex normal form and does not contain existential quantifier symbols.
Consider a sentence of the form

$$\forall x_1 \forall x_2 \ldots \forall x_n \exists y A$$

where $A$ is a sentence, possibly involving quantifiers.

- Note that $\exists y A$ generates at least one individual for each $n$-tuple $x_1, x_2, \ldots x_n$.
- As a consequence, we can think of one of these generated individuals as a function of $x_1, x_2, \ldots x_n$.
- This can be expressed using the term $f(x_1, x_2, \ldots x_n)$ to denote one of these newly created individuals.
- This function is called a **Skolem function**.
Eliminate existential quantifiers: Skolem functions

- Skolem functions allow one to remove all existential quantifiers.
  The skolemized version of $\forall x_1 \forall x_2 \ldots \forall x_n \exists y A$ is
  
  $(*): \forall x_1 \forall x_2 \ldots \forall x_n A'$

  where $A'$ is the sentence obtained from $A$ by substituting each occurrence of $y$ by $f(x_1, x_2, \ldots x_n)$.

- For instance, in $\forall x \exists y P(y, x)$, one has a different $y$ for each $x$, which means that $\exists y P(y, x)$ leads to $\forall x P(g(x), x)$.
  Here $g(x)$ is the individual created for each $x$ by the existential instantiation of $\exists y P(y, x)$.
• Note that the sentence obtained by using Skolem functions is not in general logically equivalent to the original sentence. This happens because it is very well possible that the intended interpretation has more than one individual arising from the existential quantifier. However, for our purposes, it is irrelevant how many individuals satisfy $A$ in the sentence $\exists x A$, as long as there exists at least one individual.

• It is convenient to consider constants as functions of zero arguments. If this is done, then the skolemized sentence $(\ast)$ remains valid even if an existential quantifier is not preceded by any universal quantifier.

• For any sentence in $\mathcal{L}^{\text{pred}}$ we can generate an $\exists$-free prenex normal form sentence by using the following algorithm.
Algorithm for an ∃-free prenex normal form

- **Step 1.** Transform the given sentence into a sentence $A_1$ in prenex normal form. Set $i = 1$.
- **Step 2.** Repeat until all the existential quantifiers are removed.

Assume $A_i$ is of the form $A_i = \forall x_1 \forall x_2 \ldots \forall x_n \exists y A$ where $A$ is a sentence, possibly involving quantifiers.

If $n = 0$, then $A_i$ is of the form $\exists y A$. Then $A_{i+1} = A'$, where $A'$ is obtained from $A$ by replacing all occurrences of $y$ by the constant $c$.

If $n > 0$, $A_{i+1} = \forall x_1 \forall x_2 \ldots \forall x_n A'$, where $A'$ is the sentence obtained from $A$ by replacing all occurrences of $y$ by $f(x_1, x_2, \ldots x_n)$.

Increase $i$ by 1.
Example - Existential-free PNF

Transform the following sentence into $\exists$-free prenex normal form:

$$\exists x \forall y \forall z \exists t P(x, y, z, t)$$

The left-most existential quantifier is $\exists x$. Since there is no universal quantifier preceding $\exists x$, $x$ can be replaced by a Skolem function with zero arguments, or a constant. Use $a$ for this purpose. Then

$$A_2 = \forall y \forall z \exists t P(a, y, z, t).$$

There are two universal quantifiers preceding $\exists t$, which means that the Skolem function corresponding to $t$ must have two arguments, $y$ and $z$. We use $g(y, z)$ for this Skolem function. We obtain

$$A_3 = \forall y \forall z P(a, y, z, g(y, z)).$$
Theorem. Given a sentence $F$ in $L^{pred}$, there is an effective procedure for finding an $\exists$-free prenex normal form formula $F'$ such that $F$ is satisfiable iff $F'$ is satisfiable.

Note. After all the existential quantifiers are eliminated through the use of Skolem functions, it is customary to drop the universal quantifiers also. This means that all variables are implicitly universally quantified.

(Recall that we only deal with sentences, i.e., we do not have free variables.)
From formulas in predicate calculus to clauses

**Theorem** Given a sentence \( F \) in \( \exists \)-free prenex normal form, one can effectively construct a finite set \( C_F \) of clauses such that \( F \) is satisfiable iff \( C_F \) is satisfiable.

**Example.** Construct the set of clauses \( C_F \) for

\[
F = \forall x \forall y \forall z (R(x, y) \rightarrow (R(x, z) \land R(z, y)))
\]

First, we put the matrix of \( F \) in conjunctive normal form.

\[
R(x, y) \rightarrow (R(x, z) \land R(z, y)) \models
\]

\[
\neg R(x, y) \lor (R(x, z) \land R(z, y)) \models
\]

\[
(\neg R(x, y) \lor R(x, z)) \land (\neg R(x, y) \lor R(z, y)).
\]

Now we can read off the clauses from the conjuncts, namely,

\[
C_F = \{ \neg R(x, y) \lor R(x, z), \neg R(x, y) \lor R(z, y) \}.
\]
Theorem. Given a set $\Sigma$ of sentences (premises), and $A$ a sentence (conclusion),

$$\Sigma \models A$$

iff the set

$$C_{\neg A} \cup \left[ \bigcup_{F \in \Sigma} C_F \right]$$

is not satisfiable.

In other words, the argument in predicate calculus is valid, that is the conclusion $A$ follows from the premises, iff the set of clauses consisting of the union of

- $\bigcup_{F \in \Sigma} C_F$ : the sets of clauses obtained from each premise $F$ in $\Sigma$, and
- $C_{\neg A}$: the set of clauses generated by the negation of the conclusion $A$

is not satisfiable.
Example - translate an argument into clauses

Let

\[ \Sigma = \{ \forall x R(x, x), \ \forall x \forall y (R(x, y) \rightarrow R(y, x)), \]
\[ \forall x \forall y \forall z ((R(x, y) \land R(y, z)) \rightarrow R(x, z)) \]

and the conclusion

\[ F = \forall x \forall y (\neg R(x, y) \rightarrow \forall t (R(x, t) \rightarrow \neg R(y, t))). \]

Translate the argument \( \Sigma \models F \) into clauses.
Solution. The negation of the conclusion is

\[ \neg F = \exists x \exists y (\neg R(x, y) \land \exists t (R(x, t) \land R(y, t))). \]

Putting \( \neg F \) in prenex normal form gives

\[ \exists x \exists y \exists t (\neg R(x, y) \land R(x, t) \land R(y, t)). \]

The premises are already skolemized, as they are \( \exists - \)free sentences in prenex normal form.
For \( \neg F \) we obtain the skolemized form

\[ \neg R(a, b) \land R(a, c) \land R(b, c). \]
Thus, the set of clauses corresponding to the argument is

\[
R(x, x) \\
\neg R(x, y) \lor R(y, x) \\
\neg R(x, y) \lor \neg R(y, z) \lor R(x, z) \\
\neg R(a, b) \\
R(a, c) \\
R(b, c).
\]

To show that this set of clauses is not satisfiable, we can use resolution.
Theorem. A set $S$ of clauses is not satisfiable iff there is a derivation of the empty clause (of a contradiction) by resolution.

The way to perform resolution in predicate calculus will be explained in the following.
In resolution we aim to reach a contradiction. In propositional calculus, it is impossible that from a set of formulas one can derive a contradiction unless the same variable occurs more than once. For instance, there is no way to derive a contradiction from the two formulas $P \land Q \lor R$ and $\neg S$. The two formulas do not share variables, and the truth of the first has no bearing on the truth of the second.

Similarly, in predicate calculus, one cannot derive a contradiction unless the formulas in question share atomic formulas that are either equal or can be made equal. This is done through a procedure called unification.
Definition  Two formulas are said to unify if there are legal instantiations (assignments of terms to variables) that make the formulas in question identical. The act of unifying is called unification. The instantiation that unifies the formulas in question is called a unifier.
Example. Unify the formulas $Q(a, y, z)$ and $Q(y, b, c)$.

Solution. Since $y$ in $Q(a, y, z)$ is a different variable than $y$ in $Q(y, b, c)$, rename $y$ in the second formula to become $y_1$. This means that one must unify $Q(a, y, z)$ with $Q(y_1, b, c)$.

An instance of $Q(a, y, z)$ is $Q(a, b, c)$ and an instance of $Q(y_1, b, c)$ is $Q(a, b, c)$.

Since these two instances are identical, $Q(a, y, z)$ and $Q(y, b, c)$ unify.

The unifier is $y_1 = a$, $y = b$, $z = c$. 
Resolution theorem proving

- A theorem is a logical argument in the sense that it has several premises and a conclusion.
- To prove a theorem automatically (show that the logical argument is valid), we first transform the premises and negation of the conclusion into a set of clauses.
- If the obtained set of clauses is not satisfiable, then the theorem (argument) is correct.
- The set of clauses is not satisfiable iff a contradiction can be derived.
• Any search for a contradiction in a set of clauses can be restricted to formulas that can be unified.

• **Unification** is therefore basic to efficient refutation methods.

• Resolution theorem proving uses **unification** combined with **resolution** to obtain an efficient refutation method.
In resolution theorem proving, one works with clauses. That is, given the set of premises and the negation of the conclusion:

- each formula is converted into prenex normal form.
- the existential quantifiers are replaced by Skolem functions.
- the universal quantifiers are dropped.
- the resulting quantifier-free sentences are converted into clauses, i.e., they are written as a set of disjunctions.
• Resolution theorem proving is based on literals.
• A positive literal is an atomic formula.
• A negative literal is the negation of an atomic formula.
• For example, $P(x, y)$ and $Q(z)$ are positive literals while $\neg P(x, y)$ and $\neg Q(x)$ are negative literals.
Resolution can only be applied to formulas that contain complementary literals.

The idea is to create complementary literals by means of unification and determine the resolvent.

Note that the variables occurring in different clause are distinct.

Each clause is really universally quantified, except that the quantifiers are dropped.
Example - finding a resolvent by unification

Find the resolvent of the following two clauses:

\[ G(a, x, y) \lor H(y, x) \lor D(z). \]

\[ \neg G(x, c, y) \lor H(f(x), b) \lor E(a). \]

Here \( a, b, c \) are constants and \( x, y, z \) are variables.
Solution To obtain two complementary literals, we unify $G(a, x, y)$ in the first clause with $G(x, c, y)$ in the second clause. We can do this as follows: Since $x, y, z$ in the 1st clause are (implicitly) universally quantified, we can replace these variables by any term. In particular, we can set $x := c$, which yields

$$(*) G(a, c, y) \lor H(y, c) \lor D(z).$$

Similarly, one can replace the variables in the second clause by any term. We set $x := a$ and obtain

$$(**)^{\neg} G(a, c, y) \lor H(f(a), b) \lor E(a)$$
Comments

Note that the variables $x$ and $y$ in the two formulas have to be considered as distinct variables, which allowed us to instantiate $x$ in the first formula to $c$ and in the second formula to $a$.

This also means that $y$ in the first formula is different from $y$ in the 2nd formula and it is therefore necessary to explicitly unify these two variables, even though they have the same name.

Once this is done, the resolvent of ($\ast$) and ($\ast\ast$) can be found readily as

$$H(y, c) \lor D(z) \lor H(f(a), b) \lor E(a).$$
Example of resolution theorem proving

Prove that everybody has a grandparent, provided everybody has a parent.

Let $P(x, y)$ represent $x$ is a parent of $y$. The premise can now be stated as $\forall x \exists y P(y, x)$. From this we must be able to conclude that there exists a parent of a parent, which can be expressed as $\forall x \exists y \exists z (P(z, y) \land P(y, x))$.

We must thus prove that

$$\forall x \exists y P(y, x) \models \forall x \exists y \exists z (P(z, y) \land P(y, x)).$$
Example contd.

We add the negation of the conclusion to the set of premises, which yields:

\[ \forall x \exists y P(y, x), \exists x \forall y \forall z (\neg P(z, y) \lor \neg P(y, x)). \]

Eliminate the existential quantifiers (obtain the \( \exists \)-free prenex normal form of the formulas) to obtain:

\[ \forall x P(f(x), x), \forall y \forall z (\neg P(z, y) \lor \neg P(y, a)). \]

After dropping the universal quantifiers, this yields

\[ P(f(x), x), \neg P(z, y) \lor \neg P(y, a). \]
Resolution can now be used to find a contradiction as follows:

1. \( P(f(x), x) \)  
   Given
2. \( \neg P(z, y) \lor \neg P(y, a) \)  
   Given
3. \( P(f(a), a) \)  
   (\( \forall - \)) of 1 with \( x := a \)
4. \( \neg P(z, f(a)) \lor \neg P(f(a), a) \)  
   (\( \forall - \)) of 2, \( y := f(a) \)
5. \( \neg P(z, f(a)) \)  
   Resolve 3 and 4
6. \( P(f(f(a)), f(a)) \)  
   (\( \forall - \)), 1, \( x := f(a) \)
7. \( \neg P(f(f(a)), f(a)) \)  
   (\( \forall - \)), 5, \( z := f(f(a)) \)
8. \( \{ \} \)  
   contradiction from 6, 7

This means that the original argument is valid.
Resolution strategies

- In resolution theorem proving it is important to follow an appropriate strategy.
- Applicable strategies: **unit clause resolution** and **set of support strategy**.
Example: Use resolution with the set of support strategy to prove that a relation $R \subseteq A \times A$ is reflexive if it is transitive and symmetric.

Solution. Translate in terms of logic.
Define $R(x, y)$ to be true if $x$ is related to $y$.
Since $A$ is the domain we have

$$\forall x \exists y R(x, y).$$

The relation is transitive, that is,

$$\forall x \forall y \forall z (R(x, y) \land R(y, z) \rightarrow R(x, z)).$$

The relation is symmetric, that is

$$\forall x \forall y (R(x, y) \rightarrow R(y, x)).$$

We want to conclude from these premises that the relation is reflexive, that is,

$$\forall x R(x, x).$$

To prove the theorem by refutation, negate the conclusion

$$\exists x \neg R(x, x).$$
The set of clauses is

\[ R(x, f(x)) \]
\[ \neg R(x, y) \lor \neg R(y, z) \lor R(x, z) \]
\[ \neg R(x, y) \lor R(y, x) \]
\[ \neg R(a, a). \]
Example contd.

- We use the set of support strategy with the negation of the conclusion as the only formula in the initial set of support.
- If the strategy leaves more than one choice, we prefer resolution against unit clauses, provided the unit clause contains a constant.
- Specifically, we apply all such unit clauses until none can be applied anymore before trying the next clause.
- Any resolvent that is tautologically true, such as $R(a, a) \lor \neg R(a, a)$ is avoided.
- Whenever a unit clause is found, it is compared to other unit clauses to see whether a contradiction can be derived.
1. \( \neg R(a, a) \) given
2. \( R(x, f(x)) \) given
3. \( \neg R(x, y) \lor \neg R(y, z) \lor R(x, z) \) given
4. \( \neg R(x, y) \lor R(y, x) \) given
5. \( \neg R(a, y) \lor \neg R(y, a) \lor R(a, a) \) 3, \( x := a, z := a \)
6. \( \neg R(a, y) \lor \neg R(y, a) \) resolve 1, 5
7. \( R(a, f(a)) \) 2 with \( x := a \)
8. \( \neg R(a, f(a)) \lor \neg R(f(a), a) \) 6 with \( y := f(a) \)
9. \( \neg R(f(a), a) \) resolve 7, 8
10. \( \neg R(a, f(a)) \lor R(f(a), a) \) 4, \( x := a, y := f(a) \)
11. \( \neg R(a, f(a)) \) resolve 9, 10
12. \( R(a, f(a)) \) 2 with \( x := a \)
13. \( \{} \) contrad. 11, 12
Automated Theorem Provers and Verifiers

- Because statements of a formal theory are written in symbolic form, it is possible to verify mechanically that a formal proof from a finite set of axioms is valid.

- This task, known as automatic proof verification, is closely related to automated theorem proving.

- The difference is that instead of constructing a new proof, the proof verifier simply checks that a provided formal proof (or the sequence of instructions that can be followed to create a formal proof) is correct.

- This process is not merely hypothetical; systems such as Isabelle and Coq are used today to formalize proofs and then check their validity.
Isabelle Proof Assistant

- **Isabelle** (U.Cambridge, Technische Universitat Munchen, 2016) is an interactive theorem prover.
- It is a Higher-Order Logic theorem prover.
- It allows mathematical formulas to be expressed in a formal language and provides tools for proving those formulas in a logical calculus.
- **Isabelle**’s main proof method is a higher-order version of resolution, based on higher-order unification.
- See [http://isabelle.in.tum.de/](http://isabelle.in.tum.de/)
Coq Proof Assistant

- **Coq** (INRIA, Ecole Polytechnique, Univ. Paris-Sud, Paris Diderot Univ., CNRS, Ecole Normale Superieure de Lyon, 2016) is an interactive theorem prover.
- It allows the expression of mathematical assertions, mechanically checks proofs of these assertions, helps to find formal proofs, and extracts a certified program from the constructive proof of its formal specification.
- **Coq** is not an automated theorem prover but includes automatic theorem proving tactics and various decision procedures.
- See [https://coq.inria.fr/](https://coq.inria.fr/)
Theorem Provers

- **E** is a theorem prover for full first-order logic with equality. It accepts a problem specification, typically consisting of a number of first-order clauses or formulas, and a conjecture, again either in clausal or full first-order form. The system will then try to find a formal proof for the conjecture, assuming the axioms.


- **SPASS** is an automated theorem prover for first-order logic with equality


- **Vampire** is a theorem prover (a system able to prove theorems) in first-order logic


- **CASC** is a yearly competition of fully automated theorem provers for classical first order logic.