Resolution Principle

Mădălina Erașcu

West University of Timișoara and Institute e-Austria Timișoara
bvd. V. Parvan 4, Timișsoara, Romania

madalina.erascu@e-uvt.ro

Based on the book: Chin-Liang Chang and Richard Char-Tung Lee. Symbolic Logic and Mechanical Theorem Proving, Chapter 5

Outline

Outline

Preliminaries

Goal: finding a decision procedure (algorithm) to verify the validity/inconsistency/unsatisfiability of a formula.

Bad news: there is no general decision procedure to decide the validity of a formula in **first-order logic** (FOL) (due to Church and Turing).

Good news: there are proof procedures which can verify if a FOL formula is valid if it is indeed valid.

Bad news: if a FOL formula is invalid there is no proof procedure proving that (they will not terminate).

Resolution principle works directly on a set of clauses to test the unsatisfiability of a set.

Idea: check whether the set S of clauses contains the $\square$ (empty) clause. If so, then S is unsatisfiable. If not, check whether it can be derived.

Preliminaries

Goal: finding a decision procedure (algorithm) to verify the validity/inconsistency/unsatisfiability of a formula.

Bad news: there is no general decision procedure to decide the validity of a formula in **first-order logic** (FOL) (due to Church and Turing).

Good news: there are proof procedures which can verify if a FOL formula is valid if it is indeed valid.

Bad news: if a FOL formula is invalid there is no proof procedure proving that (they will not terminate).

Resolution principle works directly on a set of clauses to test the unsatisfiability of a set.

Idea: check whether the set S of clauses contains the $\square$ (empty) clause. If so, then S is unsatisfiable. If not, check whether it can be derived.

Preliminaries

Goal: finding a decision procedure (algorithm) to verify the validity/inconsistency/unsatisfiability of a formula.

Bad news: there is no general decision procedure to decide the validity of a formula in **first-order logic** (FOL) (due to Church and Turing).

Good news: there are proof procedures which can verify if a FOL formula is valid if it is indeed valid.

Bad news: if a FOL formula is invalid there is no proof procedure proving that (they will not terminate).

Resolution principle works directly on a set of clauses to test the unsatisfiability of a set.

Idea: check whether the set S of clauses contains the $\square$ (empty) clause. If so, then S is unsatisfiable. If not, check whether it can be derived.

Preliminaries

Goal: finding a decision procedure (algorithm) to verify the validity/inconsistency/unsatisfiability of a formula.

Bad news: there is no general decision procedure to decide the validity of a formula in **first-order logic** (FOL) (due to Church and Turing).

Good news: there are proof procedures which can verify if a FOL formula is valid if it is indeed valid.

Bad news: if a FOL formula is invalid there is no proof procedure proving that (they will not terminate).

Resolution principle works directly on a set of clauses to test the unsatisfiability of a set.

Idea: check whether the set S of clauses contains the $\Box$ (empty) clause. If so, then S is unsatisfiable. If not, check whether it can be derived.

Preliminaries

Goal: finding a decision procedure (algorithm) to verify the validity/inconsistency/unsatisfiability of a formula.

Bad news: there is no general decision procedure to decide the validity of a formula in **first-order logic** (FOL) (due to Church and Turing).

Good news: there are proof procedures which can verify if a FOL formula is valid if it is indeed valid.

Bad news: if a FOL formula is invalid there is no proof procedure proving that (they will not terminate).

Resolution principle works directly on a set of clauses to test the unsatisfiability of a set.

Idea: check whether the set S of clauses contains the $\square$ (empty) clause. If so, then S is unsatisfiable. If not, check whether it can be derived.

Preliminaries

Goal: finding a decision procedure (algorithm) to verify the validity/inconsistency/unsatisfiability of a formula.

Bad news: there is no general decision procedure to decide the validity of a formula in **first-order logic** (FOL) (due to Church and Turing).

Good news: there are proof procedures which can verify if a FOL formula is valid if it is indeed valid.

Bad news: if a FOL formula is invalid there is no proof procedure proving that (they will not terminate).

Resolution principle works directly on a set of clauses to test the unsatisfiability of a set.

Idea: check whether the set S of clauses contains the $\square$ (empty) clause. If so, then S is unsatisfiable. If not, check whether it can be derived.

Resolution Principle for the Propositional Logic

Definition

For any two clauses C_1 and C_2, if there is a literal L_1 in C_1 that is complementary to a literal L_2 in C_2, then delete L_1 and L_2 from C_1 and C_2, respectively, and construct the disjunction of the remaining clauses. The constructed clause is a resolvent of C_1 and C_2.

Resolution Principle for the Propositional Logic

Definition

For any two clauses C_1 and C_2, if there is a literal L_1 in C_1 that is complementary to a literal L_2 in C_2, then delete L_1 and L_2 from C_1 and C_2, respectively, and construct the disjunction of the remaining clauses. The constructed clause is a resolvent of C_1 and C_2.

Example

$$C_1 : P \vee R$$
$$C_2 : \neg P \vee Q$$

The resolvent of C_1 and C_2 is $R \vee Q$.

Resolution Principle for the Propositional Logic

Definition

For any two clauses C_1 and C_2, if there is a literal L_1 in C_1 that is complementary to a literal L_2 in C_2, then delete L_1 and L_2 from C_1 and C_2, respectively, and construct the disjunction of the remaining clauses. The constructed clause is a resolvent of C_1 and C_2.

Example

$$
\begin{aligned}
C_1 &: \neg P \vee Q \vee R \\
C_2 &: \neg Q \vee R
\end{aligned}
$$

The resolvent of C_1 and C_2 is $\neg P \vee R \vee S$.

Resolution Principle for the Propositional Logic

Definition

For any two clauses C_1 and C_2, if there is a literal L_1 in C_1 that is complementary to a literal L_2 in C_2, then delete L_1 and L_2 from C_1 and C_2, respectively, and construct the disjunction of the remaining clauses. The constructed clause is a resolvent of C_1 and C_2.

Example

$$C_1 : \neg P \vee Q$$
$$C_2 : \neg P \vee R$$

There is no resolvent of C_1 and C_2.

Resolution Principle for the Propositional Logic (cont'd)

Theorem
Given two clauses C_1 and C_2, a resolvent C of C_1 and C_2 is a logical consequence of C_1 and C_2.

Observations:

- If we have two unit clauses, then the resolvent of them, if there is one, is the empty clause $\square$.
- If a set S of clauses is unsatisfiable, we can use the resolution principle to generate $\square$ from S.

Resolution Principle for the Propositional Logic (cont'd)

Theorem

Given two clauses C_1 and C_2, a resolvent C of C_1 and C_2 is a logical consequence of C_1 and C_2.

Observations:

- If we have two unit clauses, then the resolvent of them, if there is one, is the empty clause $\Box$.
- If a set S of clauses is unsatisfiable, we can use the resolution principle to generate $\Box$ from S.

Resolution Principle for the Propositional Logic (cont'd)

Theorem

Given two clauses C_1 and C_2, a resolvent C of C_1 and C_2 is a logical consequence of C_1 and C_2.

Observations:

- If we have two unit clauses, then the resolvent of them, if there is one, is the empty clause $\Box$.
- If a set S of clauses is unsatisfiable, we can use the resolution principle to generate $\Box$ from S.

Resolution Principle for the Propositional Logic (cont'd)

Theorem

Given two clauses C_1 and C_2, a resolvent C of C_1 and C_2 is a logical consequence of C_1 and C_2.

Observations:

- If we have two unit clauses, then the resolvent of them, if there is one, is the empty clause $\square$.
- If a set S of clauses is unsatisfiable, we can use the resolution principle to generate $\square$ from S.

Example

Let S be

$$\neg P \vee Q \tag{1}$$

$$\neg Q \tag{2}$$

$$P \tag{3}$$

From (1) and (2), we obtain P (4); from (3) and (4), we obtain $\square$. Hence, $\square$ is a logical consequence of S. Hence, S is unsatisfiable.

Resolution Principle for the Propositional Logic (cont'd)

Theorem

Given two clauses C_1 and C_2, a resolvent C of C_1 and C_2 is a logical consequence of C_1 and C_2.

Observations:

- If we have two unit clauses, then the resolvent of them, if there is one, is the empty clause $\Box$.
- If a set S of clauses is unsatisfiable, we can use the resolution principle to generate $\Box$ from S.

Example

Let S be

$$P \vee Q \tag{1}$$

$$\neg P \vee Q \tag{2}$$

$$P \vee \neg Q \tag{3}$$

$$\neg P \vee \neg Q \tag{4}$$

From (1) and (2), we obtain Q (5); from (3) and (4), we obtain $\neg Q$ (6). From (5) and (6), we obtain $\Box$. Hence, S is unsatisfiable.

Substitution

Motivation: apply resolution principle to FOL formulas.

Example: Let

$$C_1 : \quad P[x] \lor Q[x]$$
$$C_2 : \quad \neg P[f[x]] \lor R[x]$$

Substitution

Motivation: apply resolution principle to FOL formulas.

Example: Let

$$\begin{array}{ll} C_1: & P[x] \lor Q[x] \\ C_2: & \neg P[f[x]] \lor R[x] \end{array}$$

Substitution

Motivation: apply resolution principle to FOL formulas.

Example: Let

$$\begin{array}{ll} C_1 : & P[x] \vee Q[x] \\ C_2 : & \neg P[f[x]] \vee R[x] \end{array}$$

Let $x \to f[a]$ in C_1, $x \to a$ in C_2.

Substitution

Motivation: apply resolution principle to FOL formulas.

Example: Let

$$\begin{array}{ll} C_1: & P[x] \vee Q[x] \\ C_2: & \neg P[f[x]] \vee R[x] \end{array}$$

Let $x \to f[a]$ in C_1, $x \to a$ in C_2.

We have

$$\begin{array}{ll} C_1': & P[f[a]] \vee Q[f[a]] \\ C_2': & \neg P[f[a]] \vee R[a] \end{array}$$

Substitution

Motivation: apply resolution principle to FOL formulas.

Example: Let

$$\begin{array}{ll} C_1: & P[x] \vee Q[x] \\ C_2: & \neg P[f[x]] \vee R[x] \end{array}$$

Let $x \to f[a]$ in C_1, $x \to a$ in C_2.

We have

$$\begin{array}{ll} C_1': & P[f[a]] \vee Q[f[a]] \\ C_2': & \neg P[f[a]] \vee R[a] \end{array}$$

C_1' and C_2' are ground (no variables) instances.

Substitution

Motivation: apply resolution principle to FOL formulas.

Example: Let

$$\begin{array}{ll} C_1: & P[x] \lor Q[x] \\ C_2: & \neg P[f[x]] \lor R[x] \end{array}$$

Let $x \to f[a]$ in C_1, $x \to a$ in C_2.

We have

$$\begin{array}{ll} C_1': & P[f[a]] \lor Q[f[a]] \\ C_2': & \neg P[f[a]] \lor R[a] \end{array}$$

C_1' and C_2' are ground (no variables) instances.

A resolvent of C_1' and C_2' is

$$C_3': \quad Q[f[a]] \lor R[a]$$

Substitution

Motivation: apply resolution principle to FOL formulas.

Example: Let

$$\begin{array}{ll} C_1 : & P[x] \vee Q[x] \\ C_2 : & \neg P[f[x]] \vee R[x] \end{array}$$

Let $x \to f[x]$ in C_1. We have

$$C_1^* : \quad P[f[x]] \vee Q[f[x]]$$

C_1^* is an instance of C_1.

A resolvent of

$$\begin{array}{ll} C_2 : & \neg P[f[x]] \vee R[x] \\ C_1^* : & P[f[x]] \vee Q[f[x]] \end{array}$$

is

$$C_3 : \quad Q[f[x]] \vee R[x]$$

C_3' is an instance of C_3. C_3 is the most general clause.

Substitution

Motivation: apply resolution principle to FOL formulas.

Example: Let

$$\begin{array}{ll} C_1: & P[x] \vee Q[x] \\ C_2: & \neg P[f[x]] \vee R[x] \end{array}$$

Let $x \to f[x]$ in C_1. We have

$$C_1^*: \quad P[f[x]] \vee Q[f[x]]$$

C_1^* is an instance of C_1.

A resolvent of

$$\begin{array}{ll} C_2: & \neg P[f[x]] \vee R[x] \\ C_1^*: & P[f[x]] \vee Q[f[x]] \end{array}$$

is

$$C_3: \quad Q[f[x]] \vee R[x]$$

C_3' is an instance of C_3. C_3 is the most general clause.

Substitution

Motivation: apply resolution principle to FOL formulas.

Example: Let

$$\begin{array}{ll} C_1 : & P[x] \vee Q[x] \\ C_2 : & \neg P[f[x]] \vee R[x] \end{array}$$

Let $x \to f[x]$ in C_1. We have

$$C_1^* : \quad P[f[x]] \vee Q[f[x]]$$

C_1^* is an instance of C_1.

A resolvent of

$$\begin{array}{ll} C_2 : & \neg P[f[x]] \vee R[x] \\ C_1^* : & P[f[x]] \vee Q[f[x]] \end{array}$$

is

$$C_3 : \quad Q[f[x]] \vee R[x]$$

C_3' is an instance of C_3. C_3 is the most general clause.

Substitution

Motivation: apply resolution principle to FOL formulas.

Example: Let

$$\begin{array}{ll} C_1: & P[x] \vee Q[x] \\ C_2: & \neg P[f[x]] \vee R[x] \end{array}$$

Let $x \to f[x]$ in C_1. We have

$$C_1^*: \quad P[f[x]] \vee Q[f[x]]$$

C_1^* is an instance of C_1.

A resolvent of

$$\begin{array}{ll} C_2: & \neg P[f[x]] \vee R[x] \\ C_1^*: & P[f[x]] \vee Q[f[x]] \end{array}$$

is

$$C_3: \quad Q[f[x]] \vee R[x]$$

C_3' is an instance of C_3. C_3 is the most general clause.

Substitution

Motivation: apply resolution principle to FOL formulas.

Example: Let

$$\begin{array}{ll} C_1: & P[x] \vee Q[x] \\ C_2: & \neg P[f[x]] \vee R[x] \end{array}$$

Let $x \to f[x]$ in C_1. We have

$$C_1^*: \quad P[f[x]] \vee Q[f[x]]$$

C_1^* is an instance of C_1.

A resolvent of

$$\begin{array}{ll} C_2: & \neg P[f[x]] \vee R[x] \\ C_1^*: & P[f[x]] \vee Q[f[x]] \end{array}$$

is

$$C_3: \quad Q[f[x]] \vee R[x]$$

C_3' is an instance of C_3. C_3 is the most general clause.

Substitution (cont'd)

A substitution σ is a finite set of the form $\{v_1 \to t_1, ..., v_n \to t_n\}$ where every t_i is a term different from v_i and no two elements in the set have the same variable v_i.

Let σ be defined as above and E be an expression. Then $E\sigma$ is an expression obtained from E by replacing simultaneously each occurrence of v_i in E by the term t_i

Example: Let $\sigma = \{x \to z, z \to h[a, y]\}$ and $E = f[z, a, g[x], y]$. Then $E\sigma = f[h[a, y], a, g[z], y]$.

Substitution (cont'd)

A substitution σ is a finite set of the form $\{v_1 \to t_1, \ldots, v_n \to t_n\}$ where every t_i is a term different from v_i and no two elements in the set have the same variable v_i.

Let σ be defined as above and E be an expression. Then $E\sigma$ is an expression obtained from E by replacing simultaneously each occurrence of v_i in E by the term t_i

Example: Let $\sigma = \{x \to z, z \to h[a, y]\}$ and $E = f[z, a, g[x], y]$. Then $E\sigma = f[h[a, y], a, g[z], y]$.

Substitution (cont'd)

A substitution σ is a finite set of the form $\{v_1 \to t_1, ..., v_n \to t_n\}$ where every t_i is a term different from v_i and no two elements in the set have the same variable v_i.

Let σ be defined as above and E be an expression. Then $E\sigma$ is an expression obtained from E by replacing simultaneously each occurrence of v_i in E by the term t_i

Example: Let $\sigma = \{x \to z, z \to h[a, y]\}$ and $E = f[z, a, g[x], y]$. Then $E\sigma = f[h[a, y], a, g[z], y]$.

Substitution (cont'd)

Let

$$\theta = \{x_1 \to t_1, ..., x_n \to t_n\}$$
$$\lambda = \{y_1 \to u_1, ..., y_n \to u_n\}$$

Then the composition of θ and λ ($\theta \circ \lambda$) is obtained from the set

$$\{x_1 \to t_1\lambda, ..., x_n \to t_n\lambda, y_1 \to u_1, ..., y_n \to u_n\}$$

by deleting any element $x_j \to t_j\lambda$ for which $x_j = t_j\lambda$ and any element $y_i \to u_i$ such that y_i is among $\{x_1, ..., x_n\}$.

Substitution (cont'd)

Example 1:

$$\theta = \{x \to f[y], y \to z\}$$
$$\lambda = \{x \to a, y \to b, z \to y\}$$

Then

$$\theta \circ \lambda = \{x \to f[b], y \to y, x \to a, y \to b, z \to y\}$$
$$= \{x \to f[b], z \to y\}$$

Example 2:

$$\theta_1 = \{x \to a, y \to f[z], z \to y\}$$
$$\theta_2 = \{x \to b, y \to z, z \to g[x]\}$$

Then

$$\theta_1 \circ \theta_2 = \{x \to a, y \to f[g[x]], z \to z, x \to b, y \to z, z \to g[x]\}$$
$$= \{x \to a, y \to f[g[x]]\}$$

Substitution (cont'd)

Example 1:

$$\theta = \{x \to f[y], y \to z\}$$
$$\lambda = \{x \to a, y \to b, z \to y\}$$

Then

$$\begin{aligned}\theta \circ \lambda &= \{x \to f[b], y \to y, x \to a, y \to b, z \to y\} \\ &= \{x \to f[b], z \to y\}\end{aligned}$$

Example 2:

$$\theta_1 = \{x \to a, y \to f[z], z \to y\}$$
$$\theta_2 = \{x \to b, y \to z, z \to g[x]\}$$

Then

$$\begin{aligned}\theta_1 \circ \theta_2 &= \{x \to a, y \to f[g[x]], z \to z, x \to b, y \to z, z \to g[x]\} \\ &= \{x \to a, y \to f[g[x]]\}\end{aligned}$$

Substitution (cont'd)

Example 1:

$$\theta = \{x \to f[y], y \to z\}$$
$$\lambda = \{x \to a, y \to b, z \to y\}$$

Then

$$\begin{aligned}\theta \circ \lambda &= \{x \to f[b], y \to y, x \to a, y \to b, z \to y\} \\ &= \{x \to f[b], z \to y\}\end{aligned}$$

Example 2:

$$\theta_1 = \{x \to a, y \to f[z], z \to y\}$$
$$\theta_2 = \{x \to b, y \to z, z \to g[x]\}$$

Then

$$\begin{aligned}\theta_1 \circ \theta_2 &= \{x \to a, y \to f[g[x]], z \to z, x \to b, y \to z, z \to g[x]\} \\ &= \{x \to a, y \to f[g[x]]\}\end{aligned}$$

Substitution (cont'd)

Example 1:

$$\theta = \{x \to f[y], y \to z\}$$
$$\lambda = \{x \to a, y \to b, z \to y\}$$

Then

$$\begin{aligned}\theta \circ \lambda &= \{x \to f[b], y \to y, x \to a, y \to b, z \to y\} \\ &= \{x \to f[b], z \to y\}\end{aligned}$$

Example 2:

$$\theta_1 = \{x \to a, y \to f[z], z \to y\}$$
$$\theta_2 = \{x \to b, y \to z, z \to g[x]\}$$

Then

$$\begin{aligned}\theta_1 \circ \theta_2 &= \{x \to a, y \to f[g[x]], z \to z, x \to b, y \to z, z \to g[x]\} \\ &= \{x \to a, y \to f[g[x]]\}\end{aligned}$$

Unification

A substitution θ is called a unifier for a set $\{E_1, ..., E_k\}$ iff $E_1\theta = ... = E_k\theta$. The set $\{E_1, ..., E_k\}$ is said to be unifiable iff there exists an unifier for it.

A unifier σ for a set $\{E_1, ..., E_k\}$ of expressions is a most general unifier iff for each unifier θ for the set there is a substitution λ such that $\theta = \sigma \circ \lambda$.

Example:
The set $\{P[a, y], P[x, f[b]]\}$ is unifiable since $\sigma = \{x \to a, y \to f[b]\}$ is a unifier for the set.

Unification

A substitution θ is called a unifier for a set $\{E_1, ..., E_k\}$ iff $E_1\theta = ... = E_k\theta$. The set $\{E_1, ..., E_k\}$ is said to be unifiable iff there exists an unifier for it.

A unifier σ for a set $\{E_1, ..., E_k\}$ of expressions is a most general unifier iff for each unifier θ for the set there is a substitution λ such that $\theta = \sigma \circ \lambda$.

Example:
The set $\{P[a, y], P[x, f[b]]\}$ is unifiable since $\sigma = \{x \rightarrow a, y \rightarrow f[b]\}$ is a unifier for the set.

Unification

A substitution θ is called a unifier for a set $\{E_1, ..., E_k\}$ iff $E_1\theta = ... = E_k\theta$. The set $\{E_1, ..., E_k\}$ is said to be unifiable iff there exists an unifier for it.

A unifier σ for a set $\{E_1, ..., E_k\}$ of expressions is a most general unifier iff for each unifier θ for the set there is a substitution λ such that $\theta = \sigma \circ \lambda$.

Example:
The set $\{P[a, y], P[x, f[b]]\}$ is unifiable since $\sigma = \{x \to a, y \to f[b]\}$ is a unifier for the set.

Unification Algorithm

Unification algorithm for finding a most general unifier (mgu), or its nonexistence, for a finite set of nonempty expressions.

The disagreement set of a nonempty set W of expressions is obtained by

- locating the first symbol (starting from the left) at which not all the expressions in W have exactly the same symbol and then
- extracting from each expression in W the subexpression that begins with the symbol occupying that position.

Example: The disagreement set of $\{P[a, x, f[g[y]]], P[z, f[z], f[u]]\}$ is $\{a, z\}$.

Unification Algorithm

Unification algorithm for finding a most general unifier (mgu), or its nonexistence, for a finite set of nonempty expressions.

The disagreement set of a nonempty set W of expressions is obtained by

- locating the first symbol (starting from the left) at which not all the expressions in W have exactly the same symbol and then
- extracting from each expression in W the subexpression that begins with the symbol occupying that position.

Example: The disagreement set of $\{P[a, x, f[g[y]]], P[z, f[z], f[u]]\}$ is $\{a, z\}$.

Unification Algorithm

Unification algorithm for finding a most general unifier (mgu), or its nonexistence, for a finite set of nonempty expressions.

The disagreement set of a nonempty set W of expressions is obtained by

- locating the first symbol (starting from the left) at which not all the expressions in W have exactly the same symbol and then
- extracting from each expression in W the subexpression that begins with the symbol occupying that position.

Example: The disagreement set of $\{P[a, x, f[g[y]]], P[z, f[z], f[u]]\}$ is $\{a, z\}$.

Unification Algorithm

Unification algorithm for finding a most general unifier (mgu), or its nonexistence, for a finite set of nonempty expressions.

The disagreement set of a nonempty set W of expressions is obtained by

- locating the first symbol (starting from the left) at which not all the expressions in W have exactly the same symbol and then
- extracting from each expression in W the subexpression that begins with the symbol occupying that position.

Example: The disagreement set of $\{P[a, x, f[g[y]]], P[z, f[z], f[u]]\}$ is $\{a, z\}$.

Unification Algorithm

Unification algorithm for finding a most general unifier (mgu), or its nonexistence, for a finite set of nonempty expressions.

The disagreement set of a nonempty set W of expressions is obtained by

- locating the first symbol (starting from the left) at which not all the expressions in W have exactly the same symbol and then
- extracting from each expression in W the subexpression that begins with the symbol occupying that position.

Example: The disagreement set of $\{P[a, x, f[g[y]]], P[z, f[z], f[u]]\}$ is $\{a, z\}$.

Unification Algorithm (cont'd)

Unification Algorithm

1. $k := 0$, $W_k := W$, $\sigma_k := \varepsilon$
2. If W_k is singleton then stop; σ_k is mgu of W. Otherwise find the disagreement set D_k of W_k.
3. If there exists $v_k,\ t_k \in D_k$ s.t. v_k is a variable which does not occur in t_k, go to 4. Otherwise, stop; W is not unifiable.
4. Let $\sigma_{k+1} = \sigma_k \circ \{v_k \to t_k\}$ and $W_{k+1} = W_k\{v_k \to t_k\}$.
5. $k = k + 1$ and go to 2.

Example: Find a most general unifier for

1. $W = \{P[a, x, f[g[y]]],\ P[z, f[z], f[u]]\}$
2. $W = \{Q[a],\ Q[b]\}$
3. $W = \{P[x],\ P[f[x]]\}$
4. $W = \{P[x],\ Q[y]\}$

Unification Algorithm (cont'd)

Unification Algorithm

1. $k := 0$, $W_k := W$, $\sigma_k := \varepsilon$
2. If W_k is singleton then stop; σ_k is mgu of W. Otherwise find the disagreement set D_k of W_k.
3. If there exists $v_k,\ t_k \in D_k$ s.t. v_k is a variable which does not occur in t_k, go to 4. Otherwise, stop; W is not unifiable.
4. Let $\sigma_{k+1} = \sigma_k \circ \{v_k \to t_k\}$ and $W_{k+1} = W_k\{v_k \to t_k\}$.
5. $k = k + 1$ and go to 2.

Example: Find a most general unifier for

1. $W = \{P[a, x, f[g[y]]],\ P[z, f[z], f[u]]\}$
2. $W = \{Q[a],\ Q[b]\}$
3. $W = \{P[x],\ P[f[x]]\}$
4. $W = \{P[x],\ Q[y]\}$

Unification Algorithm (cont'd)

Unification Algorithm

1. $k := 0$, $W_k := W$, $\sigma_k := \varepsilon$
2. If W_k is singleton then stop; σ_k is mgu of W. Otherwise find the disagreement set D_k of W_k.
3. If there exists $v_k,\ t_k \in D_k$ s.t. v_k is a variable which does not occur in t_k, go to 4. Otherwise, stop; W is not unifiable.
4. Let $\sigma_{k+1} = \sigma_k \circ \{v_k \to t_k\}$ and $W_{k+1} = W_k\{v_k \to t_k\}$.
5. $k = k + 1$ and go to 2.

Example: Find a most general unifier for

1. $W = \{P[a, x, f[g[y]]],\ P[z, f[z], f[u]]\}$
2. $W = \{Q[a],\ Q[b]\}$
3. $W = \{P[x],\ P[f[x]]\}$
4. $W = \{P[x],\ Q[y]\}$

Unification Algorithm (cont'd)

Unification Algorithm

1. $k := 0$, $W_k := W$, $\sigma_k := \varepsilon$
2. If W_k is singleton then stop; σ_k is mgu of W. Otherwise find the disagreement set D_k of W_k.
3. If there exists $v_k,\ t_k \in D_k$ s.t. v_k is a variable which does not occur in t_k, go to 4. Otherwise, stop; W is not unifiable.
4. Let $\sigma_{k+1} = \sigma_k \circ \{v_k \to t_k\}$ and $W_{k+1} = W_k\{v_k \to t_k\}$.
5. $k = k + 1$ and go to 2.

Example: Find a most general unifier for

1. $W = \{P[a, x, f[g[y]]],\ P[z, f[z], f[u]]\}$
2. $W = \{Q[a],\ Q[b]\}$
3. $W = \{P[x],\ P[f[x]]\}$
4. $W = \{P[x],\ Q[y]\}$

Unification Algorithm (cont'd)

Unification Algorithm

1. $k := 0$, $W_k := W$, $\sigma_k := \varepsilon$
2. If W_k is singleton then stop; σ_k is mgu of W. Otherwise find the disagreement set D_k of W_k.
3. If there exists $v_k,\ t_k \in D_k$ s.t. v_k is a variable which does not occur in t_k, go to 4. Otherwise, stop; W is not unifiable.
4. Let $\sigma_{k+1} = \sigma_k \circ \{v_k \to t_k\}$ and $W_{k+1} = W_k\{v_k \to t_k\}$.
5. $k = k + 1$ and go to 2.

Example: Find a most general unifier for

1. $W = \{P[a, x, f[g[y]]],\ P[z, f[z], f[u]]\}$
2. $W = \{Q[a],\ Q[b]\}$
3. $W = \{P[x],\ P[f[x]]\}$
4. $W = \{P[x],\ Q[y]\}$

Unification Algorithm (cont'd)

Unification Algorithm

1. $k := 0$, $W_k := W$, $\sigma_k := \varepsilon$
2. If W_k is singleton then stop; σ_k is mgu of W. Otherwise find the disagreement set D_k of W_k.
3. If there exists $v_k,\ t_k \in D_k$ s.t. v_k is a variable which does not occur in t_k, go to 4. Otherwise, stop; W is not unifiable.
4. Let $\sigma_{k+1} = \sigma_k \circ \{v_k \to t_k\}$ and $W_{k+1} = W_k\{v_k \to t_k\}$.
5. $k = k + 1$ and go to 2.

Example: Find a most general unifier for

1. $W = \{P[a, x, f[g[y]]],\ P[z, f[z], f[u]]\}$
2. $W = \{Q[a],\ Q[b]\}$
3. $W = \{P[x],\ P[f[x]]\}$
4. $W = \{P[x],\ Q[y]\}$

Unification Algorithm (cont'd)

Unification Algorithm

1. $k := 0$, $W_k := W$, $\sigma_k := \varepsilon$
2. If W_k is singleton then stop; σ_k is mgu of W. Otherwise find the disagreement set D_k of W_k.
3. If there exists $v_k,\ t_k \in D_k$ s.t. v_k is a variable which does not occur in t_k, go to 4. Otherwise, stop; W is not unifiable.
4. Let $\sigma_{k+1} = \sigma_k \circ \{v_k \to t_k\}$ and $W_{k+1} = W_k\{v_k \to t_k\}$.
5. $k = k + 1$ and go to 2.

Example: Find a most general unifier for

1. $W = \{P[a, x, f[g[y]]],\ P[z, f[z], f[u]]\}$
2. $W = \{Q[a],\ Q[b]\}$
3. $W = \{P[x],\ P[f[x]]\}$
4. $W = \{P[x],\ Q[y]\}$

Unification Algorithm (cont'd)

Unification Algorithm

1. $k := 0$, $W_k := W$, $\sigma_k := \varepsilon$
2. If W_k is singleton then stop; σ_k is mgu of W. Otherwise find the disagreement set D_k of W_k.
3. If there exists $v_k,\ t_k \in D_k$ s.t. v_k is a variable which does not occur in t_k, go to 4. Otherwise, stop; W is not unifiable.
4. Let $\sigma_{k+1} = \sigma_k \circ \{v_k \to t_k\}$ and $W_{k+1} = W_k\{v_k \to t_k\}$.
5. $k = k + 1$ and go to 2.

Example: Find a most general unifier for

1. $W = \{P[a, x, f[g[y]]],\ P[z, f[z], f[u]]\}$
2. $W = \{Q[a],\ Q[b]\}$
3. $W = \{P[x],\ P[f[x]]\}$
4. $W = \{P[x],\ Q[y]\}$

Unification Algorithm (cont'd)

Unification Algorithm

1. $k := 0$, $W_k := W$, $\sigma_k := \varepsilon$
2. If W_k is singleton then stop; σ_k is mgu of W. Otherwise find the disagreement set D_k of W_k.
3. If there exists $v_k,\ t_k \in D_k$ s.t. v_k is a variable which does not occur in t_k, go to 4. Otherwise, stop; W is not unifiable.
4. Let $\sigma_{k+1} = \sigma_k \circ \{v_k \to t_k\}$ and $W_{k+1} = W_k\{v_k \to t_k\}$.
5. $k = k + 1$ and go to 2.

Example: Find a most general unifier for

1. $W = \{P[a, x, f[g[y]]],\ P[z, f[z], f[u]]\}$
2. $W = \{Q[a],\ Q[b]\}$
3. $W = \{P[x],\ P[f[x]]\}$
4. $W = \{P[x],\ Q[y]\}$

Unification Algorithm (cont'd)

Unification Algorithm

1. $k := 0$, $W_k := W$, $\sigma_k := \varepsilon$
2. If W_k is singleton then stop; σ_k is mgu of W. Otherwise find the disagreement set D_k of W_k.
3. If there exists $v_k,\ t_k \in D_k$ s.t. v_k is a variable which does not occur in t_k, go to 4. Otherwise, stop; W is not unifiable.
4. Let $\sigma_{k+1} = \sigma_k \circ \{v_k \rightarrow t_k\}$ and $W_{k+1} = W_k\{v_k \rightarrow t_k\}$.
5. $k = k + 1$ and go to 2.

Example: Find a most general unifier for

1. $W = \{P[a, x, f[g[y]]],\ P[z, f[z], f[u]]\}$
2. $W = \{Q[a],\ Q[b]\}$
3. $W = \{P[x],\ P[f[x]]\}$
4. $W = \{P[x],\ Q[y]\}$

Unification Algorithm (cont'd)

Unification Algorithm

1. $k := 0$, $W_k := W$, $\sigma_k := \varepsilon$
2. If W_k is singleton then stop; σ_k is mgu of W. Otherwise find the disagreement set D_k of W_k.
3. If there exists $v_k,\ t_k \in D_k$ s.t. v_k is a variable which does not occur in t_k, go to 4. Otherwise, stop; W is not unifiable.
4. Let $\sigma_{k+1} = \sigma_k \circ \{v_k \to t_k\}$ and $W_{k+1} = W_k\{v_k \to t_k\}$.
5. $k = k + 1$ and go to 2.

Example: Find a most general unifier for

1. $W = \{P[a, x, f[g[y]]],\ P[z, f[z], f[u]]\}$
2. $W = \{Q[a],\ Q[b]\}$
3. $W = \{P[x],\ P[f[x]]\}$
4. $W = \{P[x],\ Q[y]\}$

Resolution Principle for FOL

- Unification algorithm allows application of the resolution principle the same as for propositional logic.
- Resolution principle is complete i.e. □ is always derived if the set of clauses is unsatisfiable.

How does resolution work?
Given: formulas F_1, ..., F_n
Prove: G by resolution.

1. Bring F_1, ..., F_n, ..., $\neg G$ into standard form and write the clauses which are obtained
2. Start derivation and try to obtain the empty clause from the set C of clauses.
3. In the derivation use resolution inference rule and factoring rules to derive new clauses; these new clauses are added to C.
4. If the empty clause appears, stop: Contradiction found, G is proved.
5. If no step can be made and the empty clause is not found, then H can not be proved.

Resolution Principle for FOL

- Unification algorithm allows application of the resolution principle the same as for propositional logic.
- Resolution principle is complete i.e. $\square$ is always derived if the set of clauses is unsatisfiable.

How does resolution work?
Given: formulas F_1, ..., F_n
Prove: G by resolution.

1. Bring F_1, ..., F_n, ..., $\neg G$ into standard form and write the clauses which are obtained
2. Start derivation and try to obtain the empty clause from the set C of clauses.
3. In the derivation use resolution inference rule and factoring rules to derive new clauses; these new clauses are added to C.
4. If the empty clause appears, stop: Contradiction found, G is proved.
5. If no step can be made and the empty clause is not found, then H can not be proved.

Resolution Principle for FOL

- Unification algorithm allows application of the resolution principle the same as for propositional logic.
- Resolution principle is complete i.e. $\square$ is always derived if the set of clauses is unsatisfiable.

How does resolution work?

Given: formulas F_1, ..., F_n

Prove: G by resolution.

1. Bring F_1, ..., F_n, ..., $\neg G$ into standard form and write the clauses which are obtained
2. Start derivation and try to obtain the empty clause from the set C of clauses.
3. In the derivation use resolution inference rule and factoring rules to derive new clauses; these new clauses are added to C.
4. If the empty clause appears, stop: Contradiction found, G is proved.
5. If no step can be made and the empty clause is not found, then H can not be proved.

Resolution Principle for FOL

- Unification algorithm allows application of the resolution principle the same as for propositional logic.
- Resolution principle is complete i.e. □ is always derived if the set of clauses is unsatisfiable.

How does resolution work?
Given: formulas F_1, ..., F_n
Prove: G by resolution.

1. Bring F_1, ..., F_n, ..., $\neg G$ into standard form and write the clauses which are obtained
2. Start derivation and try to obtain the empty clause from the set C of clauses.
3. In the derivation use resolution inference rule and factoring rules to derive new clauses; these new clauses are added to C.
4. If the empty clause appears, stop: Contradiction found, G is proved.
5. If no step can be made and the empty clause is not found, then H can not be proved.

Resolution Principle for FOL

- Unification algorithm allows application of the resolution principle the same as for propositional logic.
- Resolution principle is complete i.e. $\square$ is always derived if the set of clauses is unsatisfiable.

How does resolution work?
Given: formulas F_1, ..., F_n
Prove: G by resolution.

1. Bring F_1, ..., F_n, ..., $\neg G$ into standard form and write the clauses which are obtained
2. Start derivation and try to obtain the empty clause from the set C of clauses.
3. In the derivation use resolution inference rule and factoring rules to derive new clauses; these new clauses are added to C.
4. If the empty clause appears, stop: Contradiction found, G is proved.
5. If no step can be made and the empty clause is not found, then H can not be proved.

Resolution Principle for FOL

- Unification algorithm allows application of the resolution principle the same as for propositional logic.
- Resolution principle is complete i.e. $\Box$ is always derived if the set of clauses is unsatisfiable.

How does resolution work?
Given: formulas F_1, ..., F_n
Prove: G by resolution.

1. Bring F_1, ..., F_n, ..., $\neg G$ into standard form and write the clauses which are obtained
2. Start derivation and try to obtain the empty clause from the set C of clauses.
3. In the derivation use resolution inference rule and factoring rules to derive new clauses; these new clauses are added to C.
4. If the empty clause appears, stop: Contradiction found, G is proved.
5. If no step can be made and the empty clause is not found, then H can not be proved.

Resolution Principle for FOL

- Unification algorithm allows application of the resolution principle the same as for propositional logic.
- Resolution principle is complete i.e. $\square$ is always derived if the set of clauses is unsatisfiable.

How does resolution work?
Given: formulas F_1, ..., F_n
Prove: G by resolution.

1. Bring F_1, ..., F_n, ..., $\neg G$ into standard form and write the clauses which are obtained
2. Start derivation and try to obtain the empty clause from the set C of clauses.
3. In the derivation use resolution inference rule and factoring rules to derive new clauses; these new clauses are added to C.
4. If the empty clause appears, stop: Contradiction found, G is proved.
5. If no step can be made and the empty clause is not found, then H can not be proved.

Resolution Principle for FOL

- Unification algorithm allows application of the resolution principle the same as for propositional logic.
- Resolution principle is complete i.e. $\square$ is always derived if the set of clauses is unsatisfiable.

How does resolution work?
Given: formulas F_1, ..., F_n
Prove: G by resolution.

1. Bring F_1, ..., F_n, ..., $\neg G$ into standard form and write the clauses which are obtained
2. Start derivation and try to obtain the empty clause from the set C of clauses.
3. In the derivation use resolution inference rule and factoring rules to derive new clauses; these new clauses are added to C.
4. If the empty clause appears, stop: Contradiction found, G is proved.
5. If no step can be made and the empty clause is not found, then H can not be proved.

Resolution Principle for FOL

- Unification algorithm allows application of the resolution principle the same as for propositional logic.
- Resolution principle is complete i.e. $\square$ is always derived if the set of clauses is unsatisfiable.

How does resolution work?
Given: formulas F_1, ..., F_n
Prove: G by resolution.

1. Bring F_1, ..., F_n, ..., $\neg G$ into standard form and write the clauses which are obtained
2. Start derivation and try to obtain the empty clause from the set C of clauses.
3. In the derivation use resolution inference rule and factoring rules to derive new clauses; these new clauses are added to C.
4. If the empty clause appears, stop: Contradiction found, G is proved.
5. If no step can be made and the empty clause is not found, then H can not be proved.

Resolution Principle for FOL. Examples

Example 0: Let

$$\begin{array}{ll} C_1 : & P[x] \vee Q[x] \\ C_2 : & \neg P[a] \vee R[x] \end{array}$$

Apply resolution.

Example 1: Prove by resolution the following

$$\forall_x F[x] \vee \forall_x H[x] \quad \not\equiv \quad \forall_x (F[x] \vee H[x])$$

Example 2: Prove by resolution that G is a logical consequence of F_1 and F_2 where

$$\begin{array}{ll} F_1 : & \forall_x (C[x] \Rightarrow (W[x] \wedge R[x])) \\ F_2 : & \exists_x (C[x] \wedge O[x]) \\ G : & \exists_x (O[x] \wedge R[x]) \end{array}$$

Resolution Principle for FOL. Examples

Example 0: Let

$$\begin{array}{ll} C_1: & P[x] \vee Q[x] \\ C_2: & \neg P[a] \vee R[x] \end{array}$$

Apply resolution.

Example 1: Prove by resolution the following

$$\underset{x}{\forall} F[x] \vee \underset{x}{\forall} H[x] \quad \not\equiv \quad \underset{x}{\forall} (F[x] \vee H[x])$$

Example 2: Prove by resolution that G is a logical consequence of F_1 and F_2 where

$$\begin{array}{ll} F_1: & \underset{x}{\forall} (C[x] \Rightarrow (W[x] \wedge R[x])) \\ F_2: & \underset{x}{\exists} (C[x] \wedge O[x]) \\ G: & \underset{x}{\exists} (O[x] \wedge R[x]) \end{array}$$

Resolution Principle for FOL. Examples

Example 0: Let

$$\begin{array}{ll} C_1: & P[x] \lor Q[x] \\ C_2: & \neg P[a] \lor R[x] \end{array}$$

Apply resolution.

Example 1: Prove by resolution the following

$$\underset{x}{\forall} F[x] \lor \underset{x}{\forall} H[x] \quad \not\equiv \quad \underset{x}{\forall} (F[x] \lor H[x])$$

Example 2: Prove by resolution that G is a logical consequence of F_1 and F_2 where

$$\begin{array}{ll} F_1: & \underset{x}{\forall} (C[x] \Rightarrow (W[x] \land R[x])) \\ F_2: & \underset{x}{\exists} (C[x] \land O[x]) \\ G: & \underset{x}{\exists} (O[x] \land R[x]) \end{array}$$

Resolution Principle for FOL. Examples (cont'd)

Example 3: Prove by resolution that G is a logical consequence of F_1 and F_2 where

$$
\begin{aligned}
F_1 : &\quad \underset{x}{\exists}\left(P[x] \;\wedge\; \underset{y}{\forall}(D[y] \;\Rightarrow\; L[x,y])\right)\\
F_2 : &\quad \underset{x}{\forall}\left(P[x] \;\Rightarrow\; \underset{y}{\forall}(Q[y] \;\Rightarrow\; \neg L[x,y])\right)\\
G : &\quad \underset{x}{\forall}(D[x] \;\Rightarrow\; \neg Q[x])
\end{aligned}
$$

Example 4: Prove by resolution that G is a logical consequence of F where

$$
\begin{aligned}
F : &\quad \underset{x}{\forall}\underset{y}{\exists}(S[x,y] \;\wedge\; M[y]) \;\;\Rightarrow\;\; \underset{y}{\exists}(I[y] \;\wedge\; E[x,y])\\
G : &\quad \neg\underset{x}{\exists}I[x] \;\;\Rightarrow\;\; \underset{x,y}{\forall}(S[x,y] \Rightarrow \neg M[y])
\end{aligned}
$$

Resolution Principle for FOL. Examples (cont'd)

Example 3: Prove by resolution that G is a logical consequence of F_1 and F_2 where

$$
\begin{array}{ll}
F_1 : & \underset{x}{\exists}\left(P[x] \ \wedge \ \underset{y}{\forall}(D[y] \ \Rightarrow \ L[x,y])\right) \\
F_2 : & \underset{x}{\forall}\left(P[x] \ \Rightarrow \ \underset{y}{\forall}(Q[y] \ \Rightarrow \ \neg L[x,y])\right) \\
G : & \underset{x}{\forall}(D[x] \ \Rightarrow \ \neg Q[x])
\end{array}
$$

Example 4: Prove by resolution that G is a logical consequence of F where

$$
\begin{array}{ll}
F : & \underset{x}{\forall}\underset{y}{\exists}(S[x,y] \ \wedge \ M[y]) \ \Rightarrow \ \underset{y}{\exists}(I[y] \ \wedge \ E[x,y]) \\
G : & \neg\underset{x}{\exists}I[x] \ \Rightarrow \ \underset{x,y}{\forall}(S[x,y] \Rightarrow \neg M[y])
\end{array}
$$

Resolution Principle for FOL. Examples (cont'd)

Example 5: Prove by resolution that G is a logical consequence of F_1, F_2, and F_3 where

$$
\begin{aligned}
F_1: &\quad \underset{x}{\forall}(Q[x] \;\Rightarrow\; \neg P[x]) \\
F_2: &\quad \underset{x}{\forall}\Big((R[x] \;\wedge\; \neg Q[x]) \;\;\Rightarrow\;\; \underset{y}{\exists}(T[x,y] \;\wedge\; S[y])\Big) \\
F_3: &\quad \underset{x}{\exists}\Big(P[x] \;\wedge\; \underset{y}{\forall}(T[x,y] \;\Rightarrow\; P[y]) \;\wedge\; R[x]\Big) \\
G: &\quad \underset{x}{\exists}(S[x] \wedge P[x])
\end{aligned}
$$