Basis for automated proof: First-Order Resolution

Frédéric Prost

Université Grenoble Alpes

March 2023

Plan

Plan

Idea

Skolemization yields formulae without quantifiers.

Then we must find an insatisfiable set of instances, either by trial and error or by exhaustive enumeration.

This lecture presents a generalization of resolution to first-order logic:

- Clausal form of skolemized formulae
- Resolution over clauses with variables
- Correctness and completeness of the method

Plan

Litteral, clause

Definition 5.2.19

A positive litteral is an atomic formula. Eg: $P(x,y)$

A negative litteral is the negation of an atomic formula. Eg: $\neg Q(a)$

A clause is a disjunction of litterals. Eg: $P(x,y) \vee \neg Q(a)$

Clausal form of a formula

Definition 5.2.20

The clausal form of a closed formula A is obtained in two steps:

1. Skolemize A (which yields a normal form without quantifiers)
2. Distribute $\vee$ over $\wedge$ to get a set of clauses Γ

Clausal form of a formula

Definition 5.2.20

The clausal form of a closed formula A is obtained in two steps:

1. Skolemize A (which yields a normal form without quantifiers)
2. Distribute $\vee$ over $\wedge$ to get a set of clauses Γ

Property 5.2.21

$\forall(\Gamma)$ has a model if and only if A has a model. More precisely:

- A is a consequence of $\forall(\Gamma)$
- If A has a model, then $\forall(\Gamma)$ has a model

Proof: We already know that skolemization preserves satisfiability. Then, distributivity yields a formula equivalent to the Skolem form.

Clausal form of a set of formulae

Definition 5.2.22

Let $\Gamma = A_1, \ldots, A_n$ be a set of closed formulae.
The clausal form of Γ is the union of the clausal forms of $A_1, \ldots, A_n$, paying attention, in the course of skolemization, to use a new symbol for each eliminated $\exists$.

Corollary 5.2.23

Let Γ be a set of closed formulae and Δ its clausal form:

- Γ is a consequence of $\forall(\Delta)$
- if Γ has a model then $\forall(\Delta)$ has a model.

Adapting Herbrand's theorem to clausal forms

Theorem 5.2.24

Let Γ be a set of closed formulae and Δ its clausal form:

Γ is unsatisfiable

if and only if

there exists a finite unsatisfiable subset of instances of clauses of Δ.

Proof.

- Skolemization preserves satisfiability
- Then we apply Herbrand's theorem to $\forall(\Delta)$

$\square$

Example 5.2.25

Let $A = \exists y \forall z (P(z,y) \Leftrightarrow \neg \exists x (P(z,x) \wedge P(x,z)))$. We compute the clausal form of A.

Example 5.2.25

Let $A = \exists y \forall z (P(z,y) \Leftrightarrow \neg\exists x (P(z,x) \wedge P(x,z)))$. We compute the clausal form of A.

1-4. The four steps of Skolemzation yield:
$(\neg P(z,a) \vee \neg P(z,x) \vee \neg P(x,z)) \;\wedge\; (P(z,f(z)) \wedge P(f(z),z) \;\vee\; P(z,a))$

Example 5.2.25

Let $A = \exists y \forall z (P(z,y) \Leftrightarrow \neg \exists x (P(z,x) \wedge P(x,z)))$. We compute the clausal form of A.

1-4. The four steps of Skolemzation yield:
$(\neg P(z,a) \vee \neg P(z,x) \vee \neg P(x,z)) \;\; \wedge \;\; (P(z,f(z)) \wedge P(f(z),z) \;\vee\; P(z,a))$

5. The clausal form is the following set of clauses:
   - $C_1 = \neg P(z,a) \vee \neg P(z,x) \vee \neg P(x,z)$
   - $C_2 = P(z,f(z)) \vee P(z,a)$
   - $C_3 = P(f(z),z) \vee P(z,a)$

Example 5.2.25

Let $A = \exists y \forall z (P(z,y) \Leftrightarrow \neg \exists x (P(z,x) \wedge P(x,z)))$. We compute the clausal form of A.

1-4. The four steps of Skolemzation yield:
$(\neg P(z,a) \vee \neg P(z,x) \vee \neg P(x,z)) \;\wedge\; (P(z,f(z)) \wedge P(f(z),z) \;\vee\; P(z,a))$

5. The clausal form is the following set of clauses:
   - $C_1 = \neg P(z,a) \vee \neg P(z,x) \vee \neg P(x,z)$
   - $C_2 = P(z,f(z)) \vee P(z,a)$
   - $C_3 = P(f(z),z) \vee P(z,a)$

We look for a finite unsatisfiable set of instances of C_1, C_2, C_3.
Let's instantiate:

Example 5.2.25

Let $A = \exists y \forall z (P(z,y) \Leftrightarrow \neg \exists x (P(z,x) \wedge P(x,z)))$. We compute the clausal form of A.

1-4. The four steps of Skolemzation yield:
$(\neg P(z,a) \vee \neg P(z,x) \vee \neg P(x,z)) \ \wedge \ (P(z,f(z)) \wedge P(f(z),z) \ \vee \ P(z,a))$

5. The clausal form is the following set of clauses:
   - $C_1 = \neg P(z,a) \vee \neg P(z,x) \vee \neg P(x,z)$
   - $C_2 = P(z,f(z)) \vee P(z,a)$
   - $C_3 = P(f(z),z) \vee P(z,a)$

We look for a finite unsatisfiable set of instances of C_1, C_2, C_3.
Let's instantiate:

- C_1 with $x := a, z := a$, we get $C_1' = \neg P(a,a)$

Example 5.2.25

Let $A = \exists y \forall z (P(z,y) \Leftrightarrow \neg \exists x (P(z,x) \wedge P(x,z)))$. We compute the clausal form of A.

1-4. The four steps of Skolemzation yield:
$(\neg P(z,a) \vee \neg P(z,x) \vee \neg P(x,z)) \ \wedge \ (P(z,f(z)) \wedge P(f(z),z) \ \vee \ P(z,a))$

5. The clausal form is the following set of clauses:
   - $C_1 = \neg P(z,a) \vee \neg P(z,x) \vee \neg P(x,z)$
   - $C_2 = P(z,f(z)) \vee P(z,a)$
   - $C_3 = P(f(z),z) \vee P(z,a)$

We look for a finite unsatisfiable set of instances of C_1, C_2, C_3.
Let's instantiate:

- C_1 with $x := a, z := a$, we get $C_1' = \neg P(a,a)$
- C_2 with $z := a$, we get $C_2' = P(a,f(a)) \vee P(a,a)$

Example 5.2.25

Let $A = \exists y \forall z (P(z,y) \Leftrightarrow \neg\exists x (P(z,x) \wedge P(x,z)))$. We compute the clausal form of A.

1-4. The four steps of Skolemzation yield:
$(\neg P(z,a) \vee \neg P(z,x) \vee \neg P(x,z)) \;\; \wedge \;\; (P(z,f(z)) \wedge P(f(z),z) \;\vee\; P(z,a))$

5. The clausal form is the following set of clauses:
    - $C_1 = \neg P(z,a) \vee \neg P(z,x) \vee \neg P(x,z)$
    - $C_2 = P(z,f(z)) \vee P(z,a)$
    - $C_3 = P(f(z),z) \vee P(z,a)$

We look for a finite unsatisfiable set of instances of C_1, C_2, C_3.
Let's instantiate:

- C_1 with $x := a, z := a$, we get $C_1' = \neg P(a,a)$
- C_2 with $z := a$, we get $C_2' = P(a,f(a)) \vee P(a,a)$
- C_3 with $z := a$, we get $C_3' = P(f(a),a) \vee P(a,a)$

Example 5.2.25

Let $A = \exists y \forall z (P(z,y) \Leftrightarrow \neg \exists x (P(z,x) \wedge P(x,z)))$. We compute the clausal form of A.

1-4. The four steps of Skolemzation yield:
$(\neg P(z,a) \vee \neg P(z,x) \vee \neg P(x,z)) \;\wedge\; (P(z,f(z)) \wedge P(f(z),z) \;\vee\; P(z,a))$

5. The clausal form is the following set of clauses:
   - $C_1 = \neg P(z,a) \vee \neg P(z,x) \vee \neg P(x,z)$
   - $C_2 = P(z,f(z)) \vee P(z,a)$
   - $C_3 = P(f(z),z) \vee P(z,a)$

We look for a finite unsatisfiable set of instances of C_1, C_2, C_3.
Let's instantiate:

- C_1 with $x := a, z := a$, we get $C_1' = \neg P(a,a)$
- C_2 with $z := a$, we get $C_2' = P(a,f(a)) \vee P(a,a)$
- C_3 with $z := a$, we get $C_3' = P(f(a),a) \vee P(a,a)$
- C_1 with $x := a, z := f(a)$, we get $C_1'' = \neg P(f(a),a) \vee \neg P(a,f(a))$

Example 5.2.25

Let $A = \exists y \forall z (P(z,y) \Leftrightarrow \neg \exists x (P(z,x) \wedge P(x,z)))$. We compute the clausal form of A.

1-4. The four steps of Skolemzation yield:
$(\neg P(z,a) \vee \neg P(z,x) \vee \neg P(x,z)) \;\wedge\; (P(z,f(z)) \wedge P(f(z),z) \;\vee\; P(z,a))$

5. The clausal form is the following set of clauses:
   - $C_1 = \neg P(z,a) \vee \neg P(z,x) \vee \neg P(x,z)$
   - $C_2 = P(z,f(z)) \vee P(z,a)$
   - $C_3 = P(f(z),z) \vee P(z,a)$

We look for a finite unsatisfiable set of instances of C_1, C_2, C_3.
Let's instantiate:

- C_1 with $x := a, z := a$, we get $C_1' = \neg P(a,a)$
- C_2 with $z := a$, we get $C_2' = P(a,f(a)) \vee P(a,a)$
- C_3 with $z := a$, we get $C_3' = P(f(a),a) \vee P(a,a)$
- C_1 with $x := a, z := f(a)$, we get $C_1'' = \neg P(f(a),a) \vee \neg P(a,f(a))$

This set of instances is unsatisfiable, thus A is unsatisfiable !

In practice

Let Γ be a set of clauses. We want to prove that $\forall(\Gamma)$ has no model.

- How do we choose the instances?
- How do we prove their insatisfiability?

In practice

Let Γ be a set of clauses. We want to prove that $\forall(\Gamma)$ has no model.

- How do we choose the instances?
- How do we prove their insatisfiability?

We use a formal system of “factorization, copy, binary resolution” to infer $\perp$ from Γ.

In practice

Let Γ be a set of clauses. We want to prove that $\forall(\Gamma)$ has no model.

- How do we choose the instances?
- How do we prove their insatisfiability?

We use a formal system of “factorization, copy, binary resolution” to infer $\perp$ from Γ.

Completeness of these rules is based on Herbrand’s Theorem. Unification is used to find suitable instances of these clauses.

Plan

John Alan Robinson (1930-2016)

- developed the **resolution principle**
- **unification** algorithm (1965)
  - makes the search for contradictory instances efficient
  - special case of *matching* used in functional programming
- Founder of *logic programming*

John Alan Robinson (1930-2016)

- developed the **resolution principle**
- **unification** algorithm (1965)
  - makes the search for contradictory instances efficient
  - special case of *matching* used in functional programming
- Founder of *logic programming*

```
parent(pascal, mathilde).
brother(stephane, pascal).
uncle(X,Y) :- parent(Z,Y),
              brother(X,Z).
?- uncle(stephane, mathilde).
  true.
```

(Prolog, Colmerauer & Roussel, 1972)

Unification: expression, solution

Definition 5.3.1

- A term or a litteral is an **expression**.
- A substitution σ is a **solution** of **equation** $e_1 = e_2$ if $e_1\sigma$ and $e_2\sigma$ are syntactically identical.
- A substitution is a solution of a set of equations if it is a solution of each equation in the set.

Unification: example 5.3.4

The equation $P(x, f(y)) = P(g(z), z)$ has the solution :

The set of equations $x = g(z),\ \ f(y) = z$ has the solution :

Unification: example 5.3.4

The equation $P(x, f(y)) = P(g(z), z)$ has the solution :

$\boxed{x := g(f(y)), z := f(y)}$

The set of equations $x = g(z), \;\; f(y) = z$ has the solution :

Unification: example 5.3.4

The equation $P(x, f(y)) = P(g(z), z)$ has the solution :

$\boxed{x := g(f(y)), z := f(y)}$

The set of equations $x = g(z),\ \ f(y) = z$ has the solution :

$\boxed{x := g(f(y)), z := f(y)}$

Unification: composition of substitutions

Definition 5.3.5

- Let σ and τ be two substitutions, we note $\sigma\tau$ the substitution such that for all variable x, $x\sigma\tau = (x\sigma)\tau$.
- The substitution $\sigma\tau$ is an instance of σ.
- Two substitutions are equivalent if each of them is an instance of the other.

Unification: example 5.3.6

Consider substitutions

- $\sigma_1 =< x := g(z), y := z >$
- $\sigma_2 =< x := g(y), z := y >$
- $\sigma_3 =< x := g(a), y := a, z := a >$

We have the following relations between these substitutions:

Unification: example 5.3.6

Consider substitutions

- $\sigma_1 =< x := g(z), y := z >$
- $\sigma_2 =< x := g(y), z := y >$
- $\sigma_3 =< x := g(a), y := a, z := a >$

We have the following relations between these substitutions:

- $\sigma_1 = \sigma_2 < y := z >$
- $\sigma_2 = \sigma_1 < z := y >$

σ_1 and σ_2 are equivalent.

Unification: example 5.3.6

Consider substitutions

- $\sigma_1 =< x := g(z), y := z >$
- $\sigma_2 =< x := g(y), z := y >$
- $\sigma_3 =< x := g(a), y := a, z := a >$

We have the following relations between these substitutions:

- $\sigma_1 = \sigma_2 < y := z >$
- $\sigma_2 = \sigma_1 < z := y >$

σ_1 and σ_2 are equivalent.

- $\sigma_3 = \sigma_1 < z := a >$
- $\sigma_3 = \sigma_2 < y := a >$

σ_3 is an instance of σ_1 as well as of σ_2, but is equivalent to neither of them.

Unification: definition of the most general solution

Definition 5.3.7 (mgu)

A solution of a set of equations is said to be the most general if any other solution is an instance of it.
Note that two "most general" solutions are equivalent.

Unification: definition of the most general solution

Definition 5.3.7 (mgu)

A solution of a set of equations is said to be the most general if any other solution is an instance of it.
Note that two "most general" solutions are equivalent.

Example 5.3.8

Consider the equation $f(x,g(z)) = f(g(y),x)$.

- $\sigma_1 =< x := g(z), y := z >$
- $\sigma_2 =< x := g(y), z := y >$
- $\sigma_3 =< x := g(a), y := a, z := a >$

are 3 solutions.

σ_1 and σ_2 are its most general solutions.

Unifier

Definition 5.3.2

Let E be a set of expressions and $E\sigma = \{t\sigma \mid t \in E\}$.
σ is a unifier of E if and only if the set $E\sigma$ has only one element.

If $E = \{e_1, \ldots e_n\}$, another way of writing this is that

σ is a solution of the set of equations $\begin{cases} e_1 = e_2 \\ \ldots \\ e_{n-1} = e_n \end{cases}$

Unifier

Definition 5.3.2

Let E be a set of expressions and $E\sigma = \{t\sigma \mid t \in E\}$.
σ is a unifier of E if and only if the set $E\sigma$ has only one element.

If $E = \{e_1, \dots e_n\}$, another way of writing this is that

σ is a solution of the set of equations $\begin{cases} e_1 = e_2 \\ \dots \\ e_{n-1} = e_n \end{cases}$

The notion of most general unifier (or principal unifier) extends to this definition.

Unification algorithm: a sketch

The algorithm separates equations into:

- equations to be solved, denoted by $=$
- solved equations, denoted by $:=$

Unification algorithm: a sketch

The algorithm separates equations into:

- equations to be solved, denoted by $=$
- solved equations, denoted by $:=$

Initially, there is no solved equations.

The algorithm transforms a system into an equivalent system and stops when :

- every equation is solved:
  then the list of solved equations is the most general solution
- or when it claims that there is no solution.

Unification algorithm: the rules

Choose an equation yet to be solved then:

1. **Remove the equation** if its 2 sides are identical.

Unification algorithm: the rules

Choose an equation yet to be solved then:

1. **Remove the equation** if its 2 sides are identical.
2. **Decompose**
   - $\neg A = \neg B$ becomes $A = B$
   - $f(s_1, \ldots, s_n) = f(t_1, \ldots, t_n)$ becomes $s_1 = t_1, \ldots, s_n = t_n$.
     (nothing if f is a constant)

Unification algorithm: the rules

Choose an equation yet to be solved then:

1. **Remove the equation** if its 2 sides are identical.
2. **Decompose**
   - $\neg A = \neg B$ becomes $A = B$
   - $f(s_1, \ldots, s_n) = f(t_1, \ldots, t_n)$ becomes $s_1 = t_1, \ldots, s_n = t_n$.
     (nothing if f is a constant)
3. **Failure of decomposition**
   If an equation is of the form $f(s_1, \ldots, s_n) = g(t_1, \ldots, t_p)$ with $f \neq g$
   then the algorithm claims that there is no solution.
   (in particular is the equation is $\neg A = B$ with B a positive litteral)

Unification: algorithm (rules)

4. **Orient**
   If an equation is $t = x$ where t is a (true) term and x is a variable, then we replace the equation with $x = t$.

Unification: algorithm (rules)

4. **Orient**
   If an equation is $t = x$ where t is a (true) term and x is a variable, then we replace the equation with $x = t$.
5. **Elimination of a variable**
   If an equation is $x = t$ where x is a variable and t is a term without any occurrence of x
   - remove it from the equations to be solved
   - replace x by t in every equation (unsolved and solved)
   - add $x := t$ to the solved equations

Unification: algorithm (rules)

4. **Orient**
   If an equation is $t = x$ where t is a (true) term and x is a variable, then we replace the equation with $x = t$.
5. **Elimination of a variable**
   If an equation is $x = t$ where x is a variable and t is a term without any occurrence of x
   - remove it from the equations to be solved
   - replace x by t in every equation (unsolved and solved)
   - add $x := t$ to the solved equations
6. **Failure of elimination**
   If an equation is $x = t$ where x is a variable and t contains x then the algorithm claims that there is no solution.

Unification: algorithm (example 5.3.11)

1. Solve $f(x, g(z)) = f(g(y), x)$.

Unification: algorithm (example 5.3.11)

1. Solve $f(x, g(z)) = f(g(y), x)$.

   Decomposition $\quad x = g(y), g(z) = x$

Unification: algorithm (example 5.3.11)

1. Solve $f(x, g(z)) = f(g(y), x)$.

Decomposition	$x = g(y), g(z) = x$
Elimination of x	$x := g(y), g(z) = g(y)$

Unification: algorithm (example 5.3.11)

1. Solve $f(x, g(z)) = f(g(y), x)$.

Decomposition	$x = g(y), g(z) = x$
Elimination of x	$x := g(y), g(z) = g(y)$
Decomposition	$x := g(y), z = y$

Unification: algorithm (example 5.3.11)

1. Solve $f(x, g(z)) = f(g(y), x)$.

Decomposition	$x = g(y), g(z) = x$
Elimination of x	$x := g(y), g(z) = g(y)$
Decomposition	$x := g(y), z = y$
Elimination of z	$x := g(y), z := y$ solution

Unification: algorithm (example 5.3.11)

1. Solve $f(x, g(z)) = f(g(y), x)$.

Decomposition	$x = g(y), g(z) = x$
Elimination of x	$x := g(y), g(z) = g(y)$
Decomposition	$x := g(y), z = y$
Elimination of z	$x := g(y), z := y$ solution

2. Solve $f(x, x, a) = f(g(y), g(a), y)$.

Unification: algorithm (example 5.3.11)

1. Solve $f(x, g(z)) = f(g(y), x)$.

Decomposition	$x = g(y), g(z) = x$
Elimination of x	$x := g(y), g(z) = g(y)$
Decomposition	$x := g(y), z = y$
Elimination of z	$x := g(y), z := y$ solution

2. Solve $f(x, x, a) = f(g(y), g(a), y)$.

Decomposition $x = g(y), x = g(a), a = y$

Unification: algorithm (example 5.3.11)

1. Solve $f(x, g(z)) = f(g(y), x)$.

Decomposition	$x = g(y), g(z) = x$
Elimination of x	$x := g(y), g(z) = g(y)$
Decomposition	$x := g(y), z = y$
Elimination of z	$x := g(y), z := y$ solution

2. Solve $f(x, x, a) = f(g(y), g(a), y)$.

Decomposition	$x = g(y), x = g(a), a = y$
Elimination of x in the 1st equation	$x := g(y), g(y) = g(a), a = y$

Unification: algorithm (example 5.3.11)

1. Solve $f(x, g(z)) = f(g(y), x)$.

Decomposition	$x = g(y), g(z) = x$
Elimination of x	$x := g(y), g(z) = g(y)$
Decomposition	$x := g(y), z = y$
Elimination of z	$x := g(y), z := y$ solution

2. Solve $f(x, x, a) = f(g(y), g(a), y)$.

Decomposition	$x = g(y), x = g(a), a = y$
Elimination of x in the 1st equation	$x := g(y), g(y) = g(a), a = y$
Decomposition	$x := g(y), y = a, a = y$

Unification: algorithm (example 5.3.11)

1. Solve $f(x, g(z)) = f(g(y), x)$.

Decomposition	$x = g(y), g(z) = x$
Elimination of x	$x := g(y), g(z) = g(y)$
Decomposition	$x := g(y), z = y$
Elimination of z	$x := g(y), z := y$ solution

2. Solve $f(x, x, a) = f(g(y), g(a), y)$.

Decomposition	$x = g(y), x = g(a), a = y$
Elimination of x in the 1st equation	$x := g(y), g(y) = g(a), a = y$
Decomposition	$x := g(y), y = a, a = y$
Elimination of y	$x := g(a), y := a, a = a$

Unification: algorithm (example 5.3.11)

1. Solve $f(x, g(z)) = f(g(y), x)$.

Decomposition	$x = g(y), g(z) = x$
Elimination of x	$x := g(y), g(z) = g(y)$
Decomposition	$x := g(y), z = y$
Elimination of z	$x := g(y), z := y$ solution

2. Solve $f(x, x, a) = f(g(y), g(a), y)$.

Decomposition	$x = g(y), x = g(a), a = y$
Elimination of x in the 1st equation	$x := g(y), g(y) = g(a), a = y$
Decomposition	$x := g(y), y = a, a = y$
Elimination of y	$x := g(a), y := a, a = a$
Removal	$x := g(a), y := a$ solution

Unification: algorithm (example 5.3.11)

3. Solve $f(x,x,x) = f(g(y),g(a),y)$.

   Decomposition $\quad x = g(y), x = g(a), x = y$

Unification: algorithm (example 5.3.11)

3. Solve $f(x,x,x) = f(g(y),g(a),y)$.

Decomposition	$x = g(y), x = g(a), x = y$
Elimination of x	$x := g(y), g(y) = g(a), g(y) = y$

Unification: algorithm (example 5.3.11)

3. Solve $f(x,x,x) = f(g(y),g(a),y)$.

Decomposition	$x = g(y), x = g(a), x = y$
Elimination of x	$x := g(y), g(y) = g(a), g(y) = y$
Orienting	$x := g(y), g(y) = g(a), y = g(y)$

Unification: algorithm (example 5.3.11)

3. Solve $f(x,x,x) = f(g(y),g(a),y)$.

Decomposition	$x = g(y), x = g(a), x = y$
Elimination of x	$x := g(y), g(y) = g(a), g(y) = y$
Orienting	$x := g(y), g(y) = g(a), y = g(y)$
Elimination failure	there is no solution

Unification: algorithm (example 5.3.11)

3. Solve $f(x,x,x) = f(g(y),g(a),y)$.

Decomposition	$x = g(y), x = g(a), x = y$
Elimination of x	$x := g(y), g(y) = g(a), g(y) = y$
Orienting	$x := g(y), g(y) = g(a), y = g(y)$
Elimination failure	there is no solution

Remark: correctness and termination proofs for unification algorithm are in the handout course notes.

Plan

Three rules (examples)

1. **Factorization**

$$\frac{P(x,x) \vee P(y,a) \vee Q(y)}{P(a,a) \vee Q(a)}$$

2. **Copy**

$$\frac{P(x,y)}{P(u,v)}$$

3. **Binary resolution**

$$\frac{Q(x) \vee P(x,a) \qquad \neg P(b,y) \vee R(f(y))}{Q(b) \vee R(f(a))}$$

Three rules (examples)

1. **Factorization**

$$\frac{P(x,x) \vee P(y,a) \vee Q(y)}{P(a,a) \vee Q(a)} \qquad \text{unification}$$

2. **Copy**

$$\frac{P(x,y)}{P(u,v)}$$

3. **Binary resolution**

$$\frac{Q(x) \vee P(x,a) \qquad \neg P(b,y) \vee R(f(y))}{Q(b) \vee R(f(a))} \qquad \text{unification}$$

Factorization

Definition 5.4.2

The clause C' is a factor of clause C if:

- either $C' = C$
- or $C' = C\sigma$
  where σ is the most general unifier of at least two literals in C.

Example 5.4.3

The clause $\underline{P(x)} \vee Q(g(x,y)) \vee \underline{P(f(a))}$ has two factors :

Factorization

Definition 5.4.2

The clause C' is a factor of clause C if:

- either $C' = C$
- or $C' = C\sigma$
  where σ is the most general unifier of at least two literals in C.

Example 5.4.3

The clause $\underline{P(x)} \vee Q(g(x,y)) \vee \underline{P(f(a))}$ has two factors :

- itself
- $P(f(a)) \vee Q(g(f(a),y))$ obtained by applying $x := f(a)$

Factorization

Definition 5.4.2

The clause C' is a factor of clause C if:

- either $C' = C$
- or $C' = C\sigma$
  where σ is the most general unifier of at least two literals in C.

Example 5.4.3

The clause $\underline{P(x)} \vee Q(g(x,y)) \vee \underline{P(f(a))}$ has two factors :

- itself
- $P(f(a)) \vee Q(g(f(a),y))$ obtained by applying $x := f(a)$

Property 5.4.4

Let C' be a factor of C: then $\forall(C) \models \forall(C')$.

Proof: Actually $\forall(A) \models \forall(A\sigma)$ for any formula A and any substitution σ.

Copy

Definition 5.4.5

Let σ be a substitution which:

- changes only variables into variables
- is a bijection

The clause $C\sigma$ is a copy of the clause C.

We also say that σ is a renaming of C.

Copy

Definition 5.4.5

Let σ be a substitution which:

- changes only variables into variables
- is a bijection

The clause $C\sigma$ is a copy of the clause C.

We also say that σ is a renaming of C.

Example 5.4.7

Let $\sigma = < x := u, y := v >$.
The litteral $P(u, v)$ is a copy of $P(x, y)$.

Note that $P(x, y)$ is also a copy of $P(u, v)$
by the renaming $\tau = < u := x, v := y >$, the inverse of the renaming σ.

Copy

Property 5.4.8

If σ is a renaming of C, then C is also a copy of $C\sigma$.

Proof.

It is easy to prove that σ^{-1} is a renaming of $C\sigma$. □

Property 5.4.9

If C and C' are copies of each other, then $\forall(C) \equiv \forall(C')$.

Proof.

C and C' are instances of each other.
Thus $\forall(C) \equiv \forall(C')$ and conversely. □

Binary resolvent

Definition 5.4.10

Let C and D be two clauses without common variables.
If there are two litterals:

- $L \in C$
- $M \in D$
- such that L and M^c are unifiable
- σ is the most general solution of the equation $L = M^c$

then $E = ((C - \{L\}) \cup (D - \{M\}))\sigma$ is a binary resolvent of C and D.

Binary resolvent

Example 5.4.11

Let $C = P(x, y) \vee P(y, k(z))$ and $D = \neg P(a, f(a, y_1))$.

Binary resolvent

Example 5.4.11

Let $C = P(x, y) \vee P(y, k(z))$ and $D = \neg P(a, f(a, y_1))$.

$< x := a, y := f(a, y_1) >$ is the most general solution of
$P(x, y) = P(a, f(a, y_1))$
The (only) binary resolvent is $P(f(a, y_1), k(z))$.

Binary resolvent

Example 5.4.11

Let $C = P(x, y) \vee P(y, k(z))$ and $D = \neg P(a, f(a, y_1))$.

$< x := a, y := f(a, y_1) >$ is the most general solution of
$P(x, y) = P(a, f(a, y_1))$
The (only) binary resolvent is $P(f(a, y_1), k(z))$.

Property 5.4.12

Let E be a resolvent binary of clauses C and D : $\forall(C), \forall(D) \models \forall(E)$.

Resolution:

Definition 5.4.13

A proof of C from Γ is a sequence of clauses where each clause is:

- a member of Γ,
- or a factor of a previous clause in the proof,
- or a copy of a previous clause in the proof,
- or a binary resolvent of 2 previous clauses in the proof.

terminated by C.

C is first-order inferred from Γ, denoted by $\Gamma \vdash_{1fcb} C$.

Resolution:

Definition 5.4.13

A proof of C from Γ is a sequence of clauses where each clause is:

- a member of Γ,
- or a factor of a previous clause in the proof,
- or a copy of a previous clause in the proof,
- or a binary resolvent of 2 previous clauses in the proof.

terminated by C.

C is first-order inferred from Γ, denoted by $\Gamma \vdash_{1fcb} C$.

Property 5.4.14: consistency

If $\Gamma \vdash_{1fcb} C$ then $\forall(\Gamma) \models \forall(C)$

By induction, using the consistency of the three rules.

Resolution: Example 5.4.15

Given the two clauses

1. $C_1 = P(x, y) \vee P(y, x)$
2. $C_2 = \neg P(u, z) \vee \neg P(z, u)$

Show by resolution that $\forall(C_1, C_2)$ has no model.

Resolution: Example 5.4.15

Given the two clauses

1. $C_1 = P(x, y) \vee P(y, x)$
2. $C_2 = \neg P(u, z) \vee \neg P(z, u)$

Show by resolution that $\forall(C_1, C_2)$ has no model.

1.	$P(x, y) \vee P(y, x)$	Hyp C_1

Resolution: Example 5.4.15

Given the two clauses

1. $C_1 = P(x, y) \vee P(y, x)$
2. $C_2 = \neg P(u, z) \vee \neg P(z, u)$

Show by resolution that $\forall(C_1, C_2)$ has no model.

1.	$P(x, y) \vee P(y, x)$	Hyp C_1
2.	$P(y, y)$	Factor of 1 $< x := y >$

Resolution: Example 5.4.15

Given the two clauses

1. $C_1 = P(x,y) \vee P(y,x)$
2. $C_2 = \neg P(u,z) \vee \neg P(z,u)$

Show by resolution that $\forall(C_1, C_2)$ has no model.

1.	$P(x,y) \vee P(y,x)$	Hyp C_1	
2.	$P(y,y)$	Factor of 1	$< x := y >$
3.	$\neg P(u,z) \vee \neg P(z,u)$	Hyp C_2	

Resolution: Example 5.4.15

Given the two clauses

1. $C_1 = P(x, y) \lor P(y, x)$
2. $C_2 = \neg P(u, z) \lor \neg P(z, u)$

Show by resolution that $\forall(C_1, C_2)$ has no model.

1.	$P(x, y) \lor P(y, x)$	Hyp C_1	
2.	$P(y, y)$	Factor of 1	$< x := y >$
3.	$\neg P(u, z) \lor \neg P(z, u)$	Hyp C_2	
4.	$\neg P(z, z)$	Factor of 3	$< u := z >$

Resolution: Example 5.4.15

Given the two clauses

1. $C_1 = P(x, y) \vee P(y, x)$
2. $C_2 = \neg P(u, z) \vee \neg P(z, u)$

Show by resolution that $\forall(C_1, C_2)$ has no model.

1.	$P(x, y) \vee P(y, x)$	Hyp C_1
2.	$P(y, y)$	Factor of 1 $< x := y >$
3.	$\neg P(u, z) \vee \neg P(z, u)$	Hyp C_2
4.	$\neg P(z, z)$	Factor of 3 $< u := z >$
5.	$\perp$	Binary Resolvent 2, 4 $< y := z >$

Resolution: Example 5.4.15

Given the two clauses

1. $C_1 = P(x, y) \vee P(y, x)$
2. $C_2 = \neg P(u, z) \vee \neg P(z, u)$

Show by resolution that $\forall(C_1, C_2)$ has no model.

1.	$P(x, y) \vee P(y, x)$	Hyp C_1
2.	$P(y, y)$	Factor of 1 $< x := y >$
3.	$\neg P(u, z) \vee \neg P(z, u)$	Hyp C_2
4.	$\neg P(z, z)$	Factor of 3 $< u := z >$
5.	$\perp$	Binary Resolvent 2, 4 $< y := z >$

This example shows, a contrario, that binary resolution alone is incomplete: without factorization, the empty clause cannot be inferred.

Resolution: Example 5.4.16

1. $C_1 = \neg P(z, a) \vee \neg P(z, x) \vee \neg P(x, z)$
2. $C_2 = P(z, f(z)) \vee P(z, a)$
3. $C_3 = P(f(z), z) \vee P(z, a)$

Resolution: Example 5.4.16

1. $C_1 = \neg P(z,a) \vee \neg P(z,x) \vee \neg P(x,z)$
2. $C_2 = P(z,f(z)) \vee P(z,a)$
3. $C_3 = P(f(z),z) \vee P(z,a)$

1.	$\neg P(z,a) \vee \neg P(z,x) \vee \neg P(x,z)$	Hyp C_1

Resolution: Example 5.4.16

1. $C_1 = \neg P(z,a) \vee \neg P(z,x) \vee \neg P(x,z)$
2. $C_2 = P(z,f(z)) \vee P(z,a)$
3. $C_3 = P(f(z),z) \vee P(z,a)$

1.	$\neg P(z,a) \vee \neg P(z,x) \vee \neg P(x,z)$	Hyp C_1
2.	$P(z,f(z)) \vee P(z,a)$	Hyp C_2

Resolution: Example 5.4.16

1. $C_1 = \neg P(z,a) \vee \neg P(z,x) \vee \neg P(x,z)$
2. $C_2 = P(z,f(z)) \vee P(z,a)$
3. $C_3 = P(f(z),z) \vee P(z,a)$

1.	$\neg P(z,a) \vee \neg P(z,x) \vee \neg P(x,z)$	Hyp C_1
2.	$P(z,f(z)) \vee P(z,a)$	Hyp C_2
3.	$P(v,f(v)) \vee P(v,a)$	Copy 2 $< z := v >$

Resolution: Example 5.4.16

1. $C_1 = \neg P(z,a) \vee \neg P(z,x) \vee \neg P(x,z)$
2. $C_2 = P(z,f(z)) \vee P(z,a)$
3. $C_3 = P(f(z),z) \vee P(z,a)$

1.	$\neg P(z,a) \vee \neg P(z,x) \vee \neg P(x,z)$	Hyp C_1
2.	$P(z,f(z)) \vee P(z,a)$	Hyp C_2
3.	$P(v,f(v)) \vee P(v,a)$	Copy 2 $< z := v >$
4.	$\neg P(f(v),a) \vee \neg P(f(v),v) \vee P(v,a)$	BR 1(3), 3(1) $<z{:=}f(v)\ ;\ x{:=}v>$

Resolution: Example 5.4.16

1. $C_1 = \neg P(z,a) \vee \neg P(z,x) \vee \neg P(x,z)$
2. $C_2 = P(z,f(z)) \vee P(z,a)$
3. $C_3 = P(f(z),z) \vee P(z,a)$

1.	$\neg P(z,a) \vee \neg P(z,x) \vee \neg P(x,z)$	Hyp C_1
2.	$P(z,f(z)) \vee P(z,a)$	Hyp C_2
3.	$P(v,f(v)) \vee P(v,a)$	Copy 2 $< z := v >$
4.	$\neg P(f(v),a) \vee \neg P(f(v),v) \vee P(v,a)$	BR 1(3), 3(1) $<z{:=}f(v)\ ;\ x{:=}v>$
5.	$\neg P(f(a),a) \vee P(a,a)$	Fact 4 $< v := a >$

Resolution: Example 5.4.16

1. $C_1 = \neg P(z,a) \vee \neg P(z,x) \vee \neg P(x,z)$
2. $C_2 = P(z,f(z)) \vee P(z,a)$
3. $C_3 = P(f(z),z) \vee P(z,a)$

1.	$\neg P(z,a) \vee \neg P(z,x) \vee \neg P(x,z)$	Hyp C_1
2.	$P(z,f(z)) \vee P(z,a)$	Hyp C_2
3.	$P(v,f(v)) \vee P(v,a)$	Copy 2 $< z := v >$
4.	$\neg P(f(v),a) \vee \neg P(f(v),v) \vee P(v,a)$	BR 1(3), 3(1) $<z{:=}f(v)\ ;\ x{:=}v >$
5.	$\neg P(f(a),a) \vee P(a,a)$	Fact 4 $< v := a >$
6.	$P(f(z),z) \vee P(z,a)$	Hyp C_3

Resolution: Example 5.4.16

1. $C_1 = \neg P(z,a) \vee \neg P(z,x) \vee \neg P(x,z)$
2. $C_2 = P(z,f(z)) \vee P(z,a)$
3. $C_3 = P(f(z),z) \vee P(z,a)$

1.	$\neg P(z,a) \vee \neg P(z,x) \vee \neg P(x,z)$	Hyp C_1
2.	$P(z,f(z)) \vee P(z,a)$	Hyp C_2
3.	$P(v,f(v)) \vee P(v,a)$	Copy 2 $< z := v >$
4.	$\neg P(f(v),a) \vee \neg P(f(v),v) \vee P(v,a)$	BR 1(3), 3(1) $<z{:=}f(v)\ ;\ x{:=}v >$
5.	$\color{red}{\neg P(f(a),a)} \vee P(a,a)$	Fact 4 $< v := a >$
6.	$\color{red}{P(f(z),z)} \vee P(z,a)$	Hyp C_3
7.	$P(a,a)$	BR 5(1), 6(1) $< z := a >$

Resolution: Example 5.4.16

1. $C_1 = \neg P(z,a) \vee \neg P(z,x) \vee \neg P(x,z)$
2. $C_2 = P(z,f(z)) \vee P(z,a)$
3. $C_3 = P(f(z),z) \vee P(z,a)$

1.	$\neg P(z,a) \vee \neg P(z,x) \vee \neg P(x,z)$	Hyp C_1
2.	$P(z,f(z)) \vee P(z,a)$	Hyp C_2
3.	$P(v,f(v)) \vee P(v,a)$	Copy 2 $< z := v >$
4.	$\neg P(f(v),a) \vee \neg P(f(v),v) \vee P(v,a)$	BR 1(3), 3(1) $<z{:=}f(v)\ ;\ x{:=}v>$
5.	$\neg P(f(a),a) \vee P(a,a)$	Fact 4 $< v := a >$
6.	$P(f(z),z) \vee P(z,a)$	Hyp C_3
7.	$P(a,a)$	BR 5(1), 6(1) $< z := a >$
8.	$\neg P(a,a)$	Fact 1 $< x := a; z := a >$

Resolution: Example 5.4.16

1. $C_1 = \neg P(z,a) \vee \neg P(z,x) \vee \neg P(x,z)$
2. $C_2 = P(z,f(z)) \vee P(z,a)$
3. $C_3 = P(f(z),z) \vee P(z,a)$

1.	$\neg P(z,a) \vee \neg P(z,x) \vee \neg P(x,z)$	Hyp C_1
2.	$P(z,f(z)) \vee P(z,a)$	Hyp C_2
3.	$P(v,f(v)) \vee P(v,a)$	Copy 2 $< z := v >$
4.	$\neg P(f(v),a) \vee \neg P(f(v),v) \vee P(v,a)$	BR 1(3), 3(1) $<z{:=}f(v)\ ;\ x{:=}v>$
5.	$\neg P(f(a),a) \vee P(a,a)$	Fact 4 $< v := a >$
6.	$P(f(z),z) \vee P(z,a)$	Hyp C_3
7.	$P(a,a)$	BR 5(1), 6(1) $< z := a >$
8.	$\neg P(a,a)$	Fact 1 $< x := a; z := a >$
9.	$\perp$	BR 7, 8

Plan

First-Order resolution

We define a new system with only one rule, first-order resolution, which is a combination of factorization, copy and binary resolution.

Definition 5.4.17

The clause E is a first-order resolvent of clauses C and D if:

- E is a binary resolvent of C' and D', where
- C' is a factor of C
- D' is a copy of a factor of D without any common variable with C'

$$\frac{C \qquad D}{E}$$ is called first-order resolution.

Example 5.4.18

$$\begin{array}{rl} \text{Let} & C = \neg P(z,a) \vee \neg P(z,x) \vee \neg P(x,z) \\ \text{and} & D = P(z,f(z)) \vee P(z,a). \end{array}$$

Example 5.4.18

Let $C = \neg P(z, a) \vee \neg P(z, x) \vee \neg P(x, z)$
and $D = P(z, f(z)) \vee P(z, a)$.

- $C' = \neg P(a, a)$ is a factor of C
- D is a factor of itself (without any common variable with C')
- $P(a, f(a))$ is a binary resolvent of C' and of D

Thus it is a first-order resolvent of C and D.

Three notions of proof by resolution

Let Γ be a set of clauses and C a clause.

Notations

Three notions of proof by resolution

Let Γ be a set of clauses and C a clause.

Notations

1. $\Gamma \vdash_p C$: proof of C from Γ by propositional resolution (without substitution).

Three notions of proof by resolution

Let Γ be a set of clauses and C a clause.

Notations

1. $\Gamma \vdash_p C$: proof of C from Γ by propositional resolution (without substitution).
2. $\Gamma \vdash_{1r} C$: proof of C from Γ obtained by first-order resolution.

Three notions of proof by resolution

Let Γ be a set of clauses and C a clause.

Notations

1. $\Gamma \vdash_p C$: proof of C from Γ by propositional resolution (without substitution).
2. $\Gamma \vdash_{1r} C$: proof of C from Γ obtained by first-order resolution.
3. $\Gamma \vdash_{1fcb} C$: proof of C from Γ by factorization, copy and binary resolution.

Three notions of proof by resolution

Let Γ be a set of clauses and C a clause.

Notations

1. $\Gamma \vdash_p C$: proof of C from Γ by propositional resolution (without substitution).
2. $\Gamma \vdash_{1r} C$: proof of C from Γ obtained by first-order resolution.
3. $\Gamma \vdash_{1fcb} C$: proof of C from Γ by factorization, copy and binary resolution.

By definition we have : $\Gamma \vdash_{1r} C$ implies $\Gamma \vdash_{1fcb} C$

Lifting theorem (1/3)

Theorem 5.4.19

Let C' and D' be instances of C and D.
Let E' be a propositional resolvent of C' and D'.

Then E' is an instance of a first-order resolvent E of C and D.

Lifting theorem (1/3)

Theorem 5.4.19

Let C' and D' be instances of C and D.
Let E' be a propositional resolvent of C' and D'.

Then E' is an instance of a first-order resolvent E of C and D.

Example 5.4.20

Let $C = P(x) \vee P(y) \vee R(y)$ and $D = \neg Q(x) \vee P(x) \vee \neg R(x) \vee P(y)$.

Lifting theorem (1/3)

Theorem 5.4.19

Let C' and D' be instances of C and D.
Let E' be a propositional resolvent of C' and D'.

Then E' is an instance of a first-order resolvent E of C and D.

Example 5.4.20

Let $C = P(x) \vee P(y) \vee R(y)$ and $D = \neg Q(x) \vee P(x) \vee \neg R(x) \vee P(y)$.

- $C' = P(a) \vee R(a)$ and $D' = \neg Q(a) \vee P(a) \vee \neg R(a)$ are instances of C and D.

Lifting theorem (1/3)

Theorem 5.4.19

Let C' and D' be instances of C and D.
Let E' be a propositional resolvent of C' and D'.

Then E' is an instance of a first-order resolvent E of C and D.

Example 5.4.20

Let $C = P(x) \vee P(y) \vee R(y)$ and $D = \neg Q(x) \vee P(x) \vee \neg R(x) \vee P(y)$.

- $C' = P(a) \vee R(a)$ and $D' = \neg Q(a) \vee P(a) \vee \neg R(a)$ are instances of C and D.
- $E' = P(a) \vee \neg Q(a)$ is a propositional resolvent of C' and D'.

Lifting theorem (1/3)

Theorem 5.4.19

Let C' and D' be instances of C and D.
Let E' be a propositional resolvent of C' and D'.

Then E' is an instance of a first-order resolvent E of C and D.

Example 5.4.20

Let $C = P(x) \vee P(y) \vee R(y)$ and $D = \neg Q(x) \vee P(x) \vee \neg R(x) \vee P(y)$.

- $C' = P(a) \vee R(a)$ and $D' = \neg Q(a) \vee P(a) \vee \neg R(a)$ are instances of C and D.
- $E' = P(a) \vee \neg Q(a)$ is a propositional resolvent of C' and D'.
- $E = P(x) \vee \neg Q(x)$ is a first-order resolvent of C and D having E' as an instance.

Lifting theorem (2/3)

Theorem 5.4.21

Let Δ be a set of instances of clauses from Γ.
Let $C_1, \ldots, C_n$ be a proof by propositional resolution from Δ.

There exists a proof $D_1, \ldots, D_n$ by first-order resolution from Γ such that each C_i is an instance of D_i.

Lifting theorem (2/3)

Theorem 5.4.21

Let Δ be a set of instances of clauses from Γ.
Let $C_1, \ldots, C_n$ be a proof by propositional resolution from Δ.

There exists a proof $D_1, \ldots, D_n$ by first-order resolution from Γ such that each C_i is an instance of D_i.

Proof.

By induction on n. Let $C_1, \ldots, C_n, C_{n+1}$ be a proof by propositional resolution starting with Δ. By induction, there exists a proof $D_1, \ldots, D_n$ by first-order resolution.

Lifting theorem (2/3)

Theorem 5.4.21

Let Δ be a set of instances of clauses from Γ.
Let $C_1, \ldots, C_n$ be a proof by propositional resolution from Δ.

There exists a proof $D_1, \ldots, D_n$ by first-order resolution from Γ such that each C_i is an instance of D_i.

Proof.

By induction on n. Let $C_1, \ldots, C_n, C_{n+1}$ be a proof by propositional resolution starting with Δ. By induction, there exists a proof $D_1, \ldots, D_n$ by first-order resolution.

1. If $C_{n+1} \in \Delta$, then C_{n+1} is an instance of a clause in Γ: it is D_{n+1}.

Lifting theorem (2/3)

Theorem 5.4.21

Let Δ be a set of instances of clauses from Γ.
Let $C_1, \ldots, C_n$ be a proof by propositional resolution from Δ.

There exists a proof $D_1, \ldots, D_n$ by first-order resolution from Γ such that each C_i is an instance of D_i.

Proof.

By induction on n. Let $C_1, \ldots, C_n, C_{n+1}$ be a proof by propositional resolution starting with Δ. By induction, there exists a proof $D_1, \ldots, D_n$ by first-order resolution.

1. If $C_{n+1} \in \Delta$, then C_{n+1} is an instance of a clause in Γ: it is D_{n+1}.
2. If C_{n+1} is a propositional resolvent of C_j and C_k, we use the first-order resolvent of D_j and D_k from the previous theorem.

$\square$

Lifting theorem (3/3)

Corollary 5.4.22

Let Γ be a set of clauses
and Δ a set of instances of clauses of Γ.

Suppose that $\Delta \vdash_p C$.

There exists D such that:

- $\Gamma \vdash_{1r} D$
- C is an instance of D.

The proof of C from Δ has been lifted to a first-order proof.

Example 5.4.23

$\Gamma = \{ P(f(x)) \vee P(u), \neg P(x) \vee Q(z), \neg Q(x) \vee \neg Q(y) \}$.
$\forall(\Gamma)$ is unsatisfiable and we prove it in three different ways.

Example 5.4.23

$\Gamma = \{ P(f(x)) \vee P(u), \neg P(x) \vee Q(z), \neg Q(x) \vee \neg Q(y) \}$.
$\forall(\Gamma)$ is unsatisfiable and we prove it in three different ways.

1. By instanciation on the Herbrand universe $a, f(a), f(f(a)), \ldots$:

$P(f(x)) \vee P(u)$	is instanciated to	$P(f(a))$
$\neg P(x) \vee Q(z)$	is instanciated to	$\neg P(f(a)) \vee Q(a)$
$\neg Q(x) \vee \neg Q(y)$	is instanciated to	$\neg Q(a)$

These 3 instances together are unsatisfiable, as shown below by propositional resolution :

Example 5.4.23

$\Gamma = \{ P(f(x)) \vee P(u), \neg P(x) \vee Q(z), \neg Q(x) \vee \neg Q(y) \}$.
$\forall(\Gamma)$ is unsatisfiable and we prove it in three different ways.

1. By instanciation on the Herbrand universe $a, f(a), f(f(a)), \ldots$:

$P(f(x)) \vee P(u)$	is instanciated to	$P(f(a))$
$\neg P(x) \vee Q(z)$	is instanciated to	$\neg P(f(a)) \vee Q(a)$
$\neg Q(x) \vee \neg Q(y)$	is instanciated to	$\neg Q(a)$

These 3 instances together are unsatisfiable, as shown below by propositional resolution :

$$\dfrac{\dfrac{P(f(a)) \qquad \neg P(f(a)) \vee Q(a)}{Q(a)} \qquad \neg Q(a)}{\perp}$$

Example 5.4.23

$P(f(x)) \vee P(u),\ \neg P(x) \vee Q(z),\ \neg Q(x) \vee \neg Q(y)$

2. This proof by propositional resolution is lifted to a proof by first-order resolution :

$$\dfrac{\dfrac{P(f(x)) \vee P(u) \qquad \neg P(x) \vee Q(z)}{Q(z)} \qquad \neg Q(x) \vee \neg Q(y)}{\perp}$$

Example 5.4.23

$P(f(x)) \vee P(u),\ \neg P(x) \vee Q(z),\ \neg Q(x) \vee \neg Q(y)$

2. This proof by propositional resolution is lifted to a proof by first-order resolution :

$$\dfrac{\dfrac{P(f(x)) \vee P(u) \qquad \neg P(x) \vee Q(z)}{Q(z)} \qquad \neg Q(x) \vee \neg Q(y)}{\perp}$$

3. Each first-order resolution rule is decomposed into factorization, copy and binary resolution:

Example 5.4.23

$P(f(x)) \vee P(u),\ \neg P(x) \vee Q(z),\ \neg Q(x) \vee \neg Q(y)$

2. This proof by propositional resolution is lifted to a proof by first-order resolution :

$$\dfrac{\dfrac{P(f(x)) \vee P(u) \qquad \neg P(x) \vee Q(z)}{Q(z)} \qquad \neg Q(x) \vee \neg Q(y)}{\perp}$$

3. Each first-order resolution rule is decomposed into factorization, copy and binary resolution:

$$\dfrac{\dfrac{\dfrac{P(f(x)) \vee P(u)}{P(f(x))}\,\mathit{fact} \qquad \dfrac{\neg P(x) \vee Q(z)}{\neg P(y) \vee Q(z)}\,\mathit{copy}}{Q(z)}\,\mathit{br} \qquad \dfrac{\neg Q(x) \vee \neg Q(y)}{\neg Q(x)}\,\mathit{fact}}{\perp}\,\mathit{br}$$

Refutational completeness of first-order resolution

Theorem 5.4.24

The three propositions

1. $\Gamma \vdash_{1r} \bot$
2. $\Gamma \vdash_{1fcb} \bot$
3. $\forall(\Gamma) \models \bot$

are equivalent.

Refutational completeness of first-order resolution

Theorem 5.4.24

The three propositions

1. $\Gamma \vdash_{1r} \bot$
2. $\Gamma \vdash_{1fcb} \bot$
3. $\forall(\Gamma) \models \bot$

are equivalent.

Proof.

- (1 $\Rightarrow$ 2) because first-order resolution is a combinaison of factorization, copy and binary resolution.

□

Refutational completeness of first-order resolution

Theorem 5.4.24

The three propositions

1. $\Gamma \vdash_{1r} \bot$
2. $\Gamma \vdash_{1fcb} \bot$
3. $\forall(\Gamma) \models \bot$

are equivalent.

Proof.

- $(1 \Rightarrow 2)$ because first-order resolution is a combinaison of factorization, copy and binary resolution.
- $(2 \Rightarrow 3)$ because factorization, copy and binary resolution are consistent.

□

Refutational completeness of first-order resolution

Theorem 5.4.24

The three propositions

1. $\Gamma \vdash_{1r} \perp$
2. $\Gamma \vdash_{1fcb} \perp$
3. $\forall(\Gamma) \models \perp$

are equivalent.

Proof.

- (1 $\Rightarrow$ 2) because first-order resolution is a combinaison of factorization, copy and binary resolution.
- (2 $\Rightarrow$ 3) because factorization, copy and binary resolution are consistent.
- (3 $\Rightarrow$ 1). Suppose that $\forall(\Gamma)$ is unsatisfiable.
  By Herbrand's theorem, there is a finite unsatisfiable set Δ of instances.
  By completeness of propositional resolution, we have $\Delta \vdash_p \perp$.
  By lifting, $\Gamma \vdash_{1r} D$ where $\perp$ is an instance of D; hence $D = \perp$.

□

Automated proofs

To produce automated proofs in binary resolution, one can use the software (working similarly to *complete strategy*):

http://teachinglogic.univ-grenoble-alpes.fr/ResBinSc/

Automated proofs

To produce automated proofs in binary resolution, one can use the software (working similarly to *complete strategy*):

http://teachinglogic.univ-grenoble-alpes.fr/ResBinSc/

If the set of clauses is unsatisfiable, then the software can theoretically dedeuce the empty clause (given an unlimited amount of time).

Automated proofs

To produce automated proofs in binary resolution, one can use the software (working similarly to *complete strategy*):

http://teachinglogic.univ-grenoble-alpes.fr/ResBinSc/

If the set of clauses is unsatisfiable, then the software can theoretically dedeuce the empty clause (given an unlimited amount of time).

What can we conclude ?

- if the software states that it has deduced the empty clause:
  - the clauses are unsatisfiable indeed
  - it provides a proof
- if the software states that it cannot prove the empty clause, or if it runs out of time:
  - nothing can be concluded

Plan

Today

- Unification is an effective way of finding suitable instances of clauses with variables
- First-order resolution integrates in a single deductive system both the **search for unsatisfiable instances** and the **proof of unsatisfiability** of a set of clauses
- First-order resolution is **consistent and complete**, and one way to build a first-order proof is by lifting a propositional proof.

Overview of the Semester

- Propositional logic
- Propositional resolution
- Natural deduction for propositional logic

MIDTERM EXAM

- First order logic
- First-order resolution *
- First-order natural deduction

EXAM

Next lecture

First-order Natural Deduction

- Rules
- Examples