

Basis for automated proof: First-Order Resolution

Stéphane Devismes Pascal Lafourcade Michel Lévy
Jean-François Monin (jean-francois.monin@imag.fr)

Université Joseph Fourier, Grenoble I

March 20, 2015

Plan

Introduction

Clausal form

Unification

First-Order Resolution

Completeness

Idea

Skolemization yields formulae without quantifier.

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

- ▶ **clausal form** of skolemized formulae.
- ▶ **generalization of resolution**.
- ▶ **Correctness** and **completeness** of the method.

Litteral, clause

Definition 5.2.19

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

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

Every litteral is positive or negative.

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

Clausal form of a formula

Definition 5.2.20

Let A be a closed formula. **The clausal form of A , $F(A)$** is a set of clauses obtained from A in two steps :

1. Skolemize A into B
2. Replace B with an equivalent set Γ of clauses using distributivity of disjunction over conjunction.

Clausal form of a formula

property 1

5.2.21

- ▶ The universal closure of the clausal form of a closed formula A has a model if and only if A is a consequence of $\forall(F(A))$.
- ▶ If A has a model, then $\forall(F(A))$ has a model.

Proof

Proof.

Let A be a closed formula, B its Skolem form and Γ its clausal form. From the properties of skolemization :

- ▶ A is a consequence of $\forall(B)$.
- ▶ If A has a model then $\forall(B)$ has a model.

Since Γ is obtained using distributivity, B and Γ are equivalent, hence $\forall(B)$ and $\forall(\Gamma)$ are equivalent as well. Therefore, in the two properties above, $\forall(B)$ can be replaced with $\forall(\Gamma)$. \square

Clausal form of a set of formulae

Definition 5.2.22

Let Γ be a set of closed formulae. We define the **clausal form** of Γ as the union of clausal forms of all formulae of Γ , **paying attention, in the course of skolemization**, to use a **new** symbol for each eliminated existential quantifier.

Clausal form of a set of formulae

Corollary 5.2.23

Let Γ be a set of closed formulae and Δ the clausal form of Γ . We have :

- ▶ 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 the clausal form of Γ . Γ is unsatisfiable if and only if there exists a finite unsatisfiable subset of instances of clauses of Δ on the signature of Δ .

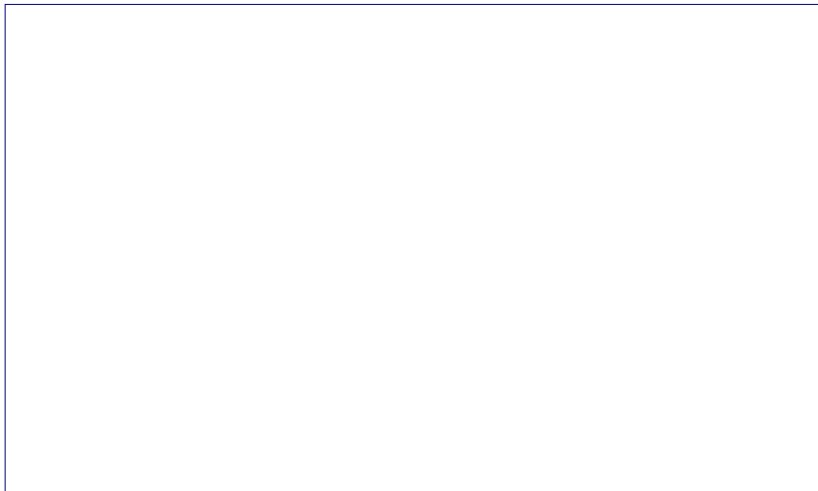
Proof.

From Corollary 5.2.23, skolemization preserves satisfiability, then : Γ is unsatisfiable if and only if $\forall(\Delta)$ is unsatisfiable.

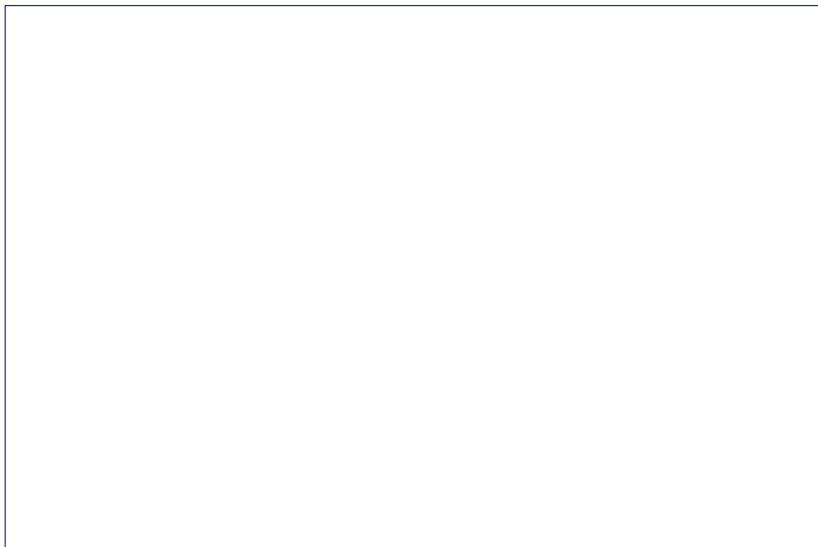
From Corollary 5.1.18 of Herbrand's theorem, $\forall(\Delta)$ is unsatisfiable if and only if there exists a finite unsatisfiable subset of instances of clauses of Δ on the signature of Δ . □

Example 5.2.25 (1/2)

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



Example 5.2.25 (2/2)



Unification : expression, solution

Definition 5.3.1

- ▶ A term or a literal is an **expression**.
- ▶ A substitution σ (see definition 5.1.3) is a **solution** of equation $e_1 = e_2$, if the two expressions $e_1\sigma$ 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 of the set.

Unification : carrier of substitution

Definition 5.3.3

The **carrier** of a substitution σ is the set of variables x such that $x\sigma \neq x$.

We only consider substitutions with a finite carrier (a finite number of variables).

Definition 5.3.3

A **substitution** σ with finite carrier is denoted by

$\langle x_1 := t_1, \dots, x_n := t_n \rangle$ or just $x_1 := t_1, \dots, x_n := t_n$ when there is no ambiguity.

Variables x_1, \dots, x_n are distinct and the substitution satisfies :

- ▶ for i from 1 to n , $x_i\sigma = t_i$
- ▶ for all variables y such that $y \notin \{x_1, \dots, x_n\}$, we have : $y\sigma = y$

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 : composition of substitution

Definition 5.3.5

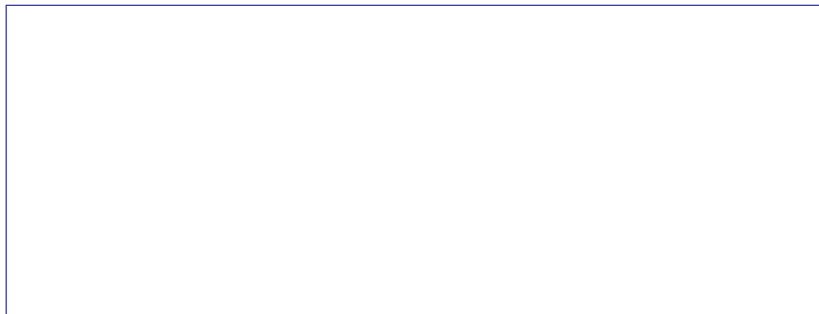
- ▶ Let σ and τ be 2 substitutions, we note $\sigma\tau$ the substitution such that for all variable x , $x\sigma\tau = (x\sigma)\tau$.
- ▶ The substitution $\sigma\tau$ is **an instance** of σ .
- ▶ Two substitutions are **equivalent** if each of them is an instance of the other.

Unification : example 5.3.6

Consider substitutions

- ▶ $\sigma_1 = \langle x := g(z), y := z \rangle$
- ▶ $\sigma_2 = \langle x := g(y), z := y \rangle$
- ▶ $\sigma_3 = \langle x := g(a), y := a, z := a \rangle$

We have the following relations between these substitutions :



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 solution is an instance of it. Note that two « most general » solutions are equivalent.

Example 5.3.8

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

Unifier

Definition 5.3.2

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

Let $\{e_i \mid 1 \leq i \leq n\}$ a finite set of expressions. The substitution σ is a **unifier** of this set if and only if it is a solution of the set of equations $\{e_i = e_{i+1} \mid 1 \leq i < n\}$.

Most General Unifier

Definition 5.3.9

Let E be a set of expressions. Recall that an expression is a term or a literal. A unifier of E is said to be a **most general** (or principal) unifier if any unifier is an instance of it.

Most General Unifier and most general solution

Remark 5.3.10

Let $E = \{e_i \mid 1 \leq i \leq n\}$ a set of expressions.

In the definition of a unifier, we mentioned that σ is a unifier of E if and only if σ is a solution of the set $S = \{e_i = e_{i+1} \mid 1 \leq i < n\}$.

Therefore, the Most General Unifier of E is the most general solution of S .

Unification : algorithm (sketch)

The algorithm separates equations into :

- ▶ equations to be solved, denoted by an equation
- ▶ solved equations, denoted by $:=$

Initially, there is no solved equations.

The algorithm stops when :

- ▶ No equations are still to be solved : the list of solved equations is the most general solution of the initial set of equations.
- ▶ or when it claims that there is no solution.

Unification : algorithm (rules)

- ▶ **Remove the equation.** If the 2 sides of an equation are identical.
- ▶ **Decompose.** If the 2 sides of an equation are distincts :
 - ▶ $\neg A = \neg B$ becomes $A = B$.
 - ▶ $f(s_1, \dots, s_n) = f(t_1, \dots, t_n)$, becomes $s_1 = t_1, \dots, s_n = t_n$.
For $n = 0$ this decomposition removes the equation.
- ▶ **Failure of decomposition** If an equation to be solved is of the form $f(s_1, \dots, s_n) = g(t_1, \dots, t_p)$ with $f \neq g$ then the algorithm claims that there is no solution.
In particular a failure is detected if we look for a solution to an equation between a positive literal and a negative literal.

Unification : algorithm (rules)

- ▶ **Orient.** If an equation is of the form $t = x$ where t is a term which is not a variable and x is a variable, then we replace the equation with $x = t$.
- ▶ **Elimination of a variable.** If an equation to be solved is of the form $x = t$ where x is a variable and t is a term **without occurrence** of x
 1. remove it from equations to be solved
 2. replace x by t in all equations (unsolved **and solved**)
 3. add $x := t$ to the solved part
- ▶ **Failure of elimination.** If an equation to be solved is of the form $x = t$ where x is a variable and t a term distinct from x and **containing** x then the algorithm claims that there is no solution.

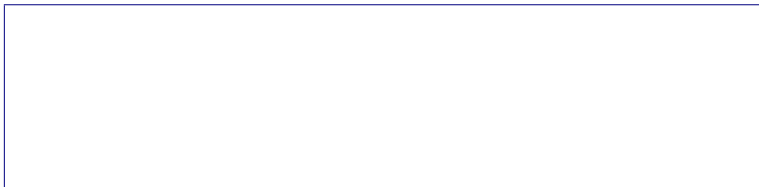
Unification : algorithm (example 5.3.11)

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

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

Unification : algorithm (example 5.3.11)

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



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

Idea

Let Γ be a set of clauses. Suppose that $\forall(\Gamma)$ has no model. What can be done ?

Rules of \ll factorization, copy, binary resolution \gg allow us to infer \perp from Γ .

Completeness of these rules is based on Herbrand's Theorem. The **unification algorithm** is used to find suitable instances of these clauses.

Three rules

1. **Factorization** : from $P(x, f(y)) \vee P(g(z), z) \vee Q(z, x)$ infer $P(g(f(y)), f(y)) \vee Q(f(y), g(f(y)))$. The inferred clause is obtained by computing the most general solution $x := g(f(y)), z := f(y)$ of $P(x, f(y)) = P(g(z), z)$.
2. **The copy rule** which renames the variables of a clause.
3. **Binary resolution** (BR) : from two premises without common variable $P(x, a) \vee Q(x)$ and $\neg P(b, y) \vee R(f(y))$ infer the resolvent $Q(b) \vee R(f(a))$, by computing the most general solution $x := b, y := a$ of $P(x, a) = P(b, y)$.

Resolution : 3 Rules

1. factorization,
2. copy,
3. resolvent

A clause, (a disjunction of literals), is identified with the set of its literals.

Factorization

Definition 5.4.2

The clause C' is a **factor** of clause C if $C' = C$ or if there exists a subset E of C such that E has two elements at least, E is unifiable and $C' = C\sigma$ where σ is the most general unifier of E .

Example 5.4.3

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

Factorization

property 1

5.4.1 Let A be a formula without quantifier and B an instance of A .

$$\forall(A) \models \forall(B)$$

Proof.

See handout course notes. □

property 1

5.4.4 Let C' be a factor of the clause C .

$$\forall(C) \models \forall(C')$$

Proof.

Since C' is an instance of C , it is a consequence of the property 5.4.1. □

Copy

Definition 5.4.5

Let C be a clause and σ a substitution, which changes only the variables of C and whose restriction of variables of C is a bijection between those variables and variables of clause $C\sigma$.

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

We also say that the substitution σ is a renaming of C .

Copy

Definition 5.4.6

Let C be a clause and σ be a renaming of C . Let f the restriction of σ to variables of C and f^{-1} the inverse of f . Let σ_C^{-1} be the substitution defined for all variable x as follows :

- ▶ If x is a variable of $C\sigma$ then $x\sigma_C^{-1} = xf^{-1}$
- ▶ Otherwise $x\sigma_C^{-1} = x$.

This substitution is called the **inverse of the renaming** σ of C .

Copy

Example 5.4.7

Let $\sigma = \langle x := u, y := v \rangle$.

σ is a **renaming** of $P(x, y)$.

The literal $P(u, v)$, where $P(u, v) = P(x, y)\sigma$, is a **copy** of $P(x, y)$.

Let $\tau = \langle u := x, v := y \rangle$. τ is the **inverse of the renaming** σ of $P(x, y)$.

Note that $P(u, v)\tau = P(x, y)$: the literal $P(x, y)$ is a copy of $P(u, v)$ by the renaming τ .

Copy

property 1

5.4.8 Let C be a clause and σ a renaming of C .

1. σ_C^{-1} is a renaming of $C\sigma$.
2. for all expressions or clauses E , whose variables are the ones of C , $E\sigma\sigma_C^{-1} = E$.

Then $C\sigma\sigma_C^{-1} = C$ and therefore **C is a copy of $C\sigma$** .

Proof.

Let f be the restriction of σ to variables of C . By the definition of renaming, f is a bijection between the variables of C and the variables of $C\sigma$.

1. By definition of σ_C^{-1} , this substitution changes only variables of $C\sigma$ and its restriction to variables of $C\sigma$ is the bijection f^{-1} . Therefore, σ_C^{-1} is a renaming of $C\sigma$.
2. Let x a variable of C . By definition of f , $x\sigma\sigma_C^{-1} = xff^{-1} = x$. Therefore, by induction on terms, literals and clauses, for all expressions or clauses E , whose variables are variables of C , we have $E\sigma\sigma_C^{-1} = E$.

□

Copy

property 1

5.4.9 Given two clauses which are a copy of each other, **their universal closures are equivalent.**

Proof.

Let C' be a copy of C . By definition, C' is an instance of C and by the previous property, C is a copy of C' , hence an instance of C .

Therefore by Property 5.4.1, the universal closure of C is a consequence of the universal closure of C' and conversely. Therefore, these two universal closures are equivalent. \square

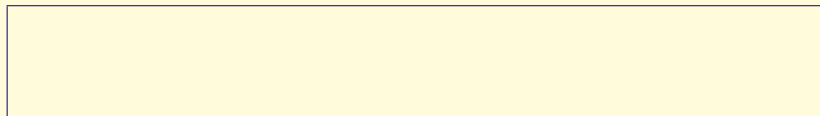
Binary resolvent

Definition 5.4.10

Let C and D be two clauses **without common variables**. The clause E is a **binary resolvent** of C and D if there is a literal $L \in C$ and a literal $M \in D$ such that L and M^c are unifiable and if $E = ((C - \{L\}) \cup (D - \{M\}))\sigma$ where σ is the most general solution of equation $L = M^c$.

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

property 1

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

Proof.

See handout course notes. □

Resolution :

Definition 5.4.13

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

A **proof** of C from Γ is a sequence of clauses terminated by C , where each clause is

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

C is **first-order inferred from Γ** , denoted by $\Gamma \vdash_{1fcb} C$, if there is a proof of C from Γ .

When there is no ambiguity, we replace \vdash_{1fcb} by \vdash .

Resolution : Consistency

property 1

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

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

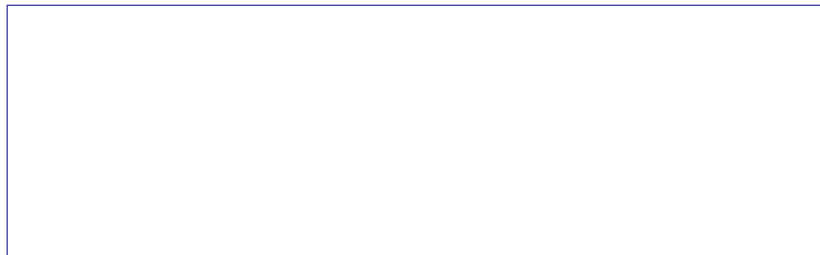
This property is an immediate consequence of consistency of factorization, copy and binary resolution, using induction. See exercise 91.

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.

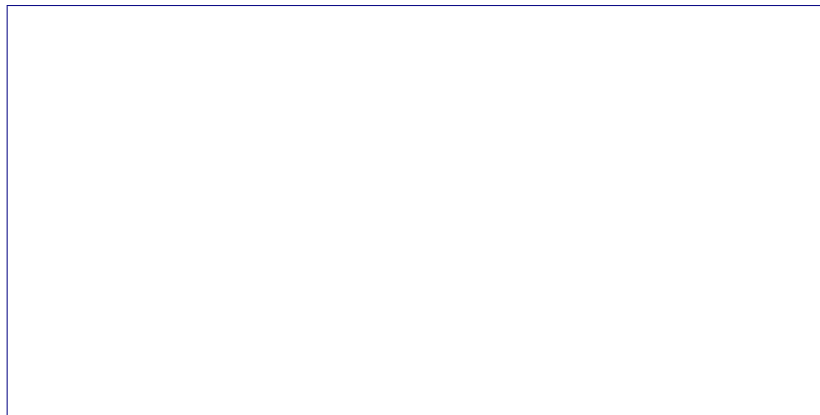


This example shows, a contrario, that binary resolution alone is incomplete : without factorization, the empty clauses cannot be inferred.

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. $C_3 = P(f(z), z) \vee P(z, a)$

We give a proof that $\forall(C_1, C_2, C_3)$ has no model.



First-Order resolution

We define a new 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 and D' is a copy of a factor of D without common variable with C' ,

The rule which infers E from C and D is called **first-order resolution**.

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 .

The clause $P(a, f(a))$ is a binary resolvent of C' and of D (which is factor of itself) then 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

1. $\Gamma \vdash_p C$: proof of C from Γ by propositional resolution (without substitution).
2. $\Gamma \vdash_{1fcb} C$: proof of C from Γ by factorization, copy and binary resolution.
3. $\Gamma \vdash_{1r} C$: proof of C from Γ obtained by first-order 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 two clauses. Let C' be an instance of C and D' be an instance of D . Let E' be a **propositional** resolvent of C' and D' , there exists E a **first-order** resolvent of C and D having E' as an instance.

Proof.

See handout course notes. □

Example 5.4.20

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

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

Lifting theorem (2/3)

Theorem 5.4.21

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

There exists a proof D_1, \dots, D_n by first-order resolution from Γ such that for i between 1 and n , the clause C_i is an instance of D_i .

Proof.

By induction on n . Let C_1, \dots, C_n, C_{n+1} a proof by propositional resolution starting with Δ . By induction, there exists a proof D_1, \dots, D_n by first-order resolution starting from Γ such that, for i between 1 and n , the clause C_i is an instance of D_i .

1. Suppose that $C_{n+1} \in \Delta$. There exists $E \in \Gamma$ where C_{n+1} is an instance then we take $D_{n+1} = E$.
2. Suppose that C_{n+1} is a propositional resolvent of C_j and C_k where $j, k \leq n$. From the previous slide, there exists E , first-order resolvent of D_j and D_k : we take $D_{n+1} = E$.

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

Example 5.4.23

Consider the set of clauses

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

The universal closure of this set of clauses is unsatisfiable and we show it in three ways

1. **By instantiation on the Herbrand's domain** $a, f(a), f(f(a)), \dots$:

$P(f(x)) \vee P(u)$ is instantiated by $x := a, u := f(a)$ to $P(f(a))$

$\neg P(x) \vee Q(z)$ is instantiated by $x := f(a), z := a$ to

$$\neg P(f(a)) \vee Q(a)$$

$\neg Q(x) \vee \neg Q(y)$ is instantiated by $x := a, y := a$ to $\neg Q(a)$

These these 3 instances together are unsatisfiable, as shown in the following proof by propositional resolution :

$$\frac{\frac{P(f(a)) \quad \neg P(f(a)) \vee Q(a)}{Q(a)} \quad \neg Q(a)}{\perp}$$

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** :

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

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

$$\frac{\frac{\frac{P(f(x)) \vee P(u)}{P(f(x))} \text{ fact} \quad \frac{\neg P(x) \vee Q(z)}{\neg P(y) \vee Q(z)} \text{ copy}}{Q(z)} \text{ rb} \quad \frac{\neg Q(x) \vee \neg Q(y)}{\neg Q(x)} \text{ fact}}{\perp} \text{ rb}$$

Refutational completeness of first-order resolution

Theorem 5.4.24

Let Γ be a set of clauses. Propositions : (1) $\Gamma \vdash_{1r} \perp$, (2) $\Gamma \vdash_{1fcb} \perp$, and (3) $\forall(\Gamma) \models \perp$ are equivalent.

Proof.

- ▶ (1) implies (2) because first-order resolution is a combinaison of factorization, copy and binary resolution.
- ▶ (2) implies (3) because factorization, copy and binary resolution are consistent.
- ▶ (3) implies (1). Suppose that $\forall(\Gamma) \models \perp$, that is, $\forall(\Gamma)$ is unsatisfiable. By Herbrand's theorem, there is a finite set Δ of instances without variable of clauses of Γ which has no propositional model. By completeness of propositional resolution, we have : $\Delta \vdash_p \perp$. From the lifting corollary 5.4.22, there exists D such that $\Gamma \vdash_{1r} D$ and \perp is an instance of D . But in this case, we have $D = \perp$.



Conclusion

Thanks of your attention.

Questions ?