Satisfiability Modulo Theories

Lecture 3: FOL Proof Systems

Lydia Kondylidou

WS 2025/26

Outline

- Semantic arguments for FOL
- PCNF and Clausal Form
- First-order Resolution

Proofs in first-order logic

Proof systems for FOL are usually **extensions** of those for PL

For example, we can extend the **semantic arguments system** by

- replacing the truth assignment v with an interpretation $\mathcal{I}$ and
- adding proof rules for quantifiers
- adding proof rules for equality (for FOL **with** equality)

Semantic arguments for FOL: propositional rules

(a) $\dfrac{\mathcal{I} \models \neg\alpha}{\mathcal{I} \not\models \alpha}$

(b) $\dfrac{\mathcal{I} \not\models \neg\alpha}{\mathcal{I} \models \alpha}$

(c) $\dfrac{\mathcal{I} \models \alpha \wedge \beta}{\mathcal{I} \models \alpha, \mathcal{I} \models \beta}$

(d) $\dfrac{\mathcal{I} \not\models \alpha \wedge \beta}{\mathcal{I} \not\models \alpha \mid \mathcal{I} \not\models \beta}$

(e) $\dfrac{\mathcal{I} \models \alpha \vee \beta}{\mathcal{I} \models \alpha \mid \mathcal{I} \models \beta}$

(f) $\dfrac{\mathcal{I} \not\models \alpha \vee \beta}{\mathcal{I} \not\models \alpha, \mathcal{I} \not\models \beta}$

(g) $\dfrac{\mathcal{I} \models \alpha \implies \beta}{\mathcal{I} \not\models \alpha \mid \mathcal{I} \models \beta}$

(h) $\dfrac{\mathcal{I} \not\models \alpha \implies \beta}{\mathcal{I} \models \alpha, \mathcal{I} \not\models \beta}$

(i) $\dfrac{\mathcal{I} \models \alpha \quad \mathcal{I} \not\models \alpha}{\mathcal{I} \models \perp}$

(k) $\dfrac{\mathcal{I} \models \alpha \iff \beta}{\mathcal{I} \models \alpha, \mathcal{I} \models \beta \mid \mathcal{I} \not\models \alpha, \mathcal{I} \not\models \beta}$

(j) $\dfrac{\mathcal{I} \not\models \alpha \iff \beta}{\mathcal{I} \not\models \alpha, \mathcal{I} \models \beta \mid \mathcal{I} \models \alpha, \mathcal{I} \not\models \beta}$

Semantic arguments for FOL: quantifier rules

Notation: if v is a variable, ε is a term/formula, and t is a term, $\varepsilon[v \leftarrow t]$ denotes
the term/formula obtained from ε by replacing every free occurrence of v in ε by t

Semantic arguments for FOL: quantifier rules

Examples:

$$x[x \leftarrow \mathsf{S}\ (\mathbf{y})] = \mathsf{S}\ (\mathbf{y}) \qquad (x + y)[x \leftarrow \mathbf{y}] = \mathbf{y} + y$$

$$x[x \leftarrow \mathsf{S}\ (\mathbf{x})] = \mathsf{S}\ (\mathbf{x}) \qquad (x \doteq y)[x \leftarrow \mathbf{0}] = \mathbf{0} \doteq y$$

$$x[x \leftarrow \mathbf{y}] = \mathbf{y} \qquad (x \doteq x)[x \leftarrow \mathsf{S}\ (\mathbf{x})] = \mathsf{S}\ (\mathbf{x}) \doteq \mathsf{S}\ (\mathbf{x})$$

$$(x \doteq y \ \lor\ x < y)[x \leftarrow \mathsf{S}\ (\mathbf{0})] = \mathsf{S}\ (\mathbf{0}) \doteq y \ \lor\ \mathsf{S}\ (\mathbf{0}) < y$$

$$(x \doteq y \ \lor\ \forall x.\, x < y)[x \leftarrow \mathsf{S}\ (\mathbf{y})] = \mathsf{S}\ (\mathbf{y}) \doteq y \ \lor\ \forall x.\, x < y$$

Semantic arguments for FOL: quantifier rules

Quantifier rules

$$\text{(m)}\ \frac{\mathcal{I} \models \forall v{:}\sigma.\,\alpha}{\mathcal{I} \models \alpha[v \leftarrow t]} \text{ for any term } t \text{ of sort } \sigma$$

$$\text{(n)}\ \frac{\mathcal{I} \not\models \exists v{:}\sigma.\,\alpha}{\mathcal{I} \not\models \alpha[v \leftarrow t]} \text{ for any term } t \text{ of sort } \sigma$$

$$\text{(o)}\ \frac{\mathcal{I} \models \exists v{:}\sigma.\,\alpha}{\mathcal{I} \models \alpha[v \leftarrow k]} \text{ for a } \textbf{fresh} \text{ variable } k \text{ of sort } \sigma$$

$$\text{(p)}\ \frac{\mathcal{I} \not\models \forall v{:}\sigma.\,\alpha}{\mathcal{I} \not\models \alpha[v \leftarrow k]} \text{ for a } \textbf{fresh} \text{ variable } k \text{ of sort } \sigma$$

Proof by deduction: Example 1

Prove that $\exists x.\, P(x) \implies \exists y. P(y)$ is valid

1. $\mathcal{I} \not\models \exists x.\, P(x) \implies \exists y.\, P(y)$
2. $\mathcal{I} \models \exists x.\, P(x)$ by (h) on 1
3. $\mathcal{I} \not\models \exists y.\, P(y)$ by (h) on 1
4. $\mathcal{I} \models P(x_0)$ by (o) on 2
5. $\mathcal{I} \not\models P(x_0)$ by (n) on 3
6. $\mathcal{I} \models \perp$ by (i) on 4, 5

Proof by deduction: Example 2

Prove that $\forall x. (P(x) \implies \exists y. P(y))$ is valid

1. $\mathcal{I} \not\models \forall x. (P(x) \implies \exists y. P(y))$
2. $\mathcal{I} \not\models P(x_0) \implies \exists y. P(y)$ by (p) on 1
3. $\mathcal{I} \models P(x_0)$ by (h) on 2
4. $\mathcal{I} \not\models \exists y. P(y)$ by (h) on 2
5. $\mathcal{I} \not\models P(x_0)$ by (n) on 4
6. $\mathcal{I} \models \perp$ by (i) on 3, 5

Proof by deduction: Example 3

Consider signature Σ with $\Sigma^S = \{\, \mathsf{A} \,\}$, $\Sigma^F = \{\, Q \,\}$, $\mathrm{rank}(Q) = \langle \mathsf{A}, \mathsf{A}, \mathsf{Bool} \rangle$, and all vars of sort A

Prove that $\exists x.\, \forall y.\, Q(x, y) \implies \forall y.\, \exists x.\, Q(x, y)$ is valid

1. $\mathcal{I} \not\models \exists x.\forall y.Q(x, y) \implies \forall y.\exists x.Q(x, y)$
2. $\mathcal{I} \models \exists x.\forall y.Q(x, y)$ by (h) on 1
3. $\mathcal{I} \not\models \forall y.\exists x.Q(x, y)$ by (h) on 1
4. $\mathcal{I} \models \forall y.Q(x_0, y)$ by (o) on 2
5. $\mathcal{I} \not\models \exists x.Q(x, y_0)$ by (p) on 3
6. $\mathcal{I} \models Q(x_0, y_0)$ by (m) on 4
7. $\mathcal{I} \not\models Q(x_0, y_0)$ by (n) on 5
8. $\mathcal{I} \models \bot$ by (i) on 6,7

Refutation Soundness and Completeness

Theorem 1 (Soundness):
For all Σ-formulas α, if there is a closed derivation tree with root $\mathcal{I} \not\models \alpha$ then α is valid

Theorem 2 (Completeness):
For all Σ-formulas α **without equality**, if α is valid, then there is a closed derivation tree with root $\mathcal{I} \not\models \alpha$

Termination?

Does the semantic argument method describe a decision procedure then?

No, for an **invalid** formula, the semantic argument proof system might **not terminate**

Intuition: Consider the invalid formula $\forall x.\, q(x, x)$

1. $\mathcal{I} \not\models \forall x. q(x, x)$
2. $\mathcal{I} \not\models q(x_0, x_0)$ by (m) on 1
3. $\mathcal{I} \not\models q(x_1, x_1)$ by (m) on 1
4. $\mathcal{I} \not\models q(x_2, x_2)$ by (m) on 1
5. ...

There is **no strategy** that guarantees termination in **all cases** of invalid formulas

This shortcoming is **not specific** to this proof system

FOL is only *semi-decidable*: you **can** always **show validity** but **not invalidity**

Prenex Normal Form (PNF)

For AR purposes, it is useful in FOL too impose **syntactic restrictions** on formulas

A Σ-formula α is in *prenex normal form* (PNF) if it has the form

$$Q_1 x_1.\ \cdots\ Q_n x_n.\, \beta$$

where each Q_i is a quantifier and β is a **quantifier-free formula**

Formula α above is in *prenex conjunctive normal form* (PCNF) if, **in addition**, β is in **conjunctive normal form**[a]

Example: The formula below is in PCNF

$$\forall y.\, \exists z.\, ((\overbrace{\underbrace{p(f(y))}_{A_1} \vee \underbrace{q(z)}_{A_2}}^{C_1}) \wedge (\overbrace{\neg \underbrace{q(z)}_{A_2} \vee \underbrace{q(x)}_{A_3}}^{C_2}))$$

[a] If we treat every atomic formula of β as if it was a propositional variable

Clausal Form

A Σ-formula is in *clausal form* if

1. it is in **PCNF**
2. it is **closed** (i.e., it has no free variables)
3. all of its quantifiers are **universal**

Exercise: Which of the following formulas are clausal form?

$\forall y.\, \exists z.\, (p(f(y)) \wedge \neg q(y, z))$ ✗

$\forall y.\, \forall z.\, (p(f(y)) \wedge \neg q(x, z))$ ✗

$\forall y.\, \forall z.\, (p(f(y)) \wedge \neg q(y, z))$ ✓

Clausal Form: transformation

Theorem 3 (Skolem's Theorem):
Any *sentence* can be transformed to an **equi-satisfiable** formula in **clausal form**.

The high level transformation strategy is the following:

$$\text{Sentence} \longrightarrow \text{PNF} \longrightarrow \text{PCNF} \longrightarrow \text{Clausal Form}$$

Running example: $(\forall x.(p(x) \implies q(x))) \implies (\forall x.p(x) \implies \forall x.q(x))$

I: Transforming into PNF

Any sentence can be transformed into a *logically equivalent* formula in PNF in 4 steps.

Example Formula

$$(\forall x.\,(p(x) \implies q(x))) \implies (\forall x.\,p(x) \implies \forall x.\,q(x))$$

Step 1: Rename Bound Variables

Rename bounded variables apart so that:

1. Bounded variables are disjoint from free variables
2. Different quantifiers use different bound variables

$$(\forall x.\,(p(x) \implies q(x))) \implies (\forall y.\,p(y) \implies \forall z.\,q(z))$$

Step 2: Eliminate Implications

Eliminate all occurrences of $\implies$ and $\iff$ using the rewrites:

- $\alpha_1 \iff \alpha_2 \longrightarrow (\alpha_1 \implies \alpha_2) \wedge (\alpha_2 \implies \alpha_1)$
- $\alpha_1 \implies \alpha_2 \longrightarrow \neg\alpha_1 \vee \alpha_2$

$$\neg(\forall x.\, (\neg p(x) \vee q(x))) \vee (\neg\forall y.\, p(y) \vee \forall z.\, q(z))$$

Step 3: Push Negations Inward

Use the rewrites:

- $\neg(\alpha \wedge \beta) \longrightarrow \neg\alpha \vee \neg\beta, \quad \neg(\alpha \vee \beta) \longrightarrow \neg\alpha \wedge \neg\beta$
- $\neg\forall v.\, \alpha \longrightarrow \exists v.\, \neg\alpha, \quad \neg\exists v.\, \alpha \longrightarrow \forall v.\, \neg\alpha$
- $\neg\neg\alpha \longrightarrow \alpha$

$$\exists x.\, (p(x) \wedge \neg q(x)) \vee (\exists y.\, \neg p(y) \vee \forall z.\, q(z))$$

Step 4: Move Quantifiers Outward

Move all quantifiers leftwards using the rewrites:

- $\alpha \bowtie Qv.\beta \longrightarrow Qv.(\alpha \bowtie \beta)$ (ok if v not free in α)
- $(Qv.\alpha) \bowtie \beta \longrightarrow Qv.(\alpha \bowtie \beta)$ (ok if v not free in β)

where $Q \in \{\forall, \exists\}$ and $\bowtie \in \{\wedge, \vee\}$

$$\exists x.\, \forall z.\, \exists y.\, ((p(x) \wedge \neg q(x)) \vee (\neg p(y) \vee q(z)))$$

II: Transforming into PCNF

Transforming a PNF to a **logically equivalent** PCNF is straightforward

We apply the **distributive laws** from propositional logic

$$\exists x.\, \forall z.\, \exists y.\, ((p(x) \wedge \neg q(x)) \vee (\neg p(y) \vee q(z)))$$

becomes

$$\exists x.\, \forall z.\, \exists y.\, ((p(x) \vee \neg p(y) \vee q(z)) \wedge (\neg q(x) \vee \neg p(y) \vee q(z)))$$

This formula contains **existentials** and is therefore **not yet in clausal form**

III: Transforming into Clausal Form (Skolemization)

$$\exists x.\, \forall z.\, \exists y.\, ((p(x) \vee \neg p(y) \vee q(z)) \wedge (\neg q(x) \vee \neg p(y) \vee q(z)))$$

For every **existential quantifier** $\exists v$ in the PCNF, let $u_1, \ldots, u_n$ be the **universally quantified** variables **preceding** $\exists v$,

1. introduce a **fresh** function symbol f_v with arity n and $\langle \mathrm{sort}(u_1), \ldots \mathrm{sort}(u_n), \mathrm{sort}(v) \rangle$
2. delete $\exists v$ and replace every occurrence of v by $f_v(u_1, \ldots, u_n)$

For the formula above, introduce **nullary** function (i.e., a constant) symbol f_x and **unary** function symbol f_y for $\exists x$ and $\exists y$, respectively

$$\forall z.\, ((p(f_x) \vee \neg p(f_y(z)) \vee q(z)) \wedge (\neg q(f_x) \vee \neg p(f_y(z)) \vee q(z)))$$

The functions f_v are called *Skolem functions* and the process of replacing existential quantifiers by functions is called *Skolemization*

Note: Technically, the resulting formula is no longer a Σ-formula, but a Σ_E-formula, where $\Sigma_E^S = \Sigma^S$ and $\Sigma_E^F = \Sigma^F \cup \bigcup_v \{ f_v \}$

Clausal forms as clause sets

As with propositional logic, we can write a formula in **clausal form unambiguously** as a set of clauses

Example:

$$\forall z. ((p(f(z)) \vee \neg p(g(z)) \vee q(z)) \wedge (\neg q(f(z)) \vee \neg p(g(z)) \vee q(z)))$$

can be written as

$$\Delta := \{ \{p(f(z)), \neg p(g(z)), q(z)\}, \{\neg q(f(z)), \neg p(g(z)), q(z)\} \}$$

where all variables are implicitly universally quantified

Traditionally, theorem provers for FOL use the latter version of the clausal form

A resolution-based proof system for PL

Recall: The satisfiability proof system consisting of the rules below is **sound**, **complete** and **terminating** for clause sets in PL

$$\textbf{Resolve}\quad \frac{C_1, C_2 \in \Delta \quad p \in C_1 \quad \neg p \in C_2 \quad C = (C_1 \setminus \{\, p \,\}) \cup (C_2 \setminus \{\, \neg p \,\}) \quad C \notin \Delta \cup \Phi}{\Delta := \Delta \cup \{\, C \,\}}$$

$$\textbf{Clash}\quad \frac{C \in \Delta \quad p, \neg p \in C}{\Delta := \Delta \setminus \{\, C \,\} \quad \Phi := \Phi \cup \{\, C \,\}}$$

$$\textbf{Unsat}\quad \frac{\{\} \in \Delta}{\text{UNSAT}} \qquad\qquad \textbf{Sat}\quad \frac{\text{No other rules apply}}{\text{SAT}}$$

Can we **extend** this proof system **to FOL**?

A resolution-based proof system for FOL?

$$\textbf{Resolve}\quad \frac{C_1, C_2 \in \Delta \quad p \in C_1 \quad \neg p \in C_2 \quad C = (C_1 \setminus \{\, p \,\}) \cup (C_2 \setminus \{\, \neg p \,\}) \quad C \notin \Delta \cup \Phi}{\Delta := \Delta \cup \{\, C \,\}}$$

$$\textbf{Clash}\quad \frac{C \in \Delta \quad p, \neg p \in C}{\Delta := \Delta \setminus \{\, C \,\} \quad \Phi := \Phi \cup \{\, C \,\}} \qquad \textbf{Unsat}\quad \frac{\{\} \in \Delta}{\textsc{unsat}}$$

$$\textbf{Sat}\quad \frac{\text{No other rules apply}}{\textsc{sat}}$$

Consider the FOL clause set below where x, z are variables and a is a constant symbol

$$\Delta := \{\, \{\neg P(z), Q(z)\}, \{P(a)\}, \{\neg Q(x)\} \,\}$$

Note that Δ is equivalent to $\forall z. (P(z) \implies Q(z)) \wedge P(a) \wedge \forall x. \neg Q(x)$, which is unsatisfiable

However, no rules above apply to Δ. We need another rule to deal with variables

A resolution-based proof system for FOL

$$\textbf{Resolve}\ \frac{C_1, C_2 \in \Delta \quad p \in C_1 \quad \neg p \in C_2 \quad C = (C_1 \setminus \{\, p \,\}) \cup (C_2 \setminus \{\, \neg p \,\}) \quad C \notin \Delta \cup \Phi}{\Delta := \Delta \cup \{\, C \,\}}$$

$$\textbf{Clash}\ \frac{C \in \Delta \quad p, \neg p \in C}{\Delta := \Delta \setminus \{\, C \,\} \quad \Phi := \Phi \cup \{\, C \,\}} \qquad \textbf{Inst}\ \frac{C \in \Delta \quad v \in \mathcal{FV}(C) \quad \mathrm{sort}(t) = \mathrm{sort}(v)}{\Delta := \Delta \cup \{C[v \leftarrow t]\}}$$

$$\textbf{Unsat}\ \frac{\{\} \in \Delta}{\textsc{unsat}} \qquad \textbf{Sat}\ \frac{\text{No other rules apply}}{\textsc{sat}}$$

A resolution-based proof system for FOL

Example: $C_1 : \{\neg P(z), Q(z)\}$ $C_2 : \{P(a)\}$ $C_3 : \{\neg Q(x)\}$

Φ	Δ	
$\{\ \}$	$\{\ C_1, C_2, C_3\ \}$	
$\{\ \}$	$\{\ C_1, C_2, C_3, C_4{:}\{\neg P(a), Q(a)\}\ \}$	by **Inst** on C_1 with $z \leftarrow a$
$\{\ \}$	$\{\ C_1, C_2, C_3, C_4, C_5{:}\{Q(a)\}\ \}$	by **Resolve** on C_2, C_4
$\{\ \}$	$\{\ C_1, C_2, C_3, C_4, C_5, C_6{:}\{\neg Q(a)\}\ \}$	by **Inst** on C_3 with $x \leftarrow a$
$\{\ \}$	$\{\ C_1, C_2, C_3, C_4, C_5, C_6, C_7{:}\{\}\ \}$	by **Resolve** on C_5, C_6
	UNSAT	by **Unsat** on C_7

A resolution-based proof system for FOL

This system is **refutation-sound and complete** for FOL clause sets **without equality**:

If a clause set Δ_0 is unsatisfiable, there is a derivation of $\textsc{unsat}$ from Δ_0

The system is also **solution-sound**:

There is a derivation of $\textsc{sat}$ from Δ_0 only if Δ_0 is satisfiable

The system is **not**, and cannot be, **terminating**:

if Δ_0 is satisfiable, it is possible for **Sat** to never apply

Note: This proof system is challenging to implement efficiently because **Inst** is not constrained enough

A resolution-based proof system for FOL

Automated theorem provers for FOL use instead a more sophisticated **Resolve** rule

where two literals in different clauses are instantiated directly, and only as needed, to make them complementary (see ML Chap. 10)

Example: $\{P(x, y), Q(\mathbf{a}, f(y))\}$, $\{\neg Q(\mathbf{z}, f(b)), R(g(\mathbf{z}))\}$ resolve to $\{P(x, b), R(g(\mathbf{a}))\}$

Problem: How do we prove the unsatisfiability of these clause sets?

$\{\{x \doteq y\}, \{\neg(y \doteq x)\}\}$ $\quad$ $\{\{x \doteq y\}, \{y \doteq z\}, \{\neg(x \doteq z)\}\}$ $\quad$ $\{\{x \doteq y\}, \{\neg(f(x) \doteq f(y))\}\}$

We need specialized rules for equality reasoning!

A resolution-based proof system for FOL

Another Problem: How to we prove the unsatisfiability of these clause sets?

$$\{\,\{x < x\}\,\}$$

$$\{\,\{x < y\}, \{y < z\}, \{\neg(x < z)\}\,\}$$

$$\{\,\{\neg(x + y \doteq y + x)\}\,\}$$

$$\{\,\{\neg(x + 0 \doteq x)\}\,\}$$

The thing is: each of these clause set is actually satisfiable in FOL!

However, they are **unsatisfiable in the theory of arithmetic**

We need proof systems for **satisfiability modulo theories**

First-order resolution

$$\frac{l \in C_1 \quad \neg l \in C_2 \quad C_1, C_2 \in \Delta}{\Delta \cup \{(C_1 - \{\, l \,\}) \cup (C_2 - \{\, \neg l \,\})\}} \; (\textit{prop. resolution})$$

where l is a literal (i.e., an atomic formula or its negation).

Now consider $\Delta := \{\, \{\neg P\, z, Q\, z\}, \{P\, a\}, \{\neg Q\, a\} \,\}$, where z is a *universally quantified* variable, and a is a constant.

Is $P\, z$ equal to $P\, a$? No, but they are equal if $z = a$

We can *instantiate* the literals to make them equal and then perform *resolution*

First-order resolution: Unification

A substitution θ is a map from *variables* to *well-sorted terms* (of matching sorts)

Note: we assume the *terms* do not contain any variables

We write $t\theta$ for the result we get by replacing variables in t according to θ

Example: Let $\theta := \{z \mapsto a\}$, then $(p(g(z, z)))\theta = p(g(a, a))$

We use $\{l_1, \ldots, l_n\}\theta$ to represent $\{l_1\theta, \ldots, l_n\theta\}$

A substitution θ is a *unifier* of two terms s and t if $t\theta = s\theta$

Can there be more than one *unifier* of two terms? *Yes. Consider* $p(x)$ *and* $p(f(y))$ *we can have* $x = f(a)$, $y = a$, $x = f(f(a))$, $y = f(a)$ *etc.*

Can there be no *unifier* of two terms? *Yes. Consider* $p(x)$ *and* $q(y)$

First-order resolution

Now we can write first-order resolution as

$$\frac{l_1 \in C_1 \quad \neg l_2 \in C_2 \quad C_1, C_2 \in \Delta \quad \theta \text{ is a unifier of } l_1, l_2}{\Delta \cup \{(C_1 - \{\, l_1 \,\}) \cup (C_2 - \{\, \neg l_2 \,\})\}\theta}\ (\textit{First-order resolution})$$

Example: $\{\neg P\, z, Q\, z\}, \{P\, a\}, \{\neg Q\, a\}$

First-order resolution

Now we can write first-order resolution as

$$\frac{l_1 \in C_1 \quad \neg l_2 \in C_2 \quad C_1, C_2 \in \Delta \quad \theta \text{ is a unifier of } l_1, l_2}{\Delta \cup \{(C_1 - \{\, l_1 \,\}) \cup (C_2 - \{\, \neg l_2 \,\})\}\theta} \; (\textit{First-order resolution})$$

Example

$$\frac{\{\neg P\, z, Q\, z\}, \{P\, a\}, \{\neg Q\, a\}}{\{\neg P\, z, Q\, z\}, \{P\, a\}, \{\neg Q\, a\}, \{\, Q\, a \,\}} \; (\theta := \{\, z \mapsto a \,\} \text{ unifies } P\, z \text{ and } P\, a)$$

First-order resolution

Now we can write first-order resolution as

$$\frac{l_1 \in C_1 \quad \neg l_2 \in C_2 \quad C_1, C_2 \in \Delta \quad \theta \text{ is a unifier of } l_1, l_2}{\Delta \cup \{(C_1 - \{\, l_1 \,\}) \cup (C_2 - \{\, \neg l_2 \,\})\}\theta} \; (\textit{First-order resolution})$$

Example

$$\cfrac{\cfrac{\{\neg P\, z, Q\, z\}, \{P\, a\}, \{\neg Q\, a\}}{\{\neg P\, z, Q\, z\}, \{P\, a\}, \{\neg Q\, a\}, \{\, Q\, a \,\}} \; (\theta := \{\, z \mapsto a \,\} \text{ unifies } P\, z \text{ and } \{P\, a\})}{\{\neg P\, z, Q\, z\}, \{P\, a\}, \{\neg Q\, a\}, \{Q\, a\}, \{\,\}} \; (\text{Resolve } \{\neg Q\, a\}, \{Q\, a\})$$

Therefore, $\alpha := \forall z.((\neg P\, z \vee Q\, z) \wedge P\, a \wedge \neg Q\, a)$ is *unsatisfiable*.

What do we know about $\neg\alpha$?

$\neg\alpha$ is *valid*.

First-order resolution

This suggests a strategy to *prove* the validity of a Σ-formula α:

1. Negate the formula;
2. Transform into Clausal Form;
3. Apply *first-order resolution* until an empty clause is derived (*might not terminate!*)