MANCHESTER 1824

The University of Manchester

Automated reasoning for first-order logic

Theory, Practice and Challenges

Konstantin Korovin[1]

The University of Manchester
UK

korovin@cs.man.ac.uk

Part I

[1]supported by a Royal Society University Fellowship

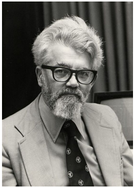
Acknowledgments

- Harald Ganzinger
- Zurab Khasidashvili
- Renate Schmidt
- Christoph Sticksel
- Andrei Voronkov
- ...

Logic and Automated Reasoning

Applications:

- software and hardware verification: Intel, Microsoft
- information management: biomedical ontologies, semantic Web, databases
- combinatorial reasoning: constraint satisfaction, planning, scheduling
- Internet security
- theorem proving in mathematics

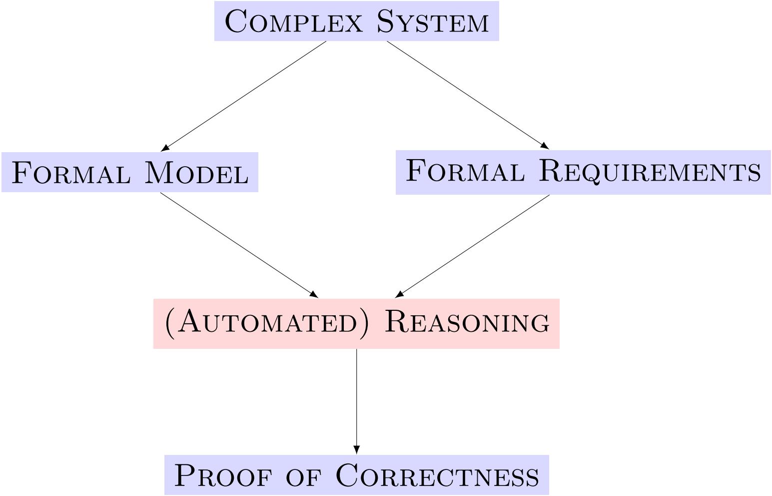


John McCarthy

"It is reasonable to hope that the relationship between computation and mathematical logic will be as fruitful in the next century as that between analysis and physics in the past."

McCarthy, 1963.

Formalising Complex Systems



Automated Reasoning

The complexity of current engineering systems is enormous:

- Intel Microprocessor: 2 billion transistors
- Microsoft Windows: 50 million lines of code

Complexity is rapidly growing!

Automated reasoning methods are crucial!

In this lectures we will focus on efficient automated reasoning for first-order logic.

Automated Reasoning

The complexity of current engineering systems is enormous:

- Intel Microprocessor: 2 billion transistors
- Microsoft Windows: 50 million lines of code

Complexity is rapidly growing!

Automated reasoning methods are crucial!

In this lectures we will focus on efficient automated reasoning for first-order logic.

First-order reasoning

- Theory:
  - resolution, superposition, instantiation
  - completeness, redundancy elimination, decision procedures
- Applications:
  - software/hardware verification
  - semantic Web, security, multi-agent systems, bio-health
- Reasoning systems for FOL:
  - Resolution/superposition-based:
    Vampire, E, SPASS, Prover9, Metis, Waldmeister
  - Instantiation-based:
    iProver, Darwin, Equinox
  - Tableaux, connection, geometric, natural deduction:
    leanCoP, Princess, GEO, Muscadet
- CASC – The World Championship for Automated Theorem Proving

First-order reasoning

- Theory:
  - resolution, superposition, instantiation
  - completeness, redundancy elimination, decision procedures
- Applications:
  - software/hardware verification
  - semantic Web, security, multi-agent systems, bio-health
- Reasoning systems for FOL:
  - Resolution/superposition-based:
    Vampire, E, SPASS, Prover9, Metis, Waldmeister
  - Instantiation-based:
    iProver, Darwin, Equinox
  - Tableaux, connection, geometric, natural deduction:
    leanCoP, Princess, GEO, Muscadet
- CASC – The World Championship for Automated Theorem Proving

First-order reasoning

- Theory:
  - resolution, superposition, instantiation
  - completeness, redundancy elimination, decision procedures
- Applications:
  - software/hardware verification
  - semantic Web, security, multi-agent systems, bio-health
- Reasoning systems for FOL:
  - Resolution/superposition-based:

    Vampire, E, SPASS, Prover9, Metis, Waldmeister
  - Instantiation-based:

    iProver, Darwin, Equinox
  - Tableaux, connection, geometric, natural deduction:

    leanCoP, Princess, GEO, Muscadet
- CASC – The World Championship for Automated Theorem Proving

First-order reasoning

- Theory:
  - resolution, superposition, instantiation
  - completeness, redundancy elimination, decision procedures
- Applications:
  - software/hardware verification
  - semantic Web, security, multi-agent systems, bio-health
- Reasoning systems for FOL:
  - Resolution/superposition-based:

    Vampire, E, SPASS, Prover9, Metis, Waldmeister
  - Instantiation-based:

    iProver, Darwin, Equinox
  - Tableaux, connection, geometric, natural deduction:

    leanCoP, Princess, GEO, Muscadet
- CASC – The World Championship for Automated Theorem Proving

These lectures

Reasoning for first-Order logic

- First-order logic
- Resolution-based methods
- Instantiation-based methods
- Effectively propositional fragment (EPR)
- Applications: bounded model checking and finite model finding
- Implementation techniques:

  proof search, indexing, redundancy elimination

Why first-order logic?

- expressive: quantifiers are needed in many applications
- expressivity comes at a price: first-order logic is semi-decidable
- reasoning can be done at a higher level and can gain in efficiency
- has efficient reasoning methods

Syntax of first-order logic

First-order logic terms

$$\forall x \forall i \forall z \quad (same\_content(store(x,i,e),z) \rightarrow [out\_of\_bounds(x,i) \vee \exists j(select(z,j) \simeq e)])$$

- Signature $\Sigma = (\mathcal{F}, \mathcal{P})$
  - function symbols with arities: $\mathcal{F} = \{store/3, select/2\}$
    constants are function symbols of arity 0,
  - predicate symbols with arities:
    $\mathcal{P} = \{same\_content/2, out\_of\_bounds/2, \simeq /2\}$
- Variables: $\mathcal{X} = \{x, y, z, i, j, \ldots\}$ – infinitely countable set
- Terms:
  - variable terms: x where $x \in \mathcal{X}$
  - function terms: $f(t_1, \ldots, t_n)$, where $f \in \mathcal{F}$ and t_i are terms
- A term is ground if it does not contain variables
- $T(\Sigma, \mathcal{X})$ – the set of all terms over signature Σ and variables $\mathcal{X}$
- $T(\Sigma, \emptyset)$ – the set of all ground terms

First-order logic terms

$$\forall x \forall i \forall z \quad (\mathit{same\_content}(\mathit{store}(x, i, e), z) \rightarrow [\mathit{out\_of\_bounds}(x, i) \vee \exists j(\mathit{select}(z, j) \simeq e)])$$

- Signature $\Sigma = (\mathcal{F}, \mathcal{P})$
  - function symbols with arities: $\mathcal{F} = \{\mathit{store}/3, \mathit{select}/2\}$
    constants are function symbols of arity 0,
  - predicate symbols with arities:
    $\mathcal{P} = \{\mathit{same\_content}/2, \mathit{out\_of\_bounds}/2, \simeq /2\}$
- Variables: $\mathcal{X} = \{x, y, z, i, j, \ldots\}$ – infinitely countable set
- Terms:
  - variable terms: x where $x \in \mathcal{X}$
  - function terms: $f(t_1, \ldots, t_n)$, where $f \in \mathcal{F}$ and t_i are terms
- A term is ground if it does not contain variables
- $T(\Sigma, \mathcal{X})$ – the set of all terms over signature Σ and variables $\mathcal{X}$
- $T(\Sigma, \emptyset)$ – the set of all ground terms

First-order logic terms

$$\forall x \forall i \forall z \quad (same\_content(store(x,i,e),z) \rightarrow [out\_of\_bounds(x,i) \vee \exists j(select(z,j) \simeq e)])$$

- Signature $\Sigma = (\mathcal{F}, \mathcal{P})$
  - function symbols with arities: $\mathcal{F} = \{store/3, select/2\}$
    constants are function symbols of arity 0,
  - predicate symbols with arities:
    $\mathcal{P} = \{same\_content/2, out\_of\_bounds/2, \simeq /2\}$
- Variables: $\mathcal{X} = \{x, y, z, i, j, \ldots\}$ – infinitely countable set
- Terms:
  - variable terms: x where $x \in \mathcal{X}$
  - function terms: $f(t_1, \ldots, t_n)$, where $f \in \mathcal{F}$ and t_i are terms
- A term is ground if it does not contain variables
- $T(\Sigma, \mathcal{X})$ – the set of all terms over signature Σ and variables $\mathcal{X}$
- $T(\Sigma, \emptyset)$ – the set of all ground terms

First-order logic terms

$$\forall x \forall i \forall z \quad (same\_content(store(x, i, e), z) \rightarrow [out\_of\_bounds(x, i) \vee \exists j (select(z, j) \simeq e)])$$

- Signature $\Sigma = (\mathcal{F}, \mathcal{P})$
  - function symbols with arities: $\mathcal{F} = \{store/3, select/2\}$
    constants are function symbols of arity 0,
  - predicate symbols with arities:
    $\mathcal{P} = \{same\_content/2, out\_of\_bounds/2, \simeq /2\}$
- Variables: $\mathcal{X} = \{x, y, z, i, j, \ldots\}$ – infinitely countable set
- Terms:
  - variable terms: x where $x \in \mathcal{X}$
  - function terms: $f(t_1, \ldots, t_n)$, where $f \in \mathcal{F}$ and t_i are terms
- A term is ground if it does not contain variables
- $T(\Sigma, \mathcal{X})$ – the set of all terms over signature Σ and variables $\mathcal{X}$
- $T(\Sigma, \emptyset)$ – the set of all ground terms

First-order logic terms

$$\forall x \forall i \forall z \quad (same\_content(store(x,i,e),z) \rightarrow [out\_of\_bounds(x,i) \vee \exists j(select(z,j) \simeq e)])$$

- Signature $\Sigma = (\mathcal{F}, \mathcal{P})$
  - function symbols with arities: $\mathcal{F} = \{store/3, select/2\}$
    constants are function symbols of arity 0,
  - predicate symbols with arities:
    $\mathcal{P} = \{same\_content/2, out\_of\_bounds/2, \simeq /2\}$
- Variables: $\mathcal{X} = \{x, y, z, i, j, \ldots\}$ – infinitely countable set
- Terms:
  - variable terms: x where $x \in \mathcal{X}$
  - function terms: $f(t_1, \ldots, t_n)$, where $f \in \mathcal{F}$ and t_i are terms
- A term is ground if it does not contain variables
- $T(\Sigma, \mathcal{X})$ – the set of all terms over signature Σ and variables $\mathcal{X}$
- $T(\Sigma, \emptyset)$ – the set of all ground terms

First-order logic terms

$$\forall x \forall i \forall z \quad (same\_content(store(x, i, e), z) \rightarrow [out\_of\_bounds(x, i) \vee \exists j(select(z, j) \simeq e)])$$

- Signature $\Sigma = (\mathcal{F}, \mathcal{P})$
  - function symbols with arities: $\mathcal{F} = \{store/3, select/2\}$
    constants are function symbols of arity 0,
  - predicate symbols with arities:
    $\mathcal{P} = \{same\_content/2, out\_of\_bounds/2, \simeq /2\}$
- Variables: $\mathcal{X} = \{x, y, z, i, j, \ldots\}$ – infinitely countable set
- Terms:
  - variable terms: x where $x \in \mathcal{X}$
  - function terms: $f(t_1, \ldots, t_n)$, where $f \in \mathcal{F}$ and t_i are terms
- A term is ground if it does not contain variables
- $T(\Sigma, \mathcal{X})$ – the set of all terms over signature Σ and variables $\mathcal{X}$
- $T(\Sigma, \emptyset)$ – the set of all ground terms

First-order logic terms

$$\forall x \forall i \forall z \quad (same\_content(store(x, i, e), z) \rightarrow [out\_of\_bounds(x, i) \vee \exists j(select(z, j) \simeq e)])$$

- Signature $\Sigma = (\mathcal{F}, \mathcal{P})$
  - function symbols with arities: $\mathcal{F} = \{store/3, select/2\}$
    constants are function symbols of arity 0,
  - predicate symbols with arities:
    $\mathcal{P} = \{same\_content/2, out\_of\_bounds/2, \simeq /2\}$
- Variables: $\mathcal{X} = \{x, y, z, i, j, \ldots\}$ – infinitely countable set
- Terms:
  - variable terms: x where $x \in \mathcal{X}$
  - function terms: $f(t_1, \ldots, t_n)$, where $f \in \mathcal{F}$ and t_i are terms
- A term is ground if it does not contain variables
- $T(\Sigma, \mathcal{X})$ – the set of all terms over signature Σ and variables $\mathcal{X}$
- $T(\Sigma, \emptyset)$ – the set of all ground terms

First-order logic syntax (formulas)

Example:

$$\forall x, i, z \quad (same\_content(store(x, i, e), z) \rightarrow [out\_of\_bounds(x, i) \vee \exists j(select(z, j) \simeq e)])$$

Formulas:

- atomic formulas: $p(t_1, \ldots, t_n)$, where p is a predicate symbol
- Boolean combinations: $\neg F,\ F_1 \wedge F_2,\ F_1 \vee F_2,\ F_1 \rightarrow F_2,\ F_1 \leftrightarrow F_2$
- quantifier applications: $\forall \bar{x} F(\bar{x}),\ \exists \bar{x} F(\bar{x})$

$\mathcal{F}(\mathcal{X})$ – the set of all formulas over variables $\mathcal{X}$.

First-order logic syntax (formulas)

Example:

$$\forall x, i, z \quad (same\_content(store(x, i, e), z) \rightarrow [out\_of\_bounds(x, i) \vee \exists j(select(z, j) \simeq e)])$$

Formulas:

- atomic formulas: $p(t_1, \ldots, t_n)$, where p is a predicate symbol
- Boolean combinations: $\neg F$, $F_1 \wedge F_2$, $F_1 \vee F_2$, $F_1 \rightarrow F_2$, $F_1 \leftrightarrow F_2$
- quantifier applications: $\forall \bar{x} F(\bar{x})$, $\exists \bar{x} F(\bar{x})$

$\mathcal{F}(\mathcal{X})$ – the set of all formulas over variables $\mathcal{X}$.

First-order logic syntax (formulas)

Example:

$$\forall x, i, z \quad (same\_content(store(x, i, e), z) \rightarrow [out\_of\_bounds(x, i) \vee \exists j(select(z, j) \simeq e)])$$

Formulas:

- atomic formulas: $p(t_1, \ldots, t_n)$, where p is a predicate symbol
- Boolean combinations: $\neg F,\ F_1 \wedge F_2,\ F_1 \vee F_2,\ F_1 \rightarrow F_2,\ F_1 \leftrightarrow F_2$
- quantifier applications: $\forall \bar{x} F(\bar{x}),\ \exists \bar{x} F(\bar{x})$

$\mathcal{F}(\mathcal{X})$ – the set of all formulas over variables $\mathcal{X}$.

First-order logic syntax (formulas)

Example:

$$\forall x, i, z \quad (same\_content(store(x, i, e), z) \rightarrow [out\_of\_bounds(x, i) \vee \exists j(select(z, j) \simeq e)])$$

Formulas:

- atomic formulas: $p(t_1, \ldots, t_n)$, where p is a predicate symbol
- Boolean combinations: $\neg F,\ F_1 \wedge F_2,\ F_1 \vee F_2,\ F_1 \rightarrow F_2,\ F_1 \leftrightarrow F_2$
- quantifier applications: $\forall \bar{x} F(\bar{x}),\ \exists \bar{x} F(\bar{x})$

$\mathcal{F}(\mathcal{X})$ – the set of all formulas over variables $\mathcal{X}$.

First-order logic syntax (formulas)

Example:

$$\forall x, i, z \quad (same\_content(store(x, i, e), z) \rightarrow [out\_of\_bounds(x, i) \vee \exists j(select(z, j) \simeq e)])$$

Formulas:

- atomic formulas: $p(t_1, \ldots, t_n)$, where p is a predicate symbol
- Boolean combinations: $\neg F,\ F_1 \wedge F_2,\ F_1 \vee F_2,\ F_1 \rightarrow F_2,\ F_1 \leftrightarrow F_2$
- quantifier applications: $\forall \bar{x} F(\bar{x}),\ \exists \bar{x} F(\bar{x})$

$\mathcal{F}(\mathcal{X})$ – the set of all formulas over variables $\mathcal{X}$.

free/bound variable occurrences

$$F(y) = \forall x (p(x, y) \rightarrow \exists y q(y, x))$$

- A variable occurrence is bound if it is under the scope of a quantifier
- A variable occurrence is free if it is not bound
- A formula is closed, also called a sentence if it does not contain free variables

Note: the same variable can have both free and bound occurrences.

Rectified formula:

- no variable occur both free and bound
- a variable is quantified only once

Rectifying a formula: rename quantified variables

$$F'(y) = \forall x (p(x, y) \rightarrow \exists y_1 q(y_1, x))$$

$F(y)$ is equivalent to $F'(y)$

We will assume that all formulas are rectified.

free/bound variable occurrences

$$F(y) = \forall x(p(x, y) \rightarrow \exists y q(y, x))$$

- A variable occurrence is bound if it is under the scope of a quantifier
- A variable occurrence is free if it is not bound
- A formula is closed, also called a sentence if it does not contain free variables

Note: the same variable can have both free and bound occurrences.

Rectified formula:

- no variable occur both free and bound
- a variable is quantified only once

Rectifying a formula: rename quantified variables

$$F'(y) = \forall x(p(x, y) \rightarrow \exists y_1 q(y_1, x))$$

$F(y)$ is equivalent to $F'(y)$

We will assume that all formulas are rectified.

free/bound variable occurrences

$$F(y) = \forall x(p(x, y) \to \exists y q(y, x))$$

- A variable occurrence is bound if it is under the scope of a quantifier
- A variable occurrence is free if it is not bound
- A formula is closed, also called a sentence if it does not contain free variables

Note: the same variable can have both free and bound occurrences.

Rectified formula:

- no variable occur both free and bound
- a variable is quantified only once

Rectifying a formula: rename quantified variables

$$F'(y) = \forall x(p(x, y) \to \exists y_1 q(y_1, x))$$

$F(y)$ is equivalent to $F'(y)$

We will assume that all formulas are rectified.

free/bound variable occurrences

$$F(y) = \forall x(p(x, y) \rightarrow \exists y q(y, x))$$

- A variable occurrence is bound if it is under the scope of a quantifier
- A variable occurrence is free if it is not bound
- A formula is closed, also called a sentence if it does not contain free variables

Note: the same variable can have both free and bound occurrences.

Rectified formula:

- no variable occur both free and bound
- a variable is quantified only once

Rectifying a formula: rename quantified variables

$$F'(y) = \forall x(p(x, y) \rightarrow \exists y_1 q(y_1, x))$$

$F(y)$ is equivalent to $F'(y)$

We will assume that all formulas are rectified.

free/bound variable occurrences

$$F(y) = \forall x(p(x, y) \rightarrow \exists y q(y, x))$$

- A variable occurrence is bound if it is under the scope of a quantifier
- A variable occurrence is free if it is not bound
- A formula is closed, also called a sentence if it does not contain free variables

Note: the same variable can have both free and bound occurrences.

Rectified formula:

- no variable occur both free and bound
- a variable is quantified only once

Rectifying a formula: rename quantified variables

$$F'(y) = \forall x(p(x, y) \rightarrow \exists y_1 q(y_1, x))$$

$F(y)$ is equivalent to $F'(y)$

We will assume that all formulas are rectified.

free/bound variable occurrences

$$F(y) = \forall x(p(x, y) \to \exists y q(y, x))$$

- A variable occurrence is bound if it is under the scope of a quantifier
- A variable occurrence is free if it is not bound
- A formula is closed, also called a sentence if it does not contain free variables

Note: the same variable can have both free and bound occurrences.

Rectified formula:

- no variable occur both free and bound
- a variable is quantified only once

Rectifying a formula: rename quantified variables

$$F'(y) = \forall x(p(x, y) \to \exists y_1 q(y_1, x))$$

$F(y)$ is equivalent to $F'(y)$

We will assume that all formulas are rectified.

Substitutions

A substitution: is a mapping $\sigma : X \mapsto T(\Sigma, X)$ such that $\sigma(x) \neq x$ is finite.

Example:

$$\sigma = \{x \mapsto a, y \mapsto f(x, g(z))\}$$

where σ is assumed to be identity for all variables different from x, y.

The domain of σ:

$$dom(\sigma) = \{x \mid x \in X, \sigma(x) \neq x\}$$

Notation:

$$\begin{aligned} \sigma = & \quad \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\} \\ \sigma = & \quad \{t_1/x_1, \ldots, t_n/x_n\} \end{aligned}$$

Application of a substitution to a term/formula: – simultaneous replacement of variables by terms.

$$(p(f(x,x),y) \vee q(g(y)))\sigma = p(f(a,a), f(x,g(z))) \vee q(g(f(x,g(z))))$$

Substitutions

A substitution: is a mapping $\sigma : X \mapsto T(\Sigma, X)$ such that $\sigma(x) \neq x$ is finite.

Example:

$$\sigma = \{x \mapsto a, y \mapsto f(x, g(z))\}$$

where σ is assumed to be identity for all variables different from x, y.

The domain of σ:

$$dom(\sigma) = \{x \mid x \in X, \sigma(x) \neq x\}$$

Notation:

$$\begin{aligned} \sigma = & \quad \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\} \\ \sigma = & \quad \{t_1/x_1, \ldots, t_n/x_n\} \end{aligned}$$

Application of a substitution to a term/formula: – simultaneous replacement of variables by terms.

$$(p(f(x, x), y) \vee q(g(y)))\sigma = p(f(a, a), f(x, g(z))) \vee q(g(f(x, g(z))))$$

Semantics of first-order logic

First-order interpretation

Consider a signature $\Sigma = (\mathcal{F}, \mathcal{P})$.

A first-order Σ-structure is a triple:

$$\mathcal{A} = (A, \mathcal{F}^{\mathcal{A}}, \mathcal{P}^{\mathcal{A}})$$

where

- $\mathcal{F}^{\mathcal{A}}$ is a collection of functions $\{f_{\mathcal{A}} : A^n \mapsto A \mid f/n \in \mathcal{F}\}$
- $\mathcal{P}^{\mathcal{A}}$ is a collection of relations $\{p_{\mathcal{A}} \subseteq A^n \mid p/n \in \mathcal{P}\}$

Examples: Let $\Sigma = (\{+/2, */2, 0\}, \{\leq /2\})$.

Σ-structures:

- $\mathbb{N} = (N, \{+_{\mathbb{N}}, *_{\mathbb{N}}, 0_{\mathbb{N}}\}, \{\leq_{\mathbb{N}}\})$ – natural numbers
- $\mathbb{R} = (R, \{+_{\mathbb{R}}, *_{\mathbb{R}}, 0_{\mathbb{R}}\}, \{\leq_{\mathbb{R}}\})$ – reals
- $\mathbb{L} = (\mathcal{P}(N), \{+_{\mathbb{L}}, *_{\mathbb{L}}, 0_{\mathbb{L}}\}, \{\leq_{\mathbb{L}}\})$ – lattice over the power set of N where $+_{\mathbb{L}}$ is union of sets, $*_{\mathbb{L}}$ is intersection of sets, $\leq_{\mathbb{L}}$ is subset relation.

First-order interpretation

Consider a signature $\Sigma = (\mathcal{F}, \mathcal{P})$.

A first-order Σ-structure is a triple:

$$\mathcal{A} = (A, \mathcal{F}^{\mathcal{A}}, \mathcal{P}^{\mathcal{A}})$$

where

- $\mathcal{F}^{\mathcal{A}}$ is a collection of functions $\{f_{\mathcal{A}} : A^n \mapsto A \mid f/n \in \mathcal{F}\}$
- $\mathcal{P}^{\mathcal{A}}$ is a collection of relations $\{p_{\mathcal{A}} \subseteq A^n \mid p/n \in \mathcal{P}\}$

Examples: Let $\Sigma = (\{+/2, */2, 0\}, \{\leq /2\})$.

Σ-structures:

- $\mathbb{N} = (N, \{+_{\mathbb{N}}, *_{\mathbb{N}}, 0_{\mathbb{N}}\}, \{\leq_{\mathbb{N}}\})$ – natural numbers
- $\mathbb{R} = (R, \{+_{\mathbb{R}}, *_{\mathbb{R}}, 0_{\mathbb{R}}\}, \{\leq_{\mathbb{R}}\})$ – reals
- $\mathbb{L} = (\mathcal{P}(N), \{+_{\mathbb{L}}, *_{\mathbb{L}}, 0_{\mathbb{L}}\}, \{\leq_{\mathbb{L}}\})$ – lattice over the power set of N where $+_{\mathbb{L}}$ is union of sets, $*_{\mathbb{L}}$ is intersection of sets, $\leq_{\mathbb{L}}$ is subset relation.

Semantics of first-order logic

Consider a structure $\mathcal{A} = (A, \mathcal{F}^{\mathcal{A}}, \mathcal{P}^{\mathcal{A}})$.

A variable assignment: $\gamma : \mathcal{X} \mapsto A$

An interpretation is a pair: $\mathcal{I} = (\mathcal{A}, \gamma)$

For every therm t define value $\mathcal{I}(t)$ of t under $\mathcal{I}$ as follows:

- $\mathcal{I}(t) = \gamma(t)$ if t is a variable
- $\mathcal{I}(f(t_1, \ldots, t_n)) = f_{\mathcal{A}}(\mathcal{I}(t_1), \ldots, \mathcal{I}(t_n))$

Note that $\mathcal{I}(t) \in A$.

Example: Consider $\mathbb{N} = (N, \{+/2, */2\}, \{\leq /2, \simeq /2\})$,
$\gamma = \{x \mapsto 0, y \mapsto 1\}$ and $\mathcal{I} = (\mathbb{N}, \gamma)$. Then

- $\mathcal{I}(x + (y + y) * (y + y)) = 4$

Notation: γ_x^a is a variable assignment coinciding with γ on all variables except x where it is equal to a.

Evaluation of formulas

A formula $F(\bar{x})$ is true in an interpretation $\mathcal{I} = (\mathcal{A}, \gamma)$, denoted as $\mathcal{I} \models F(\bar{x})$ if the following holds:

- atomic formulas: $\mathcal{I} \models p(t_1, \ldots, t_n)$ iff $(\mathcal{I}(t_1), \ldots, \mathcal{I}(t_n)) \in p^{\mathcal{A}}$.
- Boolean combinations:
  - $\mathcal{I} \models \neg F(\bar{x})$ iff $\mathcal{I} \models F(\bar{x})$ does not hold
  - $\mathcal{I} \models F_1(\bar{x}) \wedge F_2(\bar{x})$ iff $\mathcal{I} \models F_1(\bar{x})$ and $\mathcal{I} \models F_2(\bar{x})$
  - $\mathcal{I} \models F_1(\bar{x}) \vee F_2(\bar{x})$ iff $\mathcal{I} \models F_1(\bar{x})$ or $\mathcal{I} \models F_2(\bar{x})$
  - $\mathcal{I} \models F_1(\bar{x}) \rightarrow F_2(\bar{x})$ iff $\mathcal{I} \not\models F_1(\bar{x})$ or $\mathcal{I} \models F_2(\bar{x})$
  - $\mathcal{I} \models F_1(\bar{x}) \leftrightarrow F_2(\bar{x})$ iff $\mathcal{I} \models F_1(\bar{x})$ if and only if $\mathcal{I} \models F_2(\bar{x})$
- quantified formulas:
  - $\mathcal{I} \models \forall x F(\bar{x})$ iff for every $a \in A$, $(\mathcal{A}, \gamma_x^a) \models F(\bar{x})$,
  - $\mathcal{I} \models \exists x F(\bar{x})$ iff there exists $a \in A$ such that $(\mathcal{A}, \gamma_x^a) \models F(\bar{x})$

Evaluation of formulas

Example: Consider $\mathbb{N} = (N, \{+, *\}, \{\leq, \simeq\})$, $\gamma = \{x \mapsto 2, y \mapsto 1\}$ and $\mathcal{I} = (\mathbb{N}, \gamma)$. Then

- $\mathcal{I} \models \forall z(x \leq z + y \rightarrow (x \leq z \vee z + y \simeq x))$
- $\mathcal{I} \models \forall z \exists u(z \leq u)$
- $\mathcal{I} \not\models \exists u \forall z(z \leq u)$

Notation: Consider an interpretation $\mathcal{I} = (\mathcal{A}, \gamma)$ and a formula $F(x_1, \ldots, x_n)$. Assume $\gamma(x_i) = a_i$ for $1 \leq i \leq n$. Then we write $\mathcal{A} \models F[a_1, \ldots, a_n]$ in place of $\mathcal{I} \models F(x_1, \ldots, x_n)$.

Note that for any closed formula F its true value does not depend on γ, in this case we can write $\mathcal{A} \models F$. We say $\mathcal{A}$ is a model for F.

Evaluation of formulas

Example: Consider $\mathbb{N} = (N, \{+, *\}, \{\leq, \simeq\})$, $\gamma = \{x \mapsto 2, y \mapsto 1\}$ and $\mathcal{I} = (\mathbb{N}, \gamma)$. Then

- $\mathcal{I} \models \forall z(x \leq z + y \to (x \leq z \vee z + y \simeq x))$
- $\mathcal{I} \models \forall z \exists u(z \leq u)$
- $\mathcal{I} \not\models \exists u \forall z(z \leq u)$

Notation: Consider an interpretation $\mathcal{I} = (\mathcal{A}, \gamma)$ and a formula $F(x_1, \ldots, x_n)$. Assume $\gamma(x_i) = a_i$ for $1 \leq i \leq n$. Then we write $\mathcal{A} \models F[a_1, \ldots, a_n]$ in place of $\mathcal{I} \models F(x_1, \ldots, x_n)$.

Note that for any closed formula F its true value does not depend on γ, in this case we can write $\mathcal{A} \models F$. We say $\mathcal{A}$ is a model for F.

Evaluation of formulas

Example: Consider $\mathbb{N} = (N, \{+, *\}, \{\leq, \simeq\})$, $\gamma = \{x \mapsto 2, y \mapsto 1\}$ and $\mathcal{I} = (\mathbb{N}, \gamma)$. Then

- $\mathcal{I} \models \forall z(x \leq z + y \to (x \leq z \vee z + y \simeq x))$
- $\mathcal{I} \models \forall z \exists u(z \leq u)$
- $\mathcal{I} \not\models \exists u \forall z(z \leq u)$

Notation: Consider an interpretation $\mathcal{I} = (\mathcal{A}, \gamma)$ and a formula $F(x_1, \ldots, x_n)$. Assume $\gamma(x_i) = a_i$ for $1 \leq i \leq n$. Then we write $\mathcal{A} \models F[a_1, \ldots, a_n]$ in place of $\mathcal{I} \models F(x_1, \ldots, x_n)$.

Note that for any closed formula F its true value does not depend on γ, in this case we can write $\mathcal{A} \models F$. We say $\mathcal{A}$ is a model for F.

Validity, satisfiability,

A (closed) formula F is

- satisfiable if there is a Σ-structure $\mathcal{A}$ which is a model for F, $\mathcal{A} \models F$
- valid if every Σ-structure is a model for F

Note: a formula F is valid if and only if $\neg F$ is unsatisfiable

Formulas F_1, F_2 are:

- F_1 semantically imply F_2, denoted $F_1 \models F_2$, if all models of F_1 are also models of F_2
- semantically equivalent, denoted $F_1 \equiv F_2$, iff F_1 and F_2 have the same models

Validity, satisfiability,

A (closed) formula F is

- satisfiable if there is a Σ-structure $\mathcal{A}$ which is a model for F, $\mathcal{A} \models F$
- valid if every Σ-structure is a model for F

Note: a formula F is valid if and only if $\neg F$ is unsatisfiable

Formulas F_1, F_2 are:

- F_1 semantically imply F_2, denoted $F_1 \models F_2$, if all models of F_1 are also models of F_2
- semantically equivalent, denoted $F_1 \equiv F_2$, iff F_1 and F_2 have the same models

Validity, satisfiability,

A (closed) formula F is

- satisfiable if there is a Σ-structure $\mathcal{A}$ which is a model for F, $\mathcal{A} \models F$
- valid if every Σ-structure is a model for F

Note: a formula F is valid if and only if $\neg F$ is unsatisfiable

Formulas F_1, F_2 are:

- F_1 semantically imply F_2, denoted $F_1 \models F_2$, if all models of F_1 are also models of F_2
- semantically equivalent, denoted $F_1 \equiv F_2$, iff F_1 and F_2 have the same models

First-order theories

A first-order theory is T is a set of first-order formulas closed under implication: if $F \in T$ and $F \models G$ then $G \in T$.

Axioms for T is a set of formulas Ax such that $Ax \subseteq T$ and Ax imply T.

First-order theories

A first-order theory is T is a set of first-order formulas closed under implication: if $F \in T$ and $F \models G$ then $G \in T$.

Axioms for T is a set of formulas Ax such that $Ax \subseteq T$ and Ax imply T.

First-order theories

Consider a first-order theory T and a first-order formula F.

The main reasoning problem is checking whether $T \models F$.

Axioms of groups *Group*: $\Sigma = (\{\cdot/2, {}^{-1}/1, e/0\}, \{\simeq /2\})$:

- $\forall x, y, z \ \ (x \cdot (y \cdot z) \simeq (x \cdot y) \cdot z)$ – associativity
- $\forall x \ \ (x \cdot x^{-1} \simeq e)$ – inverse
- $\forall x \ \ (x \cdot e \simeq x)$ – identity

Consider $F = \forall x, y \ ((x \cdot y)^{-1} \simeq y^{-1} \cdot x^{-1})$

Is F a theorem in the group theory: $Group \models F$?

Axioms of arrays:

- $\forall a, i, e \ (select(store(a, i, e), i) \simeq e)$
- $\forall a, i, j, e \ (i \not\simeq j \rightarrow (select(store(a, i, e), j) \simeq select(a, j)))$
- $\forall a_1, a_2 \ ((\forall i \ (select(a_1, i) \simeq select(a_2, i))) \rightarrow a_1 \simeq a_2)$

Is $\exists a \exists i \forall j \ (select(a, i) \simeq select(a, j))$ a theorem in the theory of arrays ?

First-order theories

Consider a first-order theory T and a first-order formula F.

The main reasoning problem is checking whether $T \models F$.

Axioms of groups *Group*: $\Sigma = (\{\cdot/2, {}^{-1}/1, e/0\}, \{\simeq /2\})$:

- $\forall x, y, z \ \ (x \cdot (y \cdot z) \simeq (x \cdot y) \cdot z)$ – associativity
- $\forall x \ \ (x \cdot x^{-1} \simeq e)$ – inverse
- $\forall x \ \ (x \cdot e \simeq x)$ – identity

Consider $F = \forall x, y \ ((x \cdot y)^{-1} \simeq y^{-1} \cdot x^{-1})$

Is F a theorem in the group theory: $Group \models F$?

Axioms of arrays:

- $\forall a, i, e \ (select(store(a, i, e), i) \simeq e)$
- $\forall a, i, j, e \ (i \not\simeq j \rightarrow (select(store(a, i, e), j) \simeq select(a, j)))$
- $\forall a_1, a_2 \ ((\forall i \ (select(a_1, i) \simeq select(a_2, i))) \rightarrow a_1 \simeq a_2)$

Is $\exists a \exists i \forall j \ (select(a, i) \simeq select(a, j))$ a theorem in the theory of arrays ?

First-order theories

Consider a first-order theory T and a first-order formula F.

The main reasoning problem is checking whether $T \models F$.

Axioms of groups *Group*: $\Sigma = (\{\cdot/2, {}^{-1}/1, e/0\}, \{\simeq /2\})$:

- $\forall x, y, z \ \ (x \cdot (y \cdot z) \simeq (x \cdot y) \cdot z)$ – associativity
- $\forall x \ \ (x \cdot x^{-1} \simeq e)$ – inverse
- $\forall x \ \ (x \cdot e \simeq x)$ – identity

Consider $F = \forall x, y \ ((x \cdot y)^{-1} \simeq y^{-1} \cdot x^{-1})$

Is F a theorem in the group theory: $Group \models F$?

Axioms of arrays:

- $\forall a, i, e \ (select(store(a, i, e), i) \simeq e)$
- $\forall a, i, j, e \ (i \not\simeq j \rightarrow (select(store(a, i, e), j) \simeq select(a, j)))$
- $\forall a_1, a_2 \ ((\forall i \ (select(a_1, i) \simeq select(a_2, i))) \rightarrow a_1 \simeq a_2)$

Is $\exists a \exists i \forall j \ (select(a, i) \simeq select(a, j))$ a theorem in the theory of arrays ?

First-order theories

Consider a first-order theory T and a first-order formula F.

The main reasoning problem is checking whether $T \models F$.

Axioms of groups *Group*: $\Sigma = (\{\cdot/2, ^{-1}/1, e/0\}, \{\simeq /2\})$:

- $\forall x, y, z \ \ (x \cdot (y \cdot z) \simeq (x \cdot y) \cdot z)$ – associativity
- $\forall x \ \ (x \cdot x^{-1} \simeq e)$ – inverse
- $\forall x \ \ (x \cdot e \simeq x)$ – identity

Consider $F = \forall x, y \ ((x \cdot y)^{-1} \simeq y^{-1} \cdot x^{-1})$

Is F a theorem in the group theory: $Group \models F$?

Axioms of arrays:

- $\forall a, i, e \ (select(store(a, i, e), i) \simeq e)$
- $\forall a, i, j, e \ (i \not\simeq j \rightarrow (select(store(a, i, e), j) \simeq select(a, j)))$
- $\forall a_1, a_2 \ ((\forall i \ (select(a_1, i) \simeq select(a_2, i))) \rightarrow a_1 \simeq a_2)$

Is $\exists a \exists i \forall j \ (select(a, i) \simeq select(a, j))$ a theorem in the theory of arrays ?

Deduction

Semantic arguments are usually as hoc, complicated and applicable only to narrow cases.

Deduction: A simple set of syntactic rules to derive theorems.

Why deduction:

- purely syntactic derivations
- can derive any first-order theorem (completeness)
- a universal set of rules which is applicable to any first-order theory
- can be efficiently automated

Deduction

Semantic arguments are usually as hoc, complicated and applicable only to narrow cases.

Deduction: A simple set of syntactic rules to derive theorems.

Why deduction:

- purely syntactic derivations
- can derive any first-order theorem (completeness)
- a universal set of rules which is applicable to any first-order theory
- can be efficiently automated

Calculi for first-order logic

Calculi complete for first-order logic:

- natural deduction
  - difficult to automate
- tableaux-based calculi
  - popular with special fragments: modal and description logics
  - difficult to automate efficiently in the general case
- resolution/superposition calculi
  - general purpose
  - can be efficiently automated
  - decision procedure for many fragments
- instantiation-based calculi
  - combination of efficient ground reasoning with first-order reasoning
  - can be efficiently automated
  - decision procedure for the effectively propositional fragment (EPR)

Calculi for first-order logic

Calculi complete for first-order logic:

- natural deduction
  - difficult to automate
- tableaux-based calculi
  - popular with special fragments: modal and description logics
  - difficult to automate efficiently in the general case
- resolution/superposition calculi
  - general purpose
  - can be efficiently automated
  - decision procedure for many fragments
- instantiation-based calculi
  - combination of efficient ground reasoning with first-order reasoning
  - can be efficiently automated
  - decision procedure for the effectively propositional fragment (EPR)

Calculi for first-order logic

Calculi complete for first-order logic:

- natural deduction
  - difficult to automate
- tableaux-based calculi
  - popular with special fragments: modal and description logics
  - difficult to automate efficiently in the general case
- resolution/superposition calculi
  - general purpose
  - can be efficiently automated
  - decision procedure for many fragments
- instantiation-based calculi
  - combination of efficient ground reasoning with first-order reasoning
  - can be efficiently automated
  - decision procedure for the effectively propositional fragment (EPR)

Calculi for first-order logic

Calculi complete for first-order logic:

- natural deduction
  - difficult to automate
- tableaux-based calculi
  - popular with special fragments: modal and description logics
  - difficult to automate efficiently in the general case
- resolution/superposition calculi
  - general purpose
  - can be efficiently automated
  - decision procedure for many fragments
- instantiation-based calculi
  - combination of efficient ground reasoning with first-order reasoning
  - can be efficiently automated
  - decision procedure for the effectively propositional fragment (EPR)

Refutational reasoning

In reasoning methods we study, the validity problem is reformulated in terms of unsatisfiability. Proof by contradiction.

A is valid iff $\neg A$ is unsatisfiable.

In other words:

$$\models A \text{ iff } \neg A \models \bot$$

Example. The are an infinite number of prime numbers.

Other common problems:

$$\models \text{Axioms} \rightarrow \text{Theorem iff Axioms} \land \neg\text{Theorem} \models \bot$$

$$\models A \leftrightarrow B \text{ iff } A \leftrightarrow \neg B \models \bot$$

Refutational reasoning

In reasoning methods we study, the validity problem is reformulated in terms of unsatisfiability. Proof by contradiction.

A is valid iff $\neg A$ is unsatisfiable.

In other words:

$$\models A \text{ iff } \neg A \models \bot$$

Example. The are an infinite number of prime numbers.

Other common problems:

$$\models \mathrm{Axioms} \rightarrow \mathrm{Theorem} \text{ iff } \mathrm{Axioms} \wedge \neg \mathrm{Theorem} \models \bot$$

$$\models A \leftrightarrow B \text{ iff } A \leftrightarrow \neg B \models \bot$$

Refutational reasoning

In reasoning methods we study, the validity problem is reformulated in terms of unsatisfiability. Proof by contradiction.

$$A \text{ is valid iff } \neg A \text{ is unsatisfiable.}$$

In other words:

$$\models A \text{ iff } \neg A \models \bot$$

Example. The are an infinite number of prime numbers.

Other common problems:

$$\models \mathrm{Axioms} \rightarrow \mathrm{Theorem} \text{ iff } \mathrm{Axioms} \wedge \neg \mathrm{Theorem} \models \bot$$

$$\models A \leftrightarrow B \text{ iff } A \leftrightarrow \neg B \models \bot$$

Refutational reasoning

In reasoning methods we study, the validity problem is reformulated in terms of unsatisfiability. Proof by contradiction.

$$A \text{ is valid iff } \neg A \text{ is unsatisfiable.}$$

In other words:

$$\models A \text{ iff } \neg A \models \bot$$

Example. The are an infinite number of prime numbers.

Other common problems:

$$\models \mathrm{Axioms} \rightarrow \mathrm{Theorem} \text{ iff } \mathrm{Axioms} \wedge \neg \mathrm{Theorem} \models \bot$$

$$\models A \leftrightarrow B \text{ iff } A \leftrightarrow \neg B \models \bot$$

Normal forms: CNF

Normal Forms

For efficient reasoning methods we need to assume that formulas are in a certain simple normal form – conjunctive normal form (CNF).

CNF transformation: Transforms any first-order formula into an equi-satisfiable formula in CNF.

Normal Forms

For efficient reasoning methods we need to assume that formulas are in a certain simple normal form – conjunctive normal form (CNF).

CNF transformation: Transforms any first-order formula into an equi-satisfiable formula in CNF.

Literal, clause

- Literal L: either an atom $p(\bar{t})$ (*positive literal*) or its negation $\neg p(\bar{t})$ (*negative literal*).
- The complementary literal to L:

$$\overline{L} \stackrel{\text{def}}{=} \begin{cases} \neg p(\bar{t}), & \text{if } L \text{ has the form } p(\bar{t}); \\ p(\bar{t}), & \text{if } L \text{ has the form } \neg p(\bar{t}). \end{cases}$$

  In other words, $p(\bar{t})$ and $\neg p(\bar{t})$ are complementary.

- Clause: disjunction of literals

$$L_1 \vee \ldots \vee L_n, \quad n \geq 0.$$

  Variables are implicitly universally quantified.

  A clause can be seen as a mulit-set of literals $\{L_1, \ldots, L_n\}$.

- Empty clause, denoted by $\Box$: $n = 0$

  The empty clause is false in every interpretation.

Literal, clause

- Literal L: either an atom $p(\bar{t})$ (*positive literal*) or its negation $\neg p(\bar{t})$ (*negative literal*).
- The complementary literal to L:

$$\overline{L} \stackrel{\text{def}}{=} \begin{cases} \neg p(\bar{t}), & \text{if } L \text{ has the form } p(\bar{t}); \\ p(\bar{t}), & \text{if } L \text{ has the form } \neg p(\bar{t}). \end{cases}$$

  In other words, $p(\bar{t})$ and $\neg p(\bar{t})$ are complementary.

- Clause: disjunction of literals

$$L_1 \vee \ldots \vee L_n, \quad n \geq 0.$$

  Variables are implicitly universally quantified.

  A clause can be seen as a mulit-set of literals $\{L_1, \ldots, L_n\}$.

- Empty clause, denoted by $\square$: $n = 0$

  The empty clause is false in every interpretation.

Literal, clause

- Literal L: either an atom $p(\bar{t})$ (*positive literal*) or its negation $\neg p(\bar{t})$ (*negative literal*).
- The complementary literal to L:

$$\overline{L} \stackrel{\mathrm{def}}{=} \begin{cases} \neg p(\bar{t}), & \text{if } L \text{ has the form } p(\bar{t}); \\ p(\bar{t}), & \text{if } L \text{ has the form } \neg p(\bar{t}). \end{cases}$$

  In other words, $p(\bar{t})$ and $\neg p(\bar{t})$ are complementary.
- Clause: disjunction of literals

$$L_1 \vee \ldots \vee L_n, \quad n \geq 0.$$

  Variables are implicitly universally quantified.

  A clause can be seen as a mulit-set of literals $\{L_1, \ldots, L_n\}$.
- Empty clause, denoted by $\square$: $n = 0$

  The empty clause is false in every interpretation.

Literal, clause

- Literal L: either an atom $p(\bar{t})$ (*positive literal*) or its negation $\neg p(\bar{t})$ (*negative literal*).
- The complementary literal to L:

$$\overline{L} \stackrel{\text{def}}{=} \begin{cases} \neg p(\bar{t}), & \text{if } L \text{ has the form } p(\bar{t}); \\ p(\bar{t}), & \text{if } L \text{ has the form } \neg p(\bar{t}). \end{cases}$$

  In other words, $p(\bar{t})$ and $\neg p(\bar{t})$ are complementary.

- Clause: disjunction of literals

$$L_1 \vee \ldots \vee L_n, \quad n \geq 0.$$

  Variables are implicitly universally quantified.

  A clause can be seen as a mulit-set of literals $\{L_1, \ldots, L_n\}$.

- Empty clause, denoted by $\square$: $n = 0$

  The empty clause is false in every interpretation.

CNF

- A formula F is in conjunctive normal form, or simply CNF, if it is either $\top$, or $\bot$, or a universally quantified conjunction of clauses:

$$F = \forall \bar{x}\ [\bigwedge_i (\bigvee_j L_{i,j})].$$

Example:

$$\begin{array}{lll} \forall x, y, z\ [& p(x) \lor p(y) \lor \neg q(x, f(y)) & \bigwedge \\ & \neg p(f(z)) \lor q(z, z) & \bigwedge \\ & q(c, f(d)) &] \end{array}$$

Notation: a set of clauses

$$\{p(x) \lor p(y) \lor \neg q(x, f(y)), \neg p(f(z)) \lor q(z, z), q(c, f(d))\}$$

- A set of clauses S is a clausal normal form of a formula F if S is equi-satisfiable with F.

CNF

- A formula F is in conjunctive normal form, or simply CNF, if it is either $\top$, or $\bot$, or a universally quantified conjunction of clauses:

$$F = \forall \bar{x} \; [\bigwedge_i (\bigvee_j L_{i,j})].$$

Example:

$$\begin{array}{lll} \forall x, y, z \;\; [& p(x) \vee p(y) \vee \neg q(x, f(y)) & \bigwedge \\ & \neg p(f(z)) \vee q(z, z) & \bigwedge \\ & q(c, f(d)) &] \end{array}$$

Notation: a set of clauses

$$\{p(x) \vee p(y) \vee \neg q(x, f(y)), \neg p(f(z)) \vee q(z, z), q(c, f(d))\}$$

- A set of clauses S is a clausal normal form of a formula F if S is equi-satisfiable with F.

CNF transformation

Main steps in the basic CNF transformation:

1. Prenex normal form – moving all quantifiers up-front
   $\forall y\ [\forall x\ [p(f(x), y)] \rightarrow \forall v \exists z\ [q(f(z)) \wedge p(v, z)]] \Rightarrow$
   $\forall y \exists x \forall v \exists z\ [p(f(x), y) \rightarrow (q(f(z)) \wedge p(v, z))]$
2. Skolemization – eliminating existential quantifiers
   $\forall y \exists x \forall v \exists z\ [p(f(x), y) \rightarrow (q(f(z)) \wedge p(v, z))] \Rightarrow$
   $\forall y \forall v\ [p(f(sk_1(y)), y) \rightarrow (q(f(sk_2(y, v))) \wedge p(v, sk_2(y, v)))]$
3. CNF transformation of the quantifier-free part
   $\forall y \forall v\ [\quad p(f(sk_1(y)), y) \rightarrow (q(f(sk_2(y, v))) \wedge p(v, sk_2(y, v)))] \Rightarrow$
   $\forall y \forall v\ [\quad (\neg p(f(sk_1(y)), y) \vee q(f(sk_2(y, v)))) \wedge$
   $(\neg p(f(sk_1(y)), y) \vee p(v, sk_2(y, v)))]$

CNF transformation

Main steps in the basic CNF transformation:

1. Prenex normal form – moving all quantifiers up-front
   $\forall y\ [\forall x\ [p(f(x), y)] \rightarrow \forall v \exists z\ [q(f(z)) \wedge p(v, z)]] \Rightarrow$
   $\forall y \exists x \forall v \exists z\ [p(f(x), y) \rightarrow (q(f(z)) \wedge p(v, z))]$
2. Skolemization – eliminating existential quantifiers
   $\forall y \exists x \forall v \exists z\ [p(f(x), y) \rightarrow (q(f(z)) \wedge p(v, z))] \Rightarrow$
   $\forall y \forall v\ [p(f(sk_1(y)), y) \rightarrow (q(f(sk_2(y, v))) \wedge p(v, sk_2(y, v)))]$
3. CNF transformation of the quantifier-free part
   $\forall y \forall v\ [\quad p(f(sk_1(y)), y) \rightarrow (q(f(sk_2(y, v))) \wedge p(v, sk_2(y, v)))] \Rightarrow$
   $\forall y \forall v\ [\quad (\neg p(f(sk_1(y)), y) \vee q(f(sk_2(y, v)))) \wedge$
   $(\neg p(f(sk_1(y)), y) \vee p(v, sk_2(y, v)))]$

CNF transformation

Main steps in the basic CNF transformation:

1. Prenex normal form – moving all quantifiers up-front

   $\forall y\ [\forall x\ [p(f(x), y)] \rightarrow \forall v \exists z\ [q(f(z)) \wedge p(v, z)]] \Rightarrow$

   $\forall y \exists x \forall v \exists z\ [p(f(x), y) \rightarrow (q(f(z)) \wedge p(v, z))]$

2. Skolemization – eliminating existential quantifiers

   $\forall y \exists x \forall v \exists z\ [p(f(x), y) \rightarrow (q(f(z)) \wedge p(v, z))] \Rightarrow$

   $\forall y \forall v\ [p(f(sk_1(y)), y) \rightarrow (q(f(sk_2(y, v))) \wedge p(v, sk_2(y, v)))]$

3. CNF transformation of the quantifier-free part

   $\forall y \forall v\ [\quad p(f(sk_1(y)), y) \rightarrow (q(f(sk_2(y, v))) \wedge p(v, sk_2(y, v)))] \Rightarrow$

   $\forall y \forall v\ [\quad (\neg p(f(sk_1(y)), y) \vee q(f(sk_2(y, v)))) \wedge$

   $\qquad (\neg p(f(sk_1(y)), y) \vee p(v, sk_2(y, v)))]$

Prenex normal form

Prenex normal form – moving all quantifiers up-front.

Assume that the formula is rectified and

$F \leftrightarrow G$ is replaced by $(F \rightarrow G) \wedge (G \rightarrow F)$.

$$
\begin{array}{rcl}
\neg(\forall x F) & \Rightarrow_{\mathrm{PNF}} & \exists x \neg F \\
\neg(\exists x F) & \Rightarrow_{\mathrm{PNF}} & \forall x \neg F \\
(\exists\!\!\!\forall x F) \wedge\!\!\!\vee G & \Rightarrow_{\mathrm{PNF}} & \exists\!\!\!\forall x (F \wedge\!\!\!\vee G) \\
(\exists x F) \rightarrow G & \Rightarrow_{\mathrm{PNF}} & \forall x (F \rightarrow G) \\
(\forall x F) \rightarrow G & \Rightarrow_{\mathrm{PNF}} & \exists x (F \rightarrow G)
\end{array}
$$

Example:

$$
\forall y \; [\forall x \; [p(f(x), y)] \rightarrow \forall v \exists z \; [q(f(z)) \wedge p(v, z)]] \Rightarrow^{*}_{\mathrm{PNF}}
$$
$$
\forall y \exists x \forall v \exists z \; [p(f(x), y) \rightarrow (q(f(z)) \wedge p(v, z))]
$$

Prenex normal form

Prenex normal form – moving all quantifiers up-front.

Assume that the formula is rectified and

$F \leftrightarrow G$ is replaced by $(F \rightarrow G) \wedge (G \rightarrow F)$.

$$
\begin{aligned}
\neg(\forall x F) &\Rightarrow_{\mathrm{PNF}} \exists x \neg F \\
\neg(\exists x F) &\Rightarrow_{\mathrm{PNF}} \forall x \neg F \\
(\exists\!\!\!\forall x F) \wedge\!\!\!\!\vee G &\Rightarrow_{\mathrm{PNF}} \exists\!\!\!\forall x (F \wedge\!\!\!\!\vee G) \\
(\exists x F) \rightarrow G &\Rightarrow_{\mathrm{PNF}} \forall x (F \rightarrow G) \\
(\forall x F) \rightarrow G &\Rightarrow_{\mathrm{PNF}} \exists x (F \rightarrow G)
\end{aligned}
$$

Example:

$$
\forall y\ [\forall x\ [p(f(x), y)] \rightarrow \forall v \exists z\ [q(f(z)) \wedge p(v, z)]] \Rightarrow^{*}_{\mathrm{PNF}}
$$
$$
\forall y \exists x \forall v \exists z\ [p(f(x), y) \rightarrow (q(f(z)) \wedge p(v, z))]
$$

Skolemization

Skolemization – eliminating existential quantifiers.

$$F = \forall \bar{x} \exists y \; F'(\bar{x}, y)$$

Informally:

- F states that for each value of $\bar{x}$ we can choose a value for y such that $F'(\bar{x}, y)$ holds.
- We can represent this choice by a Skolem function $sk_{F'}(\bar{x})$.
- $\forall \bar{x} \exists y \; F'(\bar{x}, y)$ is equi-satisfiable with $\forall \bar{x} \; F'(\bar{x}, sk_{F'}(\bar{x}))$.

Example:

$\forall y \exists x \forall v \exists z \; [p(f(x), y) \rightarrow (q(f(z)) \wedge p(v, z))] \Rightarrow_{\mathrm{SK}}$

$\forall y \forall v \; [p(f(sk_1(y)), y) \rightarrow (q(f(sk_2(y, v))) \wedge p(v, sk_2(y, v)))]$

Skolemization

Skolemization – eliminating existential quantifiers.

$$F = \forall \bar{x} \exists y \; F'(\bar{x}, y)$$

Informally:

- F states that for each value of $\bar{x}$ we can choose a value for y such that $F'(\bar{x}, y)$ holds.
- We can represent this choice by a Skolem function $sk_{F'}(\bar{x})$.
- $\forall \bar{x} \exists y \; F'(\bar{x}, y)$ is equi-satisfiable with $\forall \bar{x} \; F'(\bar{x}, sk_{F'}(\bar{x}))$.

Example:

$\forall y \exists x \forall v \exists z \; [p(f(x), y) \rightarrow (q(f(z)) \wedge p(v, z))] \Rightarrow_{\mathrm{SK}}$
$\forall y \forall v \; [p(f(sk_1(y)), y) \rightarrow (q(f(sk_2(y, v))) \wedge p(v, sk_2(y, v)))]$

CNF transformation

CNF transformation of the quantifier-fee part:

$$
\begin{aligned}
F \leftrightarrow G &\Rightarrow_{\mathrm{CNF}} (F \rightarrow G) \wedge (G \rightarrow F) \\
F \rightarrow G &\Rightarrow_{\mathrm{CNF}} (\neg F \vee G) \\
\neg(F \vee G) &\Rightarrow_{\mathrm{CNF}} (\neg F \wedge \neg G) \\
\neg(F \wedge G) &\Rightarrow_{\mathrm{CNF}} (\neg F \vee \neg G) \\
\neg\neg F &\Rightarrow_{\mathrm{CNF}} F \\
(F \wedge G) \vee H &\Rightarrow_{\mathrm{CNF}} (F \vee H) \wedge (G \vee H)
\end{aligned}
$$

Clausal normal form

$$\begin{array}{lll} F & \Rightarrow^*_{\mathrm{PNF}} & \exists\!\!\!\forall x_1 \ldots \exists\!\!\!\forall x_n\ F' \\ & \Rightarrow^*_{\mathrm{SK}} & \forall \bar{x}\ F'' \\ & \Rightarrow^*_{\mathrm{CNF}} & \forall \bar{x}\ [\bigwedge_i (\bigvee_j L_{i,j})] \\ & \Rightarrow & \{C_1, \ldots, C_n\} \end{array}$$

Note: all variables in $C_1, \ldots, C_n$ are implicitly universally quantified.

Problems with the basic transformation:

- exponential in size
- the structure of the formula can be lost
- Skolem functions can include many irrelevant arguments

Clausal normal form

$$
\begin{array}{lll}
F & \Rightarrow^*_{\mathrm{PNF}} & \exists\!\!\!\forall x_1 \ldots \exists\!\!\!\forall x_n \; F' \\
 & \Rightarrow^*_{\mathrm{SK}} & \forall \bar{x} \; F'' \\
 & \Rightarrow^*_{\mathrm{CNF}} & \forall \bar{x} \; [\bigwedge_i (\bigvee_j L_{i,j})] \\
 & \Rightarrow & \{C_1, \ldots, C_n\}
\end{array}
$$

Note: all variables in $C_1, \ldots, C_n$ are implicitly universally quantified.

Problems with the basic transformation:

- exponential in size
- the structure of the formula can be lost
- Skolem functions can include many irrelevant arguments

Clausal normal form

$$
\begin{array}{lll}
F & \Rightarrow^*_{\mathrm{PNF}} & \exists\!\!\!\forall x_1 \ldots \exists\!\!\!\forall x_n \; F' \\
& \Rightarrow^*_{\mathrm{SK}} & \forall \bar{x} \; F'' \\
& \Rightarrow^*_{\mathrm{CNF}} & \forall \bar{x} \; [\bigwedge_i (\bigvee_j L_{i,j})] \\
& \Rightarrow & \{C_1, \ldots, C_n\}
\end{array}
$$

Note: all variables in $C_1, \ldots, C_n$ are implicitly universally quantified.

Problems with the basic transformation:

- exponential in size
- the structure of the formula can be lost
- Skolem functions can include many irrelevant arguments

Optimized CNF transformation

Optimized: do the opposite to the basic transformation.

- structural transformation: introduce names for complex sub-formulas
  - $F[G(\bar{x})]$ equi-satisfiable with $F[p_G(\bar{x})] \wedge \forall \bar{x}(p_G(\bar{x}) \leftrightarrow G(\bar{x}))$ where p_G is a fresh predicate name.
  - using naming we can obtain a linear-size CNF
- structural transformation: optimizations
  - if $G(\bar{x})$ occurs only positively then we need only one side of the definition: $\forall \bar{x}(p_G(\bar{x}) \rightarrow G(\bar{x}))$ (similar for negatively)
  - reuse names, combine with preprocessing
- miniscoping: push quantifiers inwards

  Reduces arguments of Skolem functions: $\forall x, y \exists z(p(z) \vee q(x, y))$

basic:	$p(sk(x, y)) \vee q(x, y)$	non-EPR
miniscoping:	$p(sk) \vee q(x, y)$	EPR

[Nonnengart, Weidenbach, AR'01; Hoder, Khasidashvili, Korovin, Voronkov FMCAD'12]

Optimized CNF transformation

Optimized: do the opposite to the basic transformation.

- structural transformation: introduce names for complex sub-formulas
  - $F[G(\bar{x})]$ equi-satisfiable with $F[p_G(\bar{x})] \wedge \forall \bar{x}(p_G(\bar{x}) \leftrightarrow G(\bar{x}))$ where p_G is a fresh predicate name.
  - using naming we can obtain a linear-size CNF
- structural transformation: optimizations
  - if $G(\bar{x})$ occurs only positively then we need only one side of the definition: $\forall \bar{x}(p_G(\bar{x}) \rightarrow G(\bar{x}))$ (similar for negatively)
  - reuse names, combine with preprocessing
- miniscoping: push quantifiers inwards

  Reduces arguments of Skolem functions: $\forall x, y \exists z(p(z) \vee q(x, y))$

basic:	$p(sk(x, y)) \vee q(x, y)$	non-EPR
miniscoping:	$p(sk) \vee q(x, y)$	EPR

[Nonnengart, Weidenbach, AR'01; Hoder, Khasidashvili, Korovin, Voronkov, FMCAD'12]

Optimized CNF transformation

Optimized: do the opposite to the basic transformation.

- structural transformation: introduce names for complex sub-formulas
  - $F[G(\bar{x})]$ equi-satisfiable with $F[p_G(\bar{x})] \wedge \forall\bar{x}(p_G(\bar{x}) \leftrightarrow G(\bar{x}))$ where p_G is a fresh predicate name.
  - using naming we can obtain a linear-size CNF
- structural transformation: optimizations
  - if $G(\bar{x})$ occurs only positively then we need only one side of the definition: $\forall\bar{x}(p_G(\bar{x}) \rightarrow G(\bar{x}))$ (similar for negatively)
  - reuse names, combine with preprocessing
- miniscoping: push quantifiers inwards

  Reduces arguments of Skolem functions: $\forall x, y \exists z(p(z) \vee q(x, y))$

basic:	$p(sk(x, y)) \vee q(x, y)$	non-EPR
miniscoping:	$p(sk) \vee q(x, y)$	EPR

[Nonnengart, Weidenbach, AR'01; Hoder, Khasidashvili, Korovin, Voronkov, FMCAD'12]

Optimized CNF transformation

Optimized: do the opposite to the basic transformation.

- structural transformation: introduce names for complex sub-formulas
  - $F[G(\bar{x})]$ equi-satisfiable with $F[p_G(\bar{x})] \wedge \forall \bar{x}(p_G(\bar{x}) \leftrightarrow G(\bar{x}))$ where p_G is a fresh predicate name.
  - using naming we can obtain a linear-size CNF
- structural transformation: optimizations
  - if $G(\bar{x})$ occurs only positively then we need only one side of the definition: $\forall \bar{x}(p_G(\bar{x}) \rightarrow G(\bar{x}))$ (similar for negatively)
  - reuse names, combine with preprocessing
- miniscoping: push quantifiers inwards

  Reduces arguments of Skolem functions: $\forall x, y \exists z(p(z) \vee q(x, y))$

basic:	$p(sk(x, y)) \vee q(x, y)$	non-EPR
miniscoping:	$p(sk) \vee q(x, y)$	EPR

[Nonnengart, Weidenbach, AR'01; Hoder, Khasidashvili, Korovin, Voronkov, FMCAD'12]

Optimized CNF transformation

Optimized: do the opposite to the basic transformation.

- structural transformation: introduce names for complex sub-formulas
  - $F[G(\bar{x})]$ equi-satisfiable with $F[p_G(\bar{x})] \land \forall\bar{x}(p_G(\bar{x}) \leftrightarrow G(\bar{x}))$ where p_G is a fresh predicate name.
  - using naming we can obtain a linear-size CNF
- structural transformation: optimizations
  - if $G(\bar{x})$ occurs only positively then we need only one side of the definition: $\forall\bar{x}(p_G(\bar{x}) \rightarrow G(\bar{x}))$ (similar for negatively)
  - reuse names, combine with preprocessing
- miniscoping: push quantifiers inwards

  Reduces arguments of Skolem functions: $\forall x, y \exists z(p(z) \lor q(x, y))$

basic:	$p(sk(x, y)) \lor q(x, y)$	non-EPR
miniscoping:	$p(sk) \lor q(x, y)$	EPR

[Nonnengart, Weidenbach, AR'01; Hoder, Khasidashvili, Korovin, Voronkov, FMCAD'12]

Optimized CNF transformation

Optimized: do the opposite to the basic transformation.

- structural transformation: introduce names for complex sub-formulas
  - $F[G(\bar{x})]$ equi-satisfiable with $F[p_G(\bar{x})] \land \forall \bar{x}(p_G(\bar{x}) \leftrightarrow G(\bar{x}))$ where p_G is a fresh predicate name.
  - using naming we can obtain a linear-size CNF
- structural transformation: optimizations
  - if $G(\bar{x})$ occurs only positively then we need only one side of the definition: $\forall \bar{x}(p_G(\bar{x}) \rightarrow G(\bar{x}))$ (similar for negatively)
  - reuse names, combine with preprocessing
- miniscoping: push quantifiers inwards

  Reduces arguments of Skolem functions: $\forall x, y \exists z (p(z) \lor q(x, y))$

basic:	$p(sk(x, y)) \lor q(x, y)$	non-EPR
miniscoping:	$p(sk) \lor q(x, y)$	EPR

[Nonnengart, Weidenbach, AR'01; Hoder, Khasidashvili, Korovin, Voronkov, FMCAD'12]

Optimized CNF transformation

Optimized: do the opposite to the basic transformation.

- structural transformation: introduce names for complex sub-formulas
  - $F[G(\bar{x})]$ equi-satisfiable with $F[p_G(\bar{x})] \wedge \forall \bar{x}(p_G(\bar{x}) \leftrightarrow G(\bar{x}))$ where p_G is a fresh predicate name.
  - using naming we can obtain a linear-size CNF
- structural transformation: optimizations
  - if $G(\bar{x})$ occurs only positively then we need only one side of the definition: $\forall \bar{x}(p_G(\bar{x}) \rightarrow G(\bar{x}))$ (similar for negatively)
  - reuse names, combine with preprocessing
- miniscoping: push quantifiers inwards

  Reduces arguments of Skolem functions: $\forall x, y \exists z(p(z) \vee q(x, y))$

basic:	$p(sk(x, y)) \vee q(x, y)$	non-EPR
miniscoping:	$p(sk) \vee q(x, y)$	EPR

[Nonnengart, Weidenbach, AR'01; Hoder, Khasidashvili, Korovin, Voronkov, FMCAD'12]

Optimized CNF transformation

Optimized: do the opposite to the basic transformation.

- structural transformation: introduce names for complex sub-formulas
  - $F[G(\bar{x})]$ equi-satisfiable with $F[p_G(\bar{x})] \wedge \forall \bar{x}(p_G(\bar{x}) \leftrightarrow G(\bar{x}))$ where p_G is a fresh predicate name.
  - using naming we can obtain a linear-size CNF
- structural transformation: optimizations
  - if $G(\bar{x})$ occurs only positively then we need only one side of the definition: $\forall \bar{x}(p_G(\bar{x}) \rightarrow G(\bar{x}))$ (similar for negatively)
  - reuse names, combine with preprocessing
- miniscoping: push quantifiers inwards

  Reduces arguments of Skolem functions: $\forall x, y \exists z(p(z) \vee q(x, y))$

basic:	$p(sk(x, y)) \vee q(x, y)$	non-EPR
miniscoping:	$p(sk) \vee q(x, y)$	EPR

[Nonnengart, Weidenbach, AR'01; Hoder, Khasidashvili, Korovin, Voronkov, FMCAD'12]

Herbrand interpretations

Herbrand interpretations

Basic idea: In order to check of satisfiability of (universal) formulas it is sufficient to consider only specific class of interpretations called Herbrand interpretations.

Consider a signature $\Sigma = (\mathcal{F}, \mathcal{P})$, we assume that $\mathcal{F}$ contains at least one constant in $\mathcal{F}$.

Key ingredient – ground terms.

Ground terms – terms without occurrences of variables e.g. $f(f(a, b), a)$.
The set of ground terms is $T(\Sigma, \emptyset)$.
Ground atoms, clauses are ... without occurrences of variables.
Grounding substitution is a substitution with the range in ground terms.

Herbrand interpretations

Basic idea: In order to check of satisfiability of (universal) formulas it is sufficient to consider only specific class of interpretations called Herbrand interpretations.

Consider a signature $\Sigma = (\mathcal{F}, \mathcal{P})$, we assume that $\mathcal{F}$ contains at least one constant in $\mathcal{F}$.

Key ingredient – ground terms.

Ground terms – terms without occurrences of variables e.g. $f(f(a, b), a)$.
The set of ground terms is $T(\Sigma, \emptyset)$.

Ground atoms, clauses are ... without occurrences of variables.

Grounding substitution is a substitution with the range in ground terms.

Herbrand interpretations

Basic idea: In order to check of satisfiability of (universal) formulas it is sufficient to consider only specific class of interpretations called Herbrand interpretations.

Consider a signature $\Sigma = (\mathcal{F}, \mathcal{P})$, we assume that $\mathcal{F}$ contains at least one constant in $\mathcal{F}$.

Key ingredient – ground terms.

Ground terms – terms without occurrences of variables e.g. $f(f(a, b), a)$.
The set of ground terms is $T(\Sigma, \emptyset)$.
Ground atoms, clauses are ... without occurrences of variables.
Grounding substitution is a substitution with the range in ground terms.

Herbrand interpretations

A Herbrand Σ-interpretation $\mathcal{H} = (H, \mathcal{F}^{\mathcal{H}}, \mathcal{P}^{\mathcal{H}})$ is a Σ-structure such that

- $H = T(\Sigma, \emptyset)$ –the domain is the set of all ground terms
- $f^{\mathcal{H}}(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$ – terms are interpreted by themselves

Note: the domain and the interpretation of functions are fixed, only interpretations of predicates can vary.

Example: Consider $\Sigma = (\{s/1, 0/0\}, \{p/2\})$ possible Herbrand Σ-interpretations $\mathcal{H}_1, \mathcal{H}_2$:

- $0 \in p^{\mathcal{H}_1}, s(s(0)) \in p^{\mathcal{H}_1}, \ldots, s^{2n}(0) \in p^{\mathcal{H}_1}, \ldots$.
- $p^{\mathcal{H}_2} = \emptyset$

Q: How many Herbrand interpretations over Σ exist?

We can specify any Herbrand interpretation uniquely by specifying which ground atoms are true in it.

Notation: $\mathcal{H}_1 = \{p(0), p(s(s(0))), \ldots, p(s^{2n}(0)), \ldots\}$.

Herbrand interpretations

A Herbrand Σ-interpretation $\mathcal{H} = (H, \mathcal{F}^{\mathcal{H}}, \mathcal{P}^{\mathcal{H}})$ is a Σ-structure such that

- $H = T(\Sigma, \emptyset)$ –the domain is the set of all ground terms
- $f^{\mathcal{H}}(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$ – terms are interpreted by themselves

Note: the domain and the interpretation of functions are fixed, only interpretations of predicates can vary.

Example: Consider $\Sigma = (\{s/1, 0/0\}, \{p/2\})$ possible Herbrand Σ-interpretations $\mathcal{H}_1, \mathcal{H}_2$:

- $0 \in p^{\mathcal{H}_1}, s(s(0)) \in p^{\mathcal{H}_1}, \ldots, s^{2n}(0) \in p^{\mathcal{H}_1}, \ldots$.
- $p^{\mathcal{H}_2} = \emptyset$

Q: How many Herbrand interpretations over Σ exist?

We can specify any Herbrand interpretation uniquely by specifying which ground atoms are true in it.

Notation: $\mathcal{H}_1 = \{p(0), p(s(s(0))), \ldots, p(s^{2n}(0)), \ldots\}$.

Herbrand interpretations

A Herbrand Σ-interpretation $\mathcal{H} = (H, \mathcal{F}^{\mathcal{H}}, \mathcal{P}^{\mathcal{H}})$ is a Σ-structure such that

- $H = T(\Sigma, \emptyset)$ –the domain is the set of all ground terms
- $f^{\mathcal{H}}(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$ – terms are interpreted by themselves

Note: the domain and the interpretation of functions are fixed, only interpretations of predicates can vary.

Example: Consider $\Sigma = (\{s/1, 0/0\}, \{p/2\})$ possible Herbrand Σ-interpretations $\mathcal{H}_1, \mathcal{H}_2$:

- $0 \in p^{\mathcal{H}_1}, s(s(0)) \in p^{\mathcal{H}_1}, \ldots, s^{2n}(0) \in p^{\mathcal{H}_1}, \ldots$.
- $p^{\mathcal{H}_2} = \emptyset$

Q: How many Herbrand interpretations over Σ exist?

We can specify any Herbrand interpretation uniquely by specifying which ground atoms are true in it.

Notation: $\mathcal{H}_1 = \{p(0), p(s(s(0))), \ldots, p(s^{2n}(0)), \ldots\}$.

Herbrand interpretations

A Herbrand Σ-interpretation $\mathcal{H} = (H, \mathcal{F}^{\mathcal{H}}, \mathcal{P}^{\mathcal{H}})$ is a Σ-structure such that

- $H = T(\Sigma, \emptyset)$ –the domain is the set of all ground terms
- $f^{\mathcal{H}}(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$ – terms are interpreted by themselves

Note: the domain and the interpretation of functions are fixed, only interpretations of predicates can vary.

Example: Consider $\Sigma = (\{s/1, 0/0\}, \{p/2\})$ possible Herbrand Σ-interpretations $\mathcal{H}_1, \mathcal{H}_2$:

- $0 \in p^{\mathcal{H}_1}, s(s(0)) \in p^{\mathcal{H}_1}, \ldots, s^{2n}(0) \in p^{\mathcal{H}_1}, \ldots$.
- $p^{\mathcal{H}_2} = \emptyset$

Q: How many Herbrand interpretations over Σ exist?

We can specify any Herbrand interpretation uniquely by specifying which ground atoms are true in it.

Notation: $\mathcal{H}_1 = \{p(0), p(s(s(0))), \ldots, p(s^{2n}(0)), \ldots\}$.

Herbrand interpretations suffice

Theorem. Consider a universally quantified formula F over Σ.

Then F is satisfiable if and only if F has a Herbrand model.

Proof. $\Leftarrow$) Obvious.

$\Rightarrow$) Let $F = \forall x_1, \ldots, x_n F'(\bar{x})$ where $F'(\bar{x})$ is quantifier-free.

Consider $\mathcal{A}$ such that $\mathcal{A} \models \forall x_1, \ldots, x_n F'(\bar{x})$.

Then for any $t_1, \ldots, t_n \in \mathcal{T}(\Sigma, \emptyset)$ we have $\mathcal{A} \models F'[t_1^{\mathcal{A}}, \ldots, t_n^{\mathcal{A}}]$.

Define a Herbrand interpretation $\mathcal{H}$ as follows

$$\mathcal{H} = \{p(\bar{t}) \mid \mathcal{A} \models p[\bar{t}^{\mathcal{A}}], \text{ where } p \in \mathcal{P}, \bar{t} \in \mathcal{T}(\Sigma, \emptyset)\}$$

The domain of $\mathcal{H}$ is $\mathcal{T}(\Sigma, \emptyset)$, hence to show that $\mathcal{H} \models \forall x_1, \ldots, x_n F'(\bar{x})$ it is suffices to show that for any terms $t_1, \ldots, t_n \in \mathcal{T}(\Sigma, \emptyset)$, $\mathcal{H} \models F'[t_1, \ldots, t_n]$. This holds since $\mathcal{H} \models F'[t_1, \ldots, t_n]$ iff $\mathcal{A} \models F'[t_1^{\mathcal{A}}, \ldots, t_n^{\mathcal{A}}]$ by construction.

Herbrand interpretations suffice

Theorem. Consider a universally quantified formula F over Σ.

Then F is satisfiable if and only if F has a Herbrand model.

Proof. $\Leftarrow$) Obvious.

$\Rightarrow$) Let $F = \forall x_1, \ldots, x_n F'(\bar{x})$ where $F'(\bar{x})$ is quantifier-free.

Consider $\mathcal{A}$ such that $\mathcal{A} \models \forall x_1, \ldots, x_n F'(\bar{x})$.

Then for any $t_1, \ldots, t_n \in T(\Sigma, \emptyset)$ we have $\mathcal{A} \models F'[t_1^{\mathcal{A}}, \ldots, t_n^{\mathcal{A}}]$.

Define a Herbrand interpretation $\mathcal{H}$ as follows

$$\mathcal{H} = \{p(\bar{t}) \mid \mathcal{A} \models p[\bar{t}^{\mathcal{A}}], \text{ where } p \in \mathcal{P}, \bar{t} \in T(\Sigma, \emptyset)\}.$$

The domain of $\mathcal{H}$ is $T(\Sigma, \emptyset)$, hence to show that $\mathcal{H} \models \forall x_1, \ldots, x_n F'(\bar{x})$ it is suffices to show that for any terms $t_1, \ldots, t_n \in T(\Sigma, \emptyset), \mathcal{H} \models F'[t_1, \ldots, t_n]$. This holds since $\mathcal{H} \models F'[t_1, \ldots, t_n]$ iff $\mathcal{A} \models F'[t_1^{\mathcal{A}}, \ldots, t_n^{\mathcal{A}}]$ by construction.

Herbrand interpretations suffice

Theorem. Consider a universally quantified formula F over Σ.

Then F is satisfiable if and only if F has a Herbrand model.

Proof. $\Leftarrow$) Obvious.

$\Rightarrow$) Let $F = \forall x_1, \dots, x_n F'(\bar{x})$ where $F'(\bar{x})$ is quantifier-free.

Consider $\mathcal{A}$ such that $\mathcal{A} \models \forall x_1, \dots, x_n F'(\bar{x})$.

Then for any $t_1, \dots, t_n \in T(\Sigma, \emptyset)$ we have $\mathcal{A} \models F'[t_1^{\mathcal{A}}, \dots, t_n^{\mathcal{A}}]$.

Define a Herbrand interpretation $\mathcal{H}$ as follows

$$\mathcal{H} = \{p(\bar{t}) \mid \mathcal{A} \models p[\bar{t}^{\mathcal{A}}], \text{ where } p \in \mathcal{P}, \bar{t} \in T(\Sigma, \emptyset)\}.$$

The domain of $\mathcal{H}$ is $T(\Sigma, \emptyset)$, hence to show that $\mathcal{H} \models \forall x_1, \dots, x_n F'(\bar{x})$ it is suffices to show that for any terms $t_1, \dots, t_n \in T(\Sigma, \emptyset), \mathcal{H} \models F'[t_1, \dots, t_n]$. This holds since $\mathcal{H} \models F'[t_1, \dots, t_n]$ iff $\mathcal{A} \models F'[t_1^{\mathcal{A}}, \dots, t_n^{\mathcal{A}}]$ by construction.

Herbrand interpretations suffice

Theorem. Consider a universally quantified formula F over Σ.

Then F is satisfiable if and only if F has a Herbrand model.

Proof. $\Leftarrow$) Obvious.

$\Rightarrow$) Let $F = \forall x_1, \ldots, x_n F'(\bar{x})$ where $F'(\bar{x})$ is quantifier-free.

Consider $\mathcal{A}$ such that $\mathcal{A} \models \forall x_1, \ldots, x_n F'(\bar{x})$.

Then for any $t_1, \ldots, t_n \in T(\Sigma, \emptyset)$ we have $\mathcal{A} \models F'[t_1^{\mathcal{A}}, \ldots, t_n^{\mathcal{A}}]$.

Define a Herbrand interpretation $\mathcal{H}$ as follows

$$\mathcal{H} = \{p(\bar{t}) \mid \mathcal{A} \models p[\bar{t}^{\mathcal{A}}],\ \text{where } p \in \mathcal{P}, \bar{t} \in T(\Sigma, \emptyset)\}.$$

The domain of $\mathcal{H}$ is $T(\Sigma, \emptyset)$, hence to show that $\mathcal{H} \models \forall x_1, \ldots, x_n F'(\bar{x})$ it is suffices to show that for any terms $t_1, \ldots, t_n \in T(\Sigma, \emptyset), \mathcal{H} \models F'[t_1, \ldots, t_n]$. This holds since $\mathcal{H} \models F'[t_1, \ldots, t_n]$ iff $\mathcal{A} \models F'[t_1^{\mathcal{A}}, \ldots, t_n^{\mathcal{A}}]$ by construction.

Grounding

Consider a universally quantified formula:

$F = \forall x_1, \ldots, x_n F'(x_1, \ldots, x_n)$ where $F'(x_1, \ldots, x_n)$ is quantifier-free.

A ground instance of F' (ambiguously also of F) is a ground formula $F'\sigma$ where σ is a grounding substitution.

Denote the set of all ground instances of F' as

$$Gr(F') = \{F'\sigma \mid \sigma \text{ is a grounding substitution}\}$$

For a set of formulas Φ, $Gr(\Phi) = \{Gr(F) \mid F \in \Phi\}$

For clauses and set of clauses definitions of ground instances are similar.

Example: Consider a signature $\Sigma = (\{f/1, a/0\}, \{p/1\})$.

Ground instances of $p(x) \vee \neg p(f(x))$ consist of:

$p(a) \vee \neg p(f(a)), \;\; p(f(a)) \vee \neg p(f(f(a))), \ldots, p(f^n(a)) \vee \neg p(f^{n+1}(a))), \ldots$

Grounding

Consider a universally quantified formula:

$F = \forall x_1, \ldots, x_n F'(x_1, \ldots, x_n)$ where $F'(x_1, \ldots, x_n)$ is quantifier-free.

A ground instance of F' (ambiguously also of F) is a ground formula $F'\sigma$ where σ is a grounding substitution.

Denote the set of all ground instances of F' as

$$Gr(F') = \{F'\sigma \mid \sigma \text{ is a grounding substitution}\}$$

For a set of formulas Φ, $Gr(\Phi) = \{Gr(F) \mid F \in \Phi\}$

For clauses and set of clauses definitions of ground instances are similar.

Example: Consider a signature $\Sigma = (\{f/1, a/0\}, \{p/1\})$.

Ground instances of $p(x) \vee \neg p(f(x))$ consist of:

$$p(a) \vee \neg p(f(a)), \quad p(f(a)) \vee \neg p(f(f(a))), \ldots, p(f^n(a)) \vee \neg p(f^{n+1}(a))), \ldots$$

Grounding

Consider a universally quantified formula:

$F = \forall x_1, \ldots, x_n F'(x_1, \ldots, x_n)$ where $F'(x_1, \ldots, x_n)$ is quantifier-free.

A ground instance of F' (ambiguously also of F) is a ground formula $F'\sigma$ where σ is a grounding substitution.

Denote the set of all ground instances of F' as

$$Gr(F') = \{F'\sigma \mid \sigma \text{ is a grounding substitution}\}$$

For a set of formulas Φ, $Gr(\Phi) = \{Gr(F) \mid F \in \Phi\}$

For clauses and set of clauses definitions of ground instances are similar.

Example: Consider a signature $\Sigma = (\{f/1, a/0\}, \{p/1\})$.

Ground instances of $p(x) \vee \neg p(f(x))$ consist of:

$p(a) \vee \neg p(f(a)), \quad p(f(a)) \vee \neg p(f(f(a))), \ldots, p(f^n(a)) \vee \neg p(f^{n+1}(a))), \ldots$

Reduction of first-order to ground

Theorem. A set of first-order universal formulas Φ is satisfiable if and only the set of its ground instances $Gr(\Phi)$ is satisfiable.

Proof. $\Rightarrow$) is trivial.

$\Leftarrow$) Assume $Gr(\Phi)$ is satisfiable. Then there is a Herbrand model $\mathcal{H} \models Gr(\Phi)$. Since the domain of $\mathcal{H}$ is exactly all ground terms, $\mathcal{H} \models \Phi$.

Reduction of first-order to ground

Theorem. A set of first-order universal formulas Φ is satisfiable
if and only the set of its ground instances $Gr(\Phi)$ is satisfiable.

Proof. $\Rightarrow$) is trivial.

$\Leftarrow$) Assume $Gr(\Phi)$ is satisfiable. Then there is a Herbrand model $\mathcal{H} \models Gr(\Phi)$. Since the domain of $\mathcal{H}$ is exactly all ground terms, $\mathcal{H} \models \Phi$.

Reduction first-order to propositional

Ground formulas can be seen as propositional formulas as follows:

Consider a ground formula F.

- With each ground atom A in F associate a propositional variable x_A.
- Let $Prop(F)$ be a propositional formula obtained from F by replacing all atoms by the corresponding propositional variables.
- F is satisfiable if and only if $Prop(F)$ is satisfiable.

Example:

$$\begin{aligned} F &= \{p(f(a)) \vee \neg p(a), p(a) \vee \neg p(f(a))\} \\ Prop(F) &= \{x_{p(f(a))} \vee \neg x_{p(a)}, x_{p(a)} \vee \neg x_{p(f(a))}\} \end{aligned}$$

Corollary. A set of first-order universal formulas Φ is satisfiable if and only the set of propositional formulas $Prop(Gr(\Phi))$ is satisfiable.

We will not distinguish between ground atoms and their propositional encodings.

Reduction first-order to propositional

Ground formulas can be seen as propositional formulas as follows:

Consider a ground formula F.

- With each ground atom A in F associate a propositional variable x_A.
- Let $Prop(F)$ be a propositional formula obtained from F by replacing all atoms by the corresponding propositional variables.
- F is satisfiable if and only if $Prop(F)$ is satisfiable.

Example:

$$\begin{array}{rcl} F & = & \{p(f(a)) \lor \neg p(a), p(a) \lor \neg p(f(a))\} \\ Prop(F) & = & \{x_{p(f(a))} \lor \neg x_{p(a)}, x_{p(a)} \lor \neg x_{p(f(a))}\} \end{array}$$

Corollary. A set of first-order universal formulas Φ is satisfiable if and only the set of propositional formulas $Prop(Gr(\Phi))$ is satisfiable.

We will not distinguish between ground atoms and their propositional encodings.

Reduction first-order to propositional

Ground formulas can be seen as propositional formulas as follows:

Consider a ground formula F.

- With each ground atom A in F associate a propositional variable x_A.
- Let $Prop(F)$ be a propositional formula obtained from F by replacing all atoms by the corresponding propositional variables.
- F is satisfiable if and only if $Prop(F)$ is satisfiable.

Example:

$$\begin{array}{rcl} F & = & \{p(f(a)) \lor \neg p(a), p(a) \lor \neg p(f(a))\} \\ Prop(F) & = & \{x_{p(f(a))} \lor \neg x_{p(a)}, x_{p(a)} \lor \neg x_{p(f(a))}\} \end{array}$$

Corollary. A set of first-order universal formulas Φ is satisfiable if and only the set of propositional formulas $Prop(Gr(\Phi))$ is satisfiable.

We will not distinguish between ground atoms and their propositional encodings.

Reduction first-order to propositional

Ground formulas can be seen as propositional formulas as follows:

Consider a ground formula F.

- With each ground atom A in F associate a propositional variable x_A.
- Let $Prop(F)$ be a propositional formula obtained from F by replacing all atoms by the corresponding propositional variables.
- F is satisfiable if and only if $Prop(F)$ is satisfiable.

Example:

$$\begin{array}{rcl} F & = & \{p(f(a)) \vee \neg p(a), p(a) \vee \neg p(f(a))\} \\ Prop(F) & = & \{x_{p(f(a))} \vee \neg x_{p(a)}, x_{p(a)} \vee \neg x_{p(f(a))}\} \end{array}$$

Corollary. A set of first-order universal formulas Φ is satisfiable if and only the set of propositional formulas $Prop(Gr(\Phi))$ is satisfiable.

We will not distinguish between ground atoms and their propositional encodings.

Examples

Example: Consider a signature $\Sigma = (\{a/0, b/0\}, \{p/1, q/2\})$ and a set of clauses $S = \{\neg p(x) \vee q(x,a), \neg q(x,x) \vee p(x)\}$. Is S satisfiable?.

$$Gr(S) = \{ \quad \neg p(a) \vee q(a,a)$$
$$\neg p(b) \vee q(b,a)$$
$$\neg q(a,a) \vee p(a)$$
$$\neg q(b,b) \vee p(b) \quad \}$$

Apply any propositional method to check whether $Gr(S)$ is satisfiable.

Example: Consider a signature $\Sigma = (\{f/1, a/0\}, \{p/1\})$ and a set of clauses $S = \{p(x) \vee \neg p(f(x)), \neg p(x) \vee p(f(x))\}$. Is S satisfiable?.

The set of ground instances $Gr(S)$ is infinite:

$$p(a) \vee \neg p(f(a)), \quad p(f(a)) \vee \neg p(f(f(a))), \ldots, p(f^n(a)) \vee \neg p(f^{n+1}(a))), \ldots$$
$$\neg p(a) \vee p(f(a)), \quad \neg p(f(a)) \vee p(f(f(a))), \ldots, \neg p(f^n(a)) \vee p(f^{n+1}(a))), \ldots$$

Examples

Example: Consider a signature $\Sigma = (\{a/0, b/0\}, \{p/1, q/2\})$ and a set of clauses $S = \{\neg p(x) \vee q(x, a), \neg q(x, x) \vee p(x)\}$. Is S satisfiable?.

$$Gr(S) = \{ \quad \neg p(a) \vee q(a, a) \\ \neg p(b) \vee q(b, a) \\ \neg q(a, a) \vee p(a) \\ \neg q(b, b) \vee p(b) \quad \}$$

Apply any propositional method to check whether $Gr(S)$ is satisfiable.

Example: Consider a signature $\Sigma = (\{f/1, a/0\}, \{p/1\})$ and a set of clauses $S = \{p(x) \vee \neg p(f(x)), \neg p(x) \vee p(f(x))\}$. Is S satisfiable?.
The set of ground instances $Gr(S)$ is infinite:

$$p(a) \vee \neg p(f(a)), \quad p(f(a)) \vee \neg p(f(f(a))), \ldots, p(f^n(a)) \vee \neg p(f^{n+1}(a))), \ldots$$
$$\neg p(a) \vee p(f(a)), \quad \neg p(f(a)) \vee p(f(f(a))), \ldots, \neg p(f^n(a)) \vee p(f^{n+1}(a))), \ldots$$

Examples

Example: Consider a signature $\Sigma = (\{a/0, b/0\}, \{p/1, q/2\})$ and a set of clauses $S = \{\neg p(x) \vee q(x,a), \neg q(x,x) \vee p(x)\}$. Is S satisfiable?.

$$Gr(S) = \{ \quad \neg p(a) \vee q(a,a) \quad \neg p(b) \vee q(b,a) \quad \neg q(a,a) \vee p(a) \quad \neg q(b,b) \vee p(b) \quad \}$$

Apply any propositional method to check whether $Gr(S)$ is satisfiable.

Example: Consider a signature $\Sigma = (\{f/1, a/0\}, \{p/1\})$ and a set of clauses $S = \{p(x) \vee \neg p(f(x)), \neg p(x) \vee p(f(x))\}$. Is S satisfiable?.

The set of ground instances $Gr(S)$ is infinite:

$$p(a) \vee \neg p(f(a)), \quad p(f(a)) \vee \neg p(f(f(a))), \ldots, p(f^n(a)) \vee \neg p(f^{n+1}(a))), \ldots$$
$$\neg p(a) \vee p(f(a)), \quad \neg p(f(a)) \vee p(f(f(a))), \ldots, \neg p(f^n(a)) \vee p(f^{n+1}(a))), \ldots$$

Examples

Example: Consider a signature $\Sigma = (\{a/0, b/0\}, \{p/1, q/2\})$ and a set of clauses $S = \{\neg p(x) \vee q(x, a), \neg q(x, x) \vee p(x)\}$. Is S satisfiable?.

$$Gr(S) = \{ \quad \neg p(a) \vee q(a, a)$$
$$\neg p(b) \vee q(b, a)$$
$$\neg q(a, a) \vee p(a)$$
$$\neg q(b, b) \vee p(b) \quad \}$$

Apply any propositional method to check whether $Gr(S)$ is satisfiable.

Example: Consider a signature $\Sigma = (\{f/1, a/0\}, \{p/1\})$ and a set of clauses $S = \{p(x) \vee \neg p(f(x)), \neg p(x) \vee p(f(x))\}$. Is S satisfiable?.

The set of ground instances $Gr(S)$ is infinite:

$$p(a) \vee \neg p(f(a)), \quad p(f(a)) \vee \neg p(f(f(a))), \ldots, p(f^n(a)) \vee \neg p(f^{n+1}(a))), \ldots$$
$$\neg p(a) \vee p(f(a)), \quad \neg p(f(a)) \vee p(f(f(a))), \ldots, \neg p(f^n(a)) \vee p(f^{n+1}(a))), \ldots$$

Inference systems

Deduction, Inference Systems

An inference has the form:

$$\frac{F_1 \qquad \dots \qquad F_n}{G}$$

where $n \geq 0$, $F_1, \dots, F_n, G$ are formulas.

- $F_1 \dots F_n$ are called premises.
- G is called conclusion.

An inference rule R is a set of inferences.

An inference system, (or a calculus) $\mathbb{I}$ is a set of inference rules.

Deduction, Inference Systems

An inference has the form:

$$\frac{F_1 \qquad \ldots \qquad F_n}{G}$$

where $n \geq 0$, $F_1, \ldots, F_n, G$ are formulas.

- $F_1 \ldots F_n$ are called premises.
- G is called conclusion.

An inference rule R is a set of inferences.

An inference system, (or a calculus) $\mathbb{I}$ is a set of inference rules.

Derivation, proofs

- A derivation tree in $\mathbb{I}$ is a tree built from inferences.
- A proof of F (in $\mathbb{I}$) from $F_1, \ldots, F_n$ is a tree with leaves in $F_1, \ldots, F_n$ and the root F.
- A refutation proof is a proof of $\square$.
- F is derivable, (or provable) in $\mathbb{I}$ from a set of formulas S, denoted $S \vdash_{\mathbb{I}} F$, if there is a proof of F from formulas in S.

Soundness/Completeness

Soundness.

- An inference is sound if the conclusion of this inference logically follows from the premises ($\models$).
- An inference rule is sound if all its inferences are sound.
- An inference system is sound if all its inference rules are sound.

Lemma. If an inference system $\mathbb{I}$ is sound then for any set of formulas S:

$$S \vdash_{\mathbb{I}} \square \quad \text{implies} \quad S \models \bot$$

Completeness. An inference system $\mathbb{I}$ is refutationally complete if for any set of formulas S we have:

$$S \models \bot \quad \text{implies} \quad S \vdash_{\mathbb{I}} \square.$$

Soundness/Completeness

Soundness.

- An inference is sound if the conclusion of this inference logically follows from the premises ($\models$).
- An inference rule is sound if all its inferences are sound.
- An inference system is sound if all its inference rules are sound.

Lemma. If an inference system $\mathbb{I}$ is sound then for any set of formulas S:

$$S \vdash_{\mathbb{I}} \square \quad \text{implies} \quad S \models \bot$$

Completeness. An inference system $\mathbb{I}$ is refutationally complete if for any set of formulas S we have:

$$S \models \bot \quad \text{implies} \quad S \vdash_{\mathbb{I}} \square.$$

Soundness/Completeness

Soundness.

- An inference is sound if the conclusion of this inference logically follows from the premises ($\models$).
- An inference rule is sound if all its inferences are sound.
- An inference system is sound if all its inference rules are sound.

Lemma. If an inference system $\mathbb{I}$ is sound then for any set of formulas S:

$$S \vdash_{\mathbb{I}} \square \quad \text{implies} \quad S \models \bot$$

Completeness. An inference system $\mathbb{I}$ is refutationally complete if for any set of formulas S we have:

$$S \models \bot \quad \text{implies} \quad S \vdash_{\mathbb{I}} \square.$$

Proofs and reasoning methods

Formal Proofs:

- each step of a proof is easy to check
- proofs – certificates of correctness
- independent proof checking

Reasoning methods based on inference systems:

- efficient proof search
- restrictions on applicability of inference rules
- proof search strategies

Proofs and reasoning methods

Formal Proofs:

- each step of a proof is easy to check
- proofs – certificates of correctness
- independent proof checking

Reasoning methods based on inference systems:

- efficient proof search
- restrictions on applicability of inference rules
- proof search strategies

Propositional resolution

Propositional Resolution

Propositional Resolution inference system $\mathbb{BR}$, consists of the following inference rules:

- Binary resolution rule (BR):

$$\frac{C \vee p \qquad \neg p \vee D}{C \vee D} \; (BR)$$

- Binary positive factoring rule (BF):

$$\frac{C \vee p \vee p}{C \vee p} \; (BF)$$

where p is an atom.

Example

Given: $S = \{q \lor \neg p, p \lor q, \neg q\}$

A proof in resolution calculus:

$$\dfrac{\dfrac{\dfrac{q \lor \neg p \qquad p \lor q}{q \lor q}\text{(BR)}}{q}\text{(BF)} \qquad \neg q}{\square}\text{(BR)}$$

Another proof in resolution calculus:

$$\dfrac{\dfrac{\dfrac{q \lor \neg p \qquad \neg q}{\neg p}\text{(BR)} \qquad p \lor q}{q}\text{(BR)} \qquad \neg q}{\square}\text{(BR)}$$

Example

Given: $S = \{q \vee \neg p, p \vee q, \neg q\}$

A proof in resolution calculus:

$$\cfrac{\cfrac{\cfrac{q \vee \neg p \qquad p \vee q}{q \vee q}\ \text{(BR)}}{q}\ \text{(BF)} \qquad \neg q}{\square}\ \text{(BR)}$$

Another proof in resolution calculus:

$$\cfrac{\cfrac{\cfrac{q \vee \neg p \qquad \neg q}{\neg p}\ \text{(BR)} \qquad p \vee q}{q}\ \text{(BR)} \qquad \neg q}{\square}\ \text{(BR)}$$

Example

Given: $S = \{q \vee \neg p, p \vee q, \neg q\}$

A proof in resolution calculus:

$$\dfrac{\dfrac{\dfrac{q \vee \neg p \qquad p \vee q}{q \vee q}\,(\text{BR})}{q}\,(\text{BF}) \qquad \neg q}{\square}\,(\text{BR})$$

Another proof in resolution calculus:

$$\dfrac{\dfrac{\dfrac{q \vee \neg p \qquad \neg q}{\neg p}\,(\text{BR}) \qquad p \vee q}{q}\,(\text{BR}) \qquad \neg q}{\square}\,(\text{BR})$$

Example

Given: $S = \{q \lor \neg p, p \lor q, \neg q\}$

A proof in resolution calculus:

$$\dfrac{\dfrac{\dfrac{q \lor \neg p \qquad p \lor q}{q \lor q}\ (BR)}{q}\ (BF) \qquad \neg q}{\Box}\ (BR)$$

Another proof in resolution calculus:

$$\dfrac{\dfrac{\dfrac{q \lor \neg p \qquad \neg q}{\neg p}\ (BR) \qquad p \lor q}{q}\ (BR) \qquad \neg q}{\Box}\ (BR)$$

Example

Given: $S = \{q \lor \neg p, p \lor q, \neg q\}$

A proof in resolution calculus:

$$\dfrac{\dfrac{\dfrac{q \lor \neg p \qquad p \lor q}{q \lor q}\ \text{(BR)}}{q}\ \text{(BF)} \qquad \neg q}{\square}\ \text{(BR)}$$

Another proof in resolution calculus:

$$\dfrac{\dfrac{\dfrac{q \lor \neg p \qquad \neg q}{\neg p}\ \text{(BR)} \qquad p \lor q}{q}\ \text{(BR)} \qquad \neg q}{\square}\ \text{(BR)}$$

Example

Given: $S = \{q \vee \neg p, p \vee q, \neg q\}$

A proof in resolution calculus:

$$\dfrac{\dfrac{\dfrac{q \vee \neg p \qquad p \vee q}{q \vee q}\,(BR)}{q}\,(BF) \qquad \neg q}{\square}\,(BR)$$

Another proof in resolution calculus:

$$\dfrac{\dfrac{\dfrac{q \vee \neg p \qquad \neg q}{\neg p}\,(BR) \qquad p \vee q}{q}\,(BR) \qquad \neg q}{\square}\,(BR)$$

Example

Given: $S = \{q \vee \neg p, p \vee q, \neg q\}$

A proof in resolution calculus:

$$\dfrac{\dfrac{\dfrac{q \vee \neg p \qquad p \vee q}{q \vee q}\ \text{(BR)}}{q}\ \text{(BF)} \qquad \neg q}{\square}\ \text{(BR)}$$

Another proof in resolution calculus:

$$\dfrac{\dfrac{\dfrac{q \vee \neg p \qquad \neg q}{\neg p}\ \text{(BR)} \qquad p \vee q}{q}\ \text{(BR)} \qquad \neg q}{\square}\ \text{(BR)}$$

Example

Given: $S = \{q \lor \neg p, p \lor q, \neg q\}$

A proof in resolution calculus:

$$\frac{\dfrac{\dfrac{q \lor \neg p \qquad p \lor q}{q \lor q}\;\text{(BR)}}{q}\;\text{(BF)} \qquad \neg q}{\Box}\;\text{(BR)}$$

Another proof in resolution calculus:

$$\frac{\dfrac{\dfrac{q \lor \neg p \qquad \neg q}{\neg p}\;\text{(BR)} \qquad p \lor q}{q}\;\text{(BR)} \qquad \neg q}{\Box}\;\text{(BR)}$$

Linear Proofs

Tree Proof:

$$\dfrac{\dfrac{\dfrac{q \lor \neg p \qquad p \lor q}{q \lor q}\ \text{(BR)}}{q}\ \text{(BF)} \qquad \neg q}{\square}\ \text{(BR)}$$

Linear Proof:

1.	$q \lor \neg p$	input
2.	$p \lor q$	input
3.	$\neg q$	input
4.	$q \lor q$	BR (1,2)
5.	q	BF (4)
6.	$\square$	BR (3,5)

Soundness of resolution

Theorem. [Soundness] The resolution inference system $\mathbb{BR}$ is sound.

Proof. Conclusions of BR and BF are logically implied by the premises.

- $\{C \vee p, \neg p \vee D\} \models C \vee D$
- $\{C \vee L \vee L\} \models C \vee L$

Theorem. [Completeness] The resolution inference system $\mathbb{BR}$ is refutationally complete.

We need to show that for any set of clauses S:

$$S \models \square \quad \text{implies} \quad S \vdash_{\mathbb{BR}} \square.$$

or equivalently:

$$S \nvdash_{\mathbb{BR}} \square \quad \text{implies} \quad S \text{ is satisfiable}$$

Completeness of resolution is one of the key results in automated reasoning. We will present the proof after some preparations.

Soundness of resolution

Theorem. [Soundness] The resolution inference system $\mathbb{BR}$ is sound.

Proof. Conclusions of BR and BF are logically implied by the premises.

- $\{C \lor p, \neg p \lor D\} \models C \lor D$
- $\{C \lor L \lor L\} \models C \lor L$

Theorem. [Completeness] The resolution inference system $\mathbb{BR}$ is refutationally complete.

We need to show that for any set of clauses S:

$$S \models \Box \quad \text{implies} \quad S \vdash_{\mathbb{BR}} \Box.$$

or equivalently:

$$S \not\vdash_{\mathbb{BR}} \Box \quad \text{implies} \quad S \text{ is satisfiable}$$

Completeness of resolution is one of the key results in automated reasoning. We will present the proof after some preparations.

Search for unsatisfiability

Basic Idea. A Saturation Process:

Given set of clauses S we exhaustively apply all inference rules adding the conclusions to this set until the contradiction ($\Box$) is derived.

$$S_0 \Rightarrow S_1 \Rightarrow \dots S_n \Rightarrow \dots$$

More formally: define one-step resolution expansion

$$Res(S) = \{C \mid C \text{ is a conclusion of } \mathbb{BR} \text{ applied to clauses in } S\}$$

Define

$$S_0 = S, S_1 = Res(S_0), \dots, S_n = Res(S_{n-1}), \dots$$

is called the basic saturation process.

The limit of the basic saturation process is $Res^*(S) = \bigcup_{0 \leq i < \omega} S_i$

Lemma. A clause C is derivable from S using $\mathbb{BR}$ if and only if $C \in Res^*(S)$.

Search for unsatisfiability

Basic Idea. A Saturation Process:

Given set of clauses S we exhaustively apply all inference rules adding the conclusions to this set until the contradiction ($\square$) is derived.

$$S_0 \Rightarrow S_1 \Rightarrow \ldots S_n \Rightarrow \ldots$$

More formally: define one-step resolution expansion

$$Res(S) = \{C \mid C \text{ is a conclusion of } \mathbb{BR} \text{ applied to clauses in } S\}$$

Define

$$S_0 = S, S_1 = Res(S_0), \ldots, S_n = Res(S_{n-1}), \ldots$$

is called the basic saturation process.

The limit of the basic saturation process is $Res^*(S) = \bigcup_{0 \leq i < \omega} S_i$

Lemma. A clause C is derivable from S using $\mathbb{BR}$ if and only if $C \in Res^*(S)$.

Search for unsatisfiability

Basic Idea. A Saturation Process:

Given set of clauses S we exhaustively apply all inference rules adding the conclusions to this set until the contradiction ($\square$) is derived.

$$S_0 \Rightarrow S_1 \Rightarrow \ldots S_n \Rightarrow \ldots$$

More formally: define one-step resolution expansion

$$Res(S) = \{C \mid C \text{ is a conclusion of } \mathbb{BR} \text{ applied to clauses in } S\}$$

Define

$$S_0 = S, S_1 = Res(S_0), \ldots, S_n = Res(S_{n-1}), \ldots$$

is called the basic saturation process.

The limit of the basic saturation process is $Res^*(S) = \bigcup_{0 \leq i < \omega} S_i$

Lemma. A clause C is derivable from S using $\mathbb{BR}$ if and only if $C \in Res^*(S)$.

Search for unsatisfiability

Basic Idea. A Saturation Process:

Given set of clauses S we exhaustively apply all inference rules adding the conclusions to this set until the contradiction ($\square$) is derived.

$$S_0 \Rightarrow S_1 \Rightarrow \dots S_n \Rightarrow \dots$$

More formally: define one-step resolution expansion

$$Res(S) = \{C \mid C \text{ is a conclusion of } \mathbb{BR} \text{ applied to clauses in } S\}$$

Define

$$S_0 = S, S_1 = Res(S_0), \dots, S_n = Res(S_{n-1}), \dots$$

is called the basic saturation process.

The limit of the basic saturation process is $Res^*(S) = \bigcup_{0 \leq i < \omega} S_i$

Lemma. A clause C is derivable from S using $\mathbb{BR}$ if and only if $C \in Res^*(S)$.

Saturated sets and completeness

A set of clauses S is saturated if $Res(S) \subseteq S$.

Note: The limit of any basic saturation process is a saturated set.

Completeness of the resolution calculus $\mathbb{BR}$ can be reformulated as follows. For any saturated set of clauses S:

$$\square \notin S \text{ implies } S \text{ is satisfiable}$$

Saturated sets and completeness

A set of clauses S is saturated if $Res(S) \subseteq S$.

Note: The limit of any basic saturation process is a saturated set.

Completeness of the resolution calculus $\mathbb{BR}$ can be reformulated as follows. For any saturated set of clauses S:

$$\square \notin S \text{ implies } S \text{ is satisfiable}$$

Completeness of resolution

Main idea

Consider a saturated set of clauses S such that $\square \notin S$.

How we can show that S is satisfiable?

Model construction:

1. Build a "candidate" Herbrand model I with the goal to satisfy clauses in S. The model is built inductively based on a well-founded order $\succ$ on clauses.
2. show that if S is saturated then I is indeed a model of S.

Clause representation: multi-sets of literals.

Next: multi-sets, well-founded orders on atoms, literals and clauses.

Main idea

Consider a saturated set of clauses S such that $\square \notin S$.

How we can show that S is satisfiable?

Model construction:

1. Build a "candidate" Herbrand model I with the goal to satisfy clauses in S. The model is built inductively based on a well-founded order $\succ$ on clauses.
2. show that if S is saturated then I is indeed a model of S.

Clause representation: multi-sets of literals.

Next: multi-sets, well-founded orders on atoms, literals and clauses.

Main idea

Consider a saturated set of clauses S such that $\square \notin S$.

How we can show that S is satisfiable?

Model construction:

1. Build a "candidate" Herbrand model I with the goal to satisfy clauses in S. The model is built inductively based on a well-founded order $\succ$ on clauses.
2. show that if S is saturated then I is indeed a model of S.

Clause representation: multi-sets of literals.

Next: multi-sets, well-founded orders on atoms, literals and clauses.

Main idea

Consider a saturated set of clauses S such that $\square \notin S$.

How we can show that S is satisfiable?

Model construction:

1. Build a "candidate" Herbrand model I with the goal to satisfy clauses in S. The model is built inductively based on a well-founded order $\succ$ on clauses.
2. show that if S is saturated then I is indeed a model of S.

Clause representation: multi-sets of literals.

Next: multi-sets, well-founded orders on atoms, literals and clauses.

Main idea

Consider a saturated set of clauses S such that $\square \notin S$.

How we can show that S is satisfiable?

Model construction:

1. Build a "candidate" Herbrand model I with the goal to satisfy clauses in S. The model is built inductively based on a well-founded order $\succ$ on clauses.
2. show that if S is saturated then I is indeed a model of S.

Clause representation: multi-sets of literals.

Next: multi-sets, well-founded orders on atoms, literals and clauses.

Main idea

Consider a saturated set of clauses S such that $\square \notin S$.

How we can show that S is satisfiable?

Model construction:

1. Build a "candidate" Herbrand model I with the goal to satisfy clauses in S. The model is built inductively based on a well-founded order $\succ$ on clauses.
2. show that if S is saturated then I is indeed a model of S.

Clause representation: multi-sets of literals.

Next: multi-sets, well-founded orders on atoms, literals and clauses.

Multi-Sets

Clauses will be represented as multi-sets of literals.

- Multi-sets are "sets which allow repetition".
  Example: $\{a, a, b\}, \quad \{a, b, a\}, \quad \{a, b\}$
- Formally, let X be a set.
  A multi-set S over X is a mapping $S : X \to \mathbb{N}$.
- Intuitively, $S(x)$ specifies the number of occurrences of the element x (of the base set X) within S.
- Example: $S = \{a, a, a, b, b\}$ is a multi-set over $\{a, b, c\}$,
  where $S(a) = 3$, $S(b) = 2$, $S(c) = 0$.
- We say that x is an element of S, if $S(x) > 0$.

Multi-Sets (cont'd)

- We use set notation ($\in$, $\subset$, $\subseteq$, $\cup$, $\cap$, etc.) with analogous meaning also for multi-sets, e.g.,

$$
\begin{aligned}
(S_1 \cup S_2)(x) &= S_1(x) + S_2(x) \\
(S_1 \cap S_2)(x) &= \min\{S_1(x), S_2(x)\} \\
(S_1 \setminus S_2)(x) &= S_1(x) \dot{-} S_2(x)
\end{aligned}
$$

- A multi-set S over X is called finite, if

$$|\{x \in X \mid S(x) > 0\}| < \infty.$$

- From now on we consider finite multi-sets only.

Multi-Set Orderings $\succ_{\mathrm{mul}}$

Definition

Let $(X, \succ)$ be a (strict) ordering. The multi-set extension $\succ_{\mathrm{mul}}$ of $\succ$ to (finite) multi-sets over X is defined by

$$S_1 \succ_{\mathrm{mul}} S_2 \iff S_1 \neq S_2 \text{ and } \forall x \in S_2 \backslash S_1.\ \exists y \in S_1 \backslash S_2.\ y \succ x$$

1. Remove common occurrences of elements from S_1 and S_2. Assume this gives $S_1' \neq S_2'$.
2. Then check that for every element x in S_2' there is an element $y \in S_1'$ that is greater than x. Then $S_1 \succ_{\mathrm{mul}} S_2$.

Example $\{5, 5, 4, 3, 2\} \succ_{\mathrm{mul}} \{5, 4, 4, 3, 3, 2\}$

Multi-Set Orderings $\succ_{\text{mul}}$

Definition

Let $(X, \succ)$ be a (strict) ordering. The multi-set extension $\succ_{\text{mul}}$ of $\succ$ to (finite) multi-sets over X is defined by

$$S_1 \succ_{\text{mul}} S_2 \iff S_1 \neq S_2 \text{ and } \forall x \in S_2 \backslash S_1.\ \exists y \in S_1 \backslash S_2.\ y \succ x$$

1. Remove common occurrences of elements from S_1 and S_2. Assume this gives $S_1' \neq S_2'$.
2. Then check that for every element x in S_2' there is an element $y \in S_1'$ that is greater than x. Then $S_1 \succ_{\text{mul}} S_2$.

Example $\{5, 5, 4, 3, 2\} \succ_{\text{mul}} \{5, 4, 4, 3, 3, 2\}$

Multi-Set Orderings $\succ_{\mathrm{mul}}$

Definition

Let $(X, \succ)$ be a (strict) ordering. The multi-set extension $\succ_{\mathrm{mul}}$ of $\succ$ to (finite) multi-sets over X is defined by

$$S_1 \succ_{\mathrm{mul}} S_2 \iff S_1 \neq S_2 \text{ and } \forall x \in S_2 \setminus S_1.\ \exists y \in S_1 \setminus S_2.\ y \succ x$$

1. Remove common occurrences of elements from S_1 and S_2. Assume this gives $S_1' \neq S_2'$.
2. Then check that for every element x in S_2' there is an element $y \in S_1'$ that is greater than x. Then $S_1 \succ_{\mathrm{mul}} S_2$.

Example $\{5, 5, 4, 3, 2\} \succ_{\mathrm{mul}} \{5, 4, 4, 3, 3, 2\}$

Properties of Multi-Set Orderings

An ordering over X is well-founded if if there is no infinite decreasing chain $x_0 \succ x_1 \succ x_2 \succ \ldots$ of elements $x_i \in X$.

Lemma

$(X, \succ)$ is well-founded iff every non-empty subset Y of X has a minimal element.

Theorem

Let $\succ$ be an ordering. Then

1. *$\succ_{\mathrm{mul}}$ is an ordering.*
2. *if $\succ$ well-founded then $\succ_{\mathrm{mul}}$ well-founded.*
3. *if $\succ$ total then $\succ_{\mathrm{mul}}$ total*

Q: How many multi-sets less than $\{3\}$?

Properties of Multi-Set Orderings

An ordering over X is well-founded if if there is no infinite decreasing chain $x_0 \succ x_1 \succ x_2 \succ \ldots$ of elements $x_i \in X$.

Lemma

$(X, \succ)$ is well-founded iff every non-empty subset Y of X has a minimal element.

Theorem

Let $\succ$ be an ordering. Then

1. *$\succ_{\mathrm{mul}}$ is an ordering.*
2. *if $\succ$ well-founded then $\succ_{\mathrm{mul}}$ well-founded.*
3. *if $\succ$ total then $\succ_{\mathrm{mul}}$ total*

Q: How many multi-sets less than $\{3\}$?

Properties of Multi-Set Orderings

An ordering over X is well-founded if if there is no infinite decreasing chain $x_0 \succ x_1 \succ x_2 \succ \dots$ of elements $x_i \in X$.

Lemma

$(X, \succ)$ is well-founded iff every non-empty subset Y of X has a minimal element.

Theorem

Let $\succ$ be an ordering. Then

1. $\succ_{\mathrm{mul}}$ *is an ordering.*
2. *if $\succ$ well-founded then $\succ_{\mathrm{mul}}$ well-founded.*
3. *if $\succ$ total then $\succ_{\mathrm{mul}}$ total*

Q: How many multi-sets less than $\{3\}$?

Properties of Multi-Set Orderings

An ordering over X is well-founded if if there is no infinite decreasing chain $x_0 \succ x_1 \succ x_2 \succ \ldots$ of elements $x_i \in X$.

Lemma

$(X, \succ)$ is well-founded iff every non-empty subset Y of X has a minimal element.

Theorem

Let $\succ$ be an ordering. Then

1. *$\succ_{\mathrm{mul}}$ is an ordering.*
2. *if $\succ$ well-founded then $\succ_{\mathrm{mul}}$ well-founded.*
3. *if $\succ$ total then $\succ_{\mathrm{mul}}$ total*

Q: How many multi-sets less than $\{3\}$?

Order on atoms, literals and clauses

Consider a set of ground atoms $\mathcal{P}$.

Let $\succ$ be any well-founded, total order on $\mathcal{P}$.

- Extend $\succ$ to a total well-founded order on literals as follows:
  - if $A \succ B$ then $(\neg)A \succ (\neg)B$, and
  - $\neg A \succ A$.
- Extend $\succ$ to a total well-founded order on ground clauses as follows:
  $L_1 \vee \ldots \vee L_n \succ M_1 \vee \ldots \vee M_k$ iff
  $\{L_1, \ldots, L_n\} \succ_{\text{mul}} \{M_1, \ldots, M_k\}$.

Clauses are considered as multi-sets of literals.

We will ambiguously use $\succ$ for $\succ_{\text{mul}}$.

Q: What is the smallest clause ?

Q: Consider $A_1 \prec A_2 \prec \ldots A_n \prec \ldots$

How many clauses are less than $A_2 \vee A_1$?

Order on atoms, literals and clauses

Consider a set of ground atoms $\mathcal{P}$.

Let $\succ$ be any well-founded, total order on $\mathcal{P}$.

- Extend $\succ$ to a total well-founded order on literals as follows:
  - if $A \succ B$ then $(\neg)A \succ (\neg)B$, and
  - $\neg A \succ A$.
- Extend $\succ$ to a total well-founded order on ground clauses as follows:
  $L_1 \vee \ldots \vee L_n \succ M_1 \vee \ldots \vee M_k$ iff
  $\{L_1, \ldots, L_n\} \succ_{\text{mul}} \{M_1, \ldots, M_k\}$.

Clauses are considered as multi-sets of literals.

We will ambiguously use $\succ$ for $\succ_{\text{mul}}$.

Q: What is the smallest clause ?

Q: Consider $A_1 \prec A_2 \prec \ldots A_n \prec \ldots$

How many clauses are less than $A_2 \vee A_1$?

Order on atoms, literals and clauses

Consider a set of ground atoms $\mathcal{P}$.

Let $\succ$ be any well-founded, total order on $\mathcal{P}$.

- Extend $\succ$ to a total well-founded order on literals as follows:
  - if $A \succ B$ then $(\neg)A \succ (\neg)B$, and
  - $\neg A \succ A$.
- Extend $\succ$ to a total well-founded order on ground clauses as follows:
  $L_1 \vee \ldots \vee L_n \succ M_1 \vee \ldots \vee M_k$ iff
  $\{L_1, \ldots, L_n\} \succ_{\mathrm{mul}} \{M_1, \ldots, M_k\}$.

Clauses are considered as multi-sets of literals.

We will ambiguously use $\succ$ for $\succ_{\mathrm{mul}}$.

Q: What is the smallest clause ?

Q: Consider $A_1 \prec A_2 \prec \ldots A_n \prec \ldots$

How many clauses are less than $A_2 \vee A_1$?

Order on atoms, literals and clauses

Consider a set of ground atoms $\mathcal{P}$.

Let $\succ$ be any well-founded, total order on $\mathcal{P}$.

- Extend $\succ$ to a total well-founded order on literals as follows:
  - if $A \succ B$ then $(\neg)A \succ (\neg)B$, and
  - $\neg A \succ A$.
- Extend $\succ$ to a total well-founded order on ground clauses as follows:
  $L_1 \vee \ldots \vee L_n \succ M_1 \vee \ldots \vee M_k$ iff
  $\{L_1, \ldots, L_n\} \succ_{\mathrm{mul}} \{M_1, \ldots, M_k\}$.

Clauses are considered as multi-sets of literals.

We will ambiguously use $\succ$ for $\succ_{\mathrm{mul}}$.

Q: What is the smallest clause ?

Q: Consider $A_1 \prec A_2 \prec \ldots A_n \prec \ldots$

How many clauses are less than $A_2 \vee A_1$?

Order on atoms, literals and clauses

Consider a set of ground atoms $\mathcal{P}$.

Let $\succ$ be any well-founded, total order on $\mathcal{P}$.

- Extend $\succ$ to a total well-founded order on literals as follows:
  - if $A \succ B$ then $(\neg)A \succ (\neg)B$, and
  - $\neg A \succ A$.
- Extend $\succ$ to a total well-founded order on ground clauses as follows:
  $L_1 \vee \ldots \vee L_n \succ M_1 \vee \ldots \vee M_k$ iff
  $\{L_1, \ldots, L_n\} \succ_{\mathrm{mul}} \{M_1, \ldots, M_k\}$.

Clauses are considered as multi-sets of literals.

We will ambiguously use $\succ$ for $\succ_{\mathrm{mul}}$.

Q: What is the smallest clause ?

Q: Consider $A_1 \prec A_2 \prec \ldots A_n \prec \ldots$

How many clauses are less than $A_2 \vee A_1$?

The model construction *[Bachmair, Ganzinger]*

Consider S is a set of clauses.

Construct a Herbrand interpretation I_S aiming at satisfying clauses in S.

- consider clauses in the order $\succ$ from small to large
- satisfy the next clause $A \vee C$ by adding A to I_S
  provided certain conditions are met.

The model construction *[Bachmair, Ganzinger]*

More formally: Goal construct I_S such that $I_S \models S$ if S is saturated.

Consider a clause $C \in S$ that we would like to satisfy.

By induction assume that for all smaller clauses $D \prec C$ we constructed:

- $\epsilon_D = \begin{cases} \{A\}, \text{ such that } A \in D, \text{ or} \\ \emptyset \end{cases}$

Define: interpretation up-to C as $I_C = \bigcup_{D \prec C} \epsilon_D$.

Define: satisfying atom ϵ_C for C as

- $\epsilon_C = \{A\}$ (in this case C is called productive) if
  - C is false in I_C: $I_C \not\models C$, and
  - $C = A \vee C'$ and A is maximal: $\{A\} \succ C'$.
- $\epsilon_C = \emptyset$ otherwise.

Define: interpretation at C to be $I^C = I_C \cup \epsilon_C$.

Candidate model: $I_S = \bigcup_{C \in S} I^C = \bigcup_{C \in S} \epsilon_C$.

The model construction *[Bachmair, Ganzinger]*

More formally: Goal construct I_S such that $I_S \models S$ if S is saturated.

Consider a clause $C \in S$ that we would like to satisfy.

By induction assume that for all smaller clauses $D \prec C$ we constructed:

- $\epsilon_D = \begin{cases} \{A\}, \text{ such that } A \in D, \text{ or} \\ \emptyset \end{cases}$

Define: interpretation up-to C as $I_C = \bigcup_{D \prec C} \epsilon_D$.

Define: satisfying atom ϵ_C for C as

- $\epsilon_C = \{A\}$ (in this case C is called productive) if
  - C is false in I_C: $I_C \not\models C$, and
  - $C = A \vee C'$ and A is maximal: $\{A\} \succ C'$.
- $\epsilon_C = \emptyset$ otherwise.

Define: interpretation at C to be $I^C = I_C \cup \epsilon_C$.

Candidate model: $I_S = \bigcup_{C \in S} I^C = \bigcup_{C \in S} \epsilon_C$.

The model construction *[Bachmair, Ganzinger]*

More formally: Goal construct I_S such that $I_S \models S$ if S is saturated.

Consider a clause $C \in S$ that we would like to satisfy.

By induction assume that for all smaller clauses $D \prec C$ we constructed:

- $\epsilon_D = \begin{cases} \{A\}, \text{ such that } A \in D, \text{ or} \\ \emptyset \end{cases}$

Define: interpretation up-to C as $I_C = \bigcup_{D \prec C} \epsilon_D$.

Define: satisfying atom ϵ_C for C as

- $\epsilon_C = \{A\}$ (in this case C is called productive) if
  - C is false in I_C: $I_C \not\models C$, and
  - $C = A \vee C'$ and A is maximal: $\{A\} \succ C'$.
- $\epsilon_C = \emptyset$ otherwise.

Define: interpretation at C to be $I^C = I_C \cup \epsilon_C$.

Candidate model: $I_S = \bigcup_{C \in S} I^C = \bigcup_{C \in S} \epsilon_C$.

The model construction [Bachmair, Ganzinger]

More formally: Goal construct I_S such that $I_S \models S$ if S is saturated.

Consider a clause $C \in S$ that we would like to satisfy.

By induction assume that for all smaller clauses $D \prec C$ we constructed:

- $\epsilon_D = \begin{cases} \{A\}, \text{ such that } A \in D, \text{ or} \\ \emptyset \end{cases}$

Define: interpretation up-to C as $I_C = \bigcup_{D \prec C} \epsilon_D$.

Define: satisfying atom ϵ_C for C as

- $\epsilon_C = \{A\}$ (in this case C is called productive) if
  - C is false in I_C: $I_C \not\models C$, and
  - $C = A \vee C'$ and A is maximal: $\{A\} \succ C'$.
- $\epsilon_C = \emptyset$ otherwise.

Define: interpretation at C to be $I^C = I_C \cup \epsilon_C$.

Candidate model: $I_S = \bigcup_{C \in S} I^C = \bigcup_{C \in S} \epsilon_C$.

The model construction *[Bachmair, Ganzinger]*

More formally: Goal construct I_S such that $I_S \models S$ if S is saturated.

Consider a clause $C \in S$ that we would like to satisfy.

By induction assume that for all smaller clauses $D \prec C$ we constructed:

- $\epsilon_D = \begin{cases} \{A\}, \text{ such that } A \in D, \text{ or} \\ \emptyset \end{cases}$

Define: interpretation up-to C as $I_C = \bigcup_{D \prec C} \epsilon_D$.

Define: satisfying atom ϵ_C for C as

- $\epsilon_C = \{A\}$ (in this case C is called productive) if
  - C is false in I_C: $I_C \not\models C$, and
  - $C = A \vee C'$ and A is maximal: $\{A\} \succ C'$.
- $\epsilon_C = \emptyset$ otherwise.

Define: interpretation at C to be $I^C = I_C \cup \epsilon_C$.

Candidate model: $I_S = \bigcup_{C \in S} I^C = \bigcup_{C \in S} \epsilon_C$.

The model construction *[Bachmair, Ganzinger]*

More formally: Goal construct I_S such that $I_S \models S$ if S is saturated.

Consider a clause $C \in S$ that we would like to satisfy.

By induction assume that for all smaller clauses $D \prec C$ we constructed:

- $\epsilon_D = \begin{cases} \{A\}, \text{ such that } A \in D, \text{ or} \\ \emptyset \end{cases}$

Define: interpretation up-to C as $I_C = \bigcup_{D \prec C} \epsilon_D$.

Define: satisfying atom ϵ_C for C as

- $\epsilon_C = \{A\}$ (in this case C is called productive) if
  - C is false in I_C: $I_C \not\models C$, and
  - $C = A \vee C'$ and A is maximal: $\{A\} \succ C'$.
- $\epsilon_C = \emptyset$ otherwise.

Define: interpretation at C to be $I^C = I_C \cup \epsilon_C$.

Candidate model: $I_S = \bigcup_{C \in S} I^C = \bigcup_{C \in S} \epsilon_C$.

The model construction [Bachmair, Ganzinger]

More formally: Goal construct I_S such that $I_S \models S$ if S is saturated.

Consider a clause $C \in S$ that we would like to satisfy.

By induction assume that for all smaller clauses $D \prec C$ we constructed:

- $\epsilon_D = \begin{cases} \{A\}, \text{ such that } A \in D, \text{ or} \\ \emptyset \end{cases}$

Define: interpretation up-to C as $I_C = \bigcup_{D \prec C} \epsilon_D$.

Define: satisfying atom ϵ_C for C as

- $\epsilon_C = \{A\}$ (in this case C is called productive) if
  - C is false in I_C: $I_C \not\models C$, and
  - $C = A \vee C'$ and A is maximal: $\{A\} \succ C'$.
- $\epsilon_C = \emptyset$ otherwise.

Define: interpretation at C to be $I^C = I_C \cup \epsilon_C$.

Candidate model: $I_S = \bigcup_{C \in S} I^C = \bigcup_{C \in S} \epsilon_C$.

Counter-example reduction [Bachmair, Ganzinger]

Lemma Model construction is monotone:

If $I^C \models C$ then for all $D \succeq C$: $I^D \models C$ and $I_S \models C$.

If $I^C \not\models C$ then for all $D \succeq C$: $I^D \not\models C$ and $I_S \not\models C$.

Theorem. If S is saturated and $\square \notin S$ then $I_S \models S$.

Poof. (Main ideas) Assume S is saturated and $I_S \not\models S$.

- The smallest counter-example: there is the smallest clause $C \in S$ such $I_S \not\models C$. (Because $\succ$ is well-founded).
- Inference by $\mathbb{BR}$ is applicable to C in S with the conclusion G s.t.
  - $G \prec C$, and
  - $I_S \not\models G$, and
  - $G \in S$
- G is a smaller counter-example! Contradiction with minimality of C.

Key property: resolution reduces counter-examples

Counter-example reduction [Bachmair, Ganzinger]

Lemma Model construction is monotone:

If $I^C \models C$ then for all $D \succeq C$: $I^D \models C$ and $I_S \models C$.

If $I^C \not\models C$ then for all $D \succeq C$: $I^D \not\models C$ and $I_S \not\models C$.

Theorem. If S is saturated and $\square \notin S$ then $I_S \models S$.

Poof. (Main ideas) Assume S is saturated and $I_S \not\models S$.

- The smallest counter-example: there is the smallest clause $C \in S$ such $I_S \not\models C$. (Because $\succ$ is well-founded).
- Inference by $\mathbb{BR}$ is applicable to C in S with the conclusion G s.t.
  - $G \prec C$, and
  - $I_S \not\models G$, and
  - $G \in S$
- G is a smaller counter-example! Contradiction with minimality of C.

Key property: resolution reduces counter-examples

Counter-example reduction *[Bachmair, Ganzinger]*

Lemma Model construction is monotone:

If $I^C \models C$ then for all $D \succeq C$: $I^D \models C$ and $I_S \models C$.

If $I^C \not\models C$ then for all $D \succeq C$: $I^D \not\models C$ and $I_S \not\models C$.

Theorem. If S is saturated and $\square \notin S$ then $I_S \models S$.

Poof. (Main ideas) Assume S is saturated and $I_S \not\models S$.

- The smallest counter-example: there is the smallest clause $C \in S$ such $I_S \not\models C$. (Because $\succ$ is well-founded).
- Inference by $\mathbb{BR}$ is applicable to C in S with the conclusion G s.t.
  - $G \prec C$, and
  - $I_S \not\models G$, and
  - $G \in S$
- G is a smaller counter-example! Contradiction with minimality of C.

Key property: resolution reduces counter-examples

Counter-example reduction [Bachmair, Ganzinger]

Lemma Model construction is monotone:

If $I^C \models C$ then for all $D \succeq C$: $I^D \models C$ and $I_S \models C$.

If $I^C \not\models C$ then for all $D \succeq C$: $I^D \not\models C$ and $I_S \not\models C$.

Theorem. If S is saturated and $\Box \notin S$ then $I_S \models S$.

Poof. (Main ideas) Assume S is saturated and $I_S \not\models S$.

- The smallest counter-example: there is the smallest clause $C \in S$ such $I_S \not\models C$. (Because $\succ$ is well-founded).
- Inference by $\mathbb{BR}$ is applicable to C in S with the conclusion G s.t.
  - $G \prec C$, and
  - $I_S \not\models G$, and
  - $G \in S$
- G is a smaller counter-example! Contradiction with minimality of C.

Key property: resolution reduces counter-examples

Counter-example reduction *[Bachmair, Ganzinger]*

Lemma Model construction is monotone:

If $I^C \models C$ then for all $D \succeq C$: $I^D \models C$ and $I_S \models C$.

If $I^C \not\models C$ then for all $D \succeq C$: $I^D \not\models C$ and $I_S \not\models C$.

Theorem. If S is saturated and $\square \notin S$ then $I_S \models S$.

Poof. (Main ideas) Assume S is saturated and $I_S \not\models S$.

- The smallest counter-example: there is the smallest clause $C \in S$ such $I_S \not\models C$. (Because $\succ$ is well-founded).
- Inference by $\mathbb{BR}$ is applicable to C in S with the conclusion G s.t.
  - $G \prec C$, and
  - $I_S \not\models G$, and
  - $G \in S$
- G is a smaller counter-example! Contradiction with minimality of C.

Key property: resolution reduces counter-examples

Literal selection functions

Unrestricted resolution is a very prolific inference system.

Use selection function to restrict applicability of rules to selected literals.

Selection function: selects a subset of literals in a clause $sel(C) \subseteq C$.

Informally: only selected literals are eligible for inferences.

A selection function *sel* is admissible if

- $sel(C) = \emptyset$ only when C is the empty clause.
- if $sel(C)$ consists of only positive literals then $sel(C)$ also contains all maximal literals in C.

We will underline selected literals: $\underline{\neg A} \vee B \vee C$

Literal selection functions

Unrestricted resolution is a very prolific inference system.

Use selection function to restrict applicability of rules to selected literals.

Selection function: selects a subset of literals in a clause $sel(C) \subseteq C$.

Informally: only selected literals are eligible for inferences.

A selection function *sel* is admissible if

- $sel(C) = \emptyset$ only when C is the empty clause.
- if $sel(C)$ consists of only positive literals then $sel(C)$ also contains all maximal literals in C.

We will underline selected literals: $\underline{\neg A} \vee B \vee C$

Literal selection functions

Unrestricted resolution is a very prolific inference system.

Use selection function to restrict applicability of rules to selected literals.

Selection function: selects a subset of literals in a clause $sel(C) \subseteq C$.

Informally: only selected literals are eligible for inferences.

A selection function *sel* is admissible if

- $sel(C) = \emptyset$ only when C is the empty clause.
- if $sel(C)$ consists of only positive literals then $sel(C)$ also contains all maximal literals in C.

We will underline selected literals: $\underline{\neg A} \vee B \vee C$

Literal selection functions

Unrestricted resolution is a very prolific inference system.

Use selection function to restrict applicability of rules to selected literals.

Selection function: selects a subset of literals in a clause $sel(C) \subseteq C$.

Informally: only selected literals are eligible for inferences.

A selection function *sel* is admissible if

- $sel(C) = \emptyset$ only when C is the empty clause.
- if $sel(C)$ consists of only positive literals then $sel(C)$ also contains all maximal literals in C.

We will underline selected literals: $\underline{\neg A} \vee B \vee C$

Ordered resolution with selection

Let *sel* be a selection function.

Ordered resolution with selection function *sel*, denoted $\mathbb{BRS}$, consists of the following inference rules:

- Resolution with selection rule (BRS):

$$\frac{C \lor \underline{p} \qquad \underline{\neg p} \lor D}{C \lor D}\ (BR)$$

- Ordered factoring with selection rule (BFS):

$$\frac{C \lor \underline{p} \lor \underline{p}}{C \lor p}\ (BF)$$

Applications of the inference rules are restricted to selected literals only.

Theorem. $\mathbb{BRS}$ with any admissible selection functions is complete.

Exercise Resolution with arbitrary selection is incomplete.

Ordered resolution with selection

Let *sel* be a selection function.

Ordered resolution with selection function *sel*, denoted $\mathbb{BRS}$, consists of the following inference rules:

- Resolution with selection rule (BRS):

$$\frac{C \lor \underline{p} \qquad \underline{\neg p} \lor D}{C \lor D}\ (BR)$$

- Ordered factoring with selection rule (BFS):

$$\frac{C \lor \underline{p} \lor \underline{p}}{C \lor p}\ (BF)$$

Applications of the inference rules are restricted to selected literals only.

Theorem. $\mathbb{BRS}$ with any admissible selection functions is complete.

Exercise Resolution with arbitrary selection is incomplete.

Ordered resolution with selection

Let *sel* be a selection function.

Ordered resolution with selection function *sel*, denoted $\mathbb{BRS}$, consists of the following inference rules:

- Resolution with selection rule (BRS):

$$\frac{C \vee \underline{p} \qquad \underline{\neg p} \vee D}{C \vee D}\;(BR)$$

- Ordered factoring with selection rule (BFS):

$$\frac{C \vee \underline{p} \vee \underline{p}}{C \vee p}\;(BF)$$

Applications of the inference rules are restricted to selected literals only.

Theorem. $\mathbb{BRS}$ with any admissible selection functions is complete.

Exercise Resolution with arbitrary selection is incomplete.

Redundancy elimination

Abstract notion of redundancy.

A clause C is redundant in S if there exists $\{C_1, \ldots, C_n\} \subseteq S$ such that

- $\{C_1, \ldots, C_n\} \models C$
- $C_1 \prec C, \ldots, C_n \prec C$

We can remove redundant clauses from the search space!

Practical redundancies:

- tautology elimination: $p \vee \neg p \vee C$ can be eliminated
  indeed: $\models p \vee \neg p \vee C$
- subsumption elimination: if $C \subset D$, D can be eliminated
  indeed: $C \models D$ and $C \prec D$.

Redundancy elimination

Abstract notion of redundancy.

A clause C is redundant in S if there exists $\{C_1, \ldots, C_n\} \subseteq S$ such that

- $\{C_1, \ldots, C_n\} \models C$
- $C_1 \prec C, \ldots, C_n \prec C$

We can remove redundant clauses from the search space!

Practical redundancies:

- tautology elimination: $p \vee \neg p \vee C$ can be eliminated
  indeed: $\models p \vee \neg p \vee C$
- subsumption elimination: if $C \subset D$, D can be eliminated
  indeed: $C \models D$ and $C \prec D$.

Redundancy elimination

Abstract notion of redundancy.

A clause C is redundant in S if there exists $\{C_1, \ldots, C_n\} \subseteq S$ such that

- $\{C_1, \ldots, C_n\} \models C$
- $C_1 \prec C, \ldots, C_n \prec C$

We can remove redundant clauses from the search space!

Practical redundancies:

- tautology elimination: $p \vee \neg p \vee C$ can be eliminated
  indeed: $\models p \vee \neg p \vee C$
- subsumption elimination: if $C \subset D$, D can be eliminated
  indeed: $C \models D$ and $C \prec D$.

Non-ground resolution

- A non-ground clause can be seen as representation of a (possibly infinite) set of its ground instances.
- Consider $q(x, a) \vee \underline{p(x)}$ and $q(y, z) \vee \neg \underline{p(f(y))}$.
  A common instance to which ground resolution is applicable:
  $q(f(a), a) \vee \underline{p(f(a))}$ and $q(a, a) \vee \neg \underline{p(f(a))}$
- There are other ground instances e.g.:
  $q(f(f(a)), a) \vee \underline{p(f(f(a)))}$ and $q(f(a), f(f(f(a))) \vee \neg \underline{p(f(f(a))}$
- In order to apply ground resolution we need find substitution which make atoms $\underline{p(x)}$ and $\underline{p(f(y))}$ equal.
- Such substitutions are called unifiers.

Non-ground resolution

- A non-ground clause can be seen as representation of a (possibly infinite) set of its ground instances.
- Consider $q(x, a) \lor \underline{p(x)}$ and $q(y, z) \lor \neg\underline{p(f(y))}$.
  A common instance to which ground resolution is applicable:
  $q(f(a), a) \lor \underline{p(f(a))}$ and $q(a, a) \lor \neg\underline{p(f(a))}$
- There are other ground instances e.g.:
  $q(f(f(a)), a) \lor \underline{p(f(f(a)))}$ and $q(f(a), f(f(f(a))) \lor \neg\underline{p(f(f(a))}$
- In order to apply ground resolution we need find substitution which make atoms $\underline{p(x)}$ and $\underline{p(f(y))}$ equal.
- Such substitutions are called unifiers.

Non-ground resolution

- A non-ground clause can be seen as representation of a (possibly infinite) set of its ground instances.
- Consider $q(x, a) \vee \underline{p(x)}$ and $q(y, z) \vee \neg\underline{p(f(y))}$.
  A common instance to which ground resolution is applicable:
  $q(f(a), a) \vee \underline{p(f(a))}$ and $q(a, a) \vee \neg\underline{p(f(a))}$
- There are other ground instances e.g.:
  $q(f(f(a)), a) \vee \underline{p(f(f(a)))}$ and $q(f(a), f(f(f(a)))) \vee \neg\underline{p(f(f(a)))}$
- In order to apply ground resolution we need find substitution which make atoms $\underline{p(x)}$ and $\underline{p(f(y))}$ equal.
- Such substitutions are called unifiers.

Non-ground resolution

- A non-ground clause can be seen as representation of a (possibly infinite) set of its ground instances.
- Consider $q(x, a) \vee \underline{p(x)}$ and $q(y, z) \vee \neg\underline{p(f(y))}$.
  A common instance to which ground resolution is applicable:
  $q(f(a), a) \vee \underline{p(f(a))}$ and $q(a, a) \vee \neg\underline{p(f(a))}$
- There are other ground instances e.g.:
  $q(f(f(a)), a) \vee \underline{p(f(f(a)))}$ and $q(f(a), f(f(f(a))) \vee \neg\underline{p(f(f(a))}$
- In order to apply ground resolution we need find substitution which make atoms $\underline{p(x)}$ and $\underline{p(f(y))}$ equal.
- Such substitutions are called unifiers.

Non-ground resolution

- A non-ground clause can be seen as representation of a (possibly infinite) set of its ground instances.
- Consider $q(x, a) \vee \underline{p(x)}$ and $q(y, z) \vee \neg\underline{p(f(y))}$.
  A common instance to which ground resolution is applicable:
  $q(f(a), a) \vee \underline{p(f(a))}$ and $q(a, a) \vee \neg\underline{p(f(a))}$
- There are other ground instances e.g.:
  $q(f(f(a)), a) \vee \underline{p(f(f(a)))}$ and $q(f(a), f(f(f(a))) \vee \neg\underline{p(f(f(a))}$
- In order to apply ground resolution we need find substitution which make atoms $\underline{p(x)}$ and $\underline{p(f(y))}$ equal.
- Such substitutions are called unifiers.

Unifiers

- Consider

$$E = \{s_1 \doteq t_1, \ldots, s_n \doteq t_n\}$$

  a simultaneous unification problem, where s_i and t_i are terms or atoms.

- A substitution σ is a unifier of E, if $s_i\sigma = t_i\sigma$ for each $1 \leq i \leq n$.
- If a unifier of E exists, then E is said to be unifiable.
- σ is called a simultaneous unifier of E.

Unifiers

- Consider

$$E = \{s_1 \doteq t_1, \ldots, s_n \doteq t_n\}$$

  a simultaneous unification problem, where s_i and t_i are terms or atoms.

- A substitution σ is a unifier of E, if $s_i\sigma = t_i\sigma$ for each $1 \leq i \leq n$.

- If a unifier of E exists, then E is said to be unifiable.

- σ is called a simultaneous unifier of E.

Unifiers

- Consider

$$E = \{s_1 \doteq t_1, \dots, s_n \doteq t_n\}$$

  a simultaneous unification problem, where s_i and t_i are terms or atoms.

- A substitution σ is a unifier of E, if $s_i\sigma = t_i\sigma$ for each $1 \leq i \leq n$.
- If a unifier of E exists, then E is said to be unifiable.
- σ is called a simultaneous unifier of E.

Unifiers

- Consider

$$E = \{s_1 \doteq t_1, \ldots, s_n \doteq t_n\}$$

  a simultaneous unification problem, where s_i and t_i are terms or atoms.
- A substitution σ is a unifier of E, if $s_i\sigma = t_i\sigma$ for each $1 \leq i \leq n$.
- If a unifier of E exists, then E is said to be unifiable.
- σ is called a simultaneous unifier of E.

Most general unifiers

- The most general unifier of $\sigma = \mathrm{mgu}(\{s \doteq t\})$:
  - is a unifier $s\sigma \doteq t\sigma$.
  - any other unifier is an instance of σ:

    if $\gamma : s\gamma = t\gamma$ then there is γ' such that $\gamma = \sigma\gamma'$.
- $\mathrm{mgu}(\{g(x,x) \simeq g(z,f(y))\})$ is $\sigma = \{f(y)/x, f(y)/z\}$
  - $g(x,x)\sigma \doteq g(f(y),f(y)) = g(z,f(y))\sigma$
  - any other unifier γ that makes $g(x,x)\gamma = g(z,f(y))\gamma$ e.g.

    $\gamma = \{f(f(a))/x, f(f(a))/z\}$ is an instance of σ: $\gamma = \sigma \cdot \{f(a)/y\}$

Theorem [Robinson 1965] For any unifiable system of equations $E = \{s_1 \doteq t_1, \ldots, s_n \doteq t_n\}$ there is the most general unifier $\mathrm{mgu}(E)$, which is unique up to renaming.

Most general unifiers

- The most general unifier of $\sigma = \mathrm{mgu}(\{s \doteq t\})$:
  - is a unifier $s\sigma \doteq t\sigma$.
  - any other unifier is an instance of σ:

    if $\gamma : s\gamma = t\gamma$ then there is γ' such that $\gamma = \sigma\gamma'$.
- $\mathrm{mgu}(\{g(x,x) \simeq g(z,f(y))\})$ is $\sigma = \{f(y)/x, f(y)/z\}$
  - $g(x,x)\sigma \doteq g(f(y),f(y)) = g(z,f(y))\sigma$
  - any other unifier γ that makes $g(x,x)\gamma = g(z,f(y))\gamma$ e.g.

    $\gamma = \{f(f(a))/x, f(f(a))/z\}$ is an instance of σ: $\gamma = \sigma \cdot \{f(a)/y\}$

Theorem [Robinson 1965] For any unifiable system of equations $E = \{s_1 \doteq t_1, \ldots, s_n \doteq t_n\}$ there is the most general unifier $\mathrm{mgu}(E)$, which is unique up to renaming.

Most general unifiers

- The most general unifier of $\sigma = \mathrm{mgu}(\{s \doteq t\})$:
  - is a unifier $s\sigma \doteq t\sigma$.
  - any other unifier is an instance of σ:

    if $\gamma : s\gamma = t\gamma$ then there is γ' such that $\gamma = \sigma\gamma'$.
- $\mathrm{mgu}(\{g(x,x) \simeq g(z,f(y))\})$ is $\sigma = \{f(y)/x, f(y)/z\}$
  - $g(x,x)\sigma \doteq g(f(y),f(y)) = g(z,f(y))\sigma$
  - any other unifier γ that makes $g(x,x)\gamma = g(z,f(y))\gamma$ e.g.

    $\gamma = \{f(f(a))/x, f(f(a))/z\}$ is an instance of σ: $\gamma = \sigma \cdot \{f(a)/y\}$

Theorem [Robinson 1965] For any unifiable system of equations $E = \{s_1 \doteq t_1, \ldots, s_n \doteq t_n\}$ there is the most general unifier $\mathrm{mgu}(E)$, which is unique up to renaming.

Most general unifiers

- The most general unifier of $\sigma = \mathrm{mgu}(\{s \doteq t\})$:
  - is a unifier $s\sigma \doteq t\sigma$.
  - any other unifier is an instance of σ:

    if $\gamma : s\gamma = t\gamma$ then there is γ' such that $\gamma = \sigma\gamma'$.
- $\mathrm{mgu}(\{g(x,x) \simeq g(z, f(y))\})$ is $\sigma = \{f(y)/x, f(y)/z\}$
  - $g(x,x)\sigma \doteq g(f(y), f(y)) = g(z, f(y))\sigma$
  - any other unifier γ that makes $g(x,x)\gamma = g(z, f(y))\gamma$ e.g.

    $\gamma = \{f(f(a))/x, f(f(a))/z\}$ is an instance of σ: $\gamma = \sigma \cdot \{f(a)/y\}$

Theorem [Robinson 1965] For any unifiable system of equations $E = \{s_1 \doteq t_1, \dots, s_n \doteq t_n\}$ there is the most general unifier $\mathrm{mgu}(E)$, which is unique up to renaming.

Unification algorithm:

Apply unification transformation rules to E to obtain $\mathrm{mgu}(E)$.

- Orientation: $t \doteq x,\ E \Rightarrow_U\ x \doteq t,\ E$ if $t \notin \mathcal{X}$
- Trivial: $t \doteq t,\ E\ \Rightarrow_U\ E$
- Clash: $f(\ldots) \doteq g(\ldots),\ E\ \Rightarrow_U\ \bot$
- Decomposition:
  $f(s_1, \ldots, s_n) \doteq f(t_1, \ldots, t_n),\ E\ \Rightarrow_U$
  $s_1 \doteq t_1,\ \ldots,\ s_n \doteq t_n,\ E$
- Occur-check: $x \doteq t,\ E\ \Rightarrow_U\ \bot$
  if $x \in var(t)$, $x \neq t$
- Substitution: $x \doteq t,\ E\ \Rightarrow_U\ x \doteq t,\ E\{t \mapsto x\}$
  if $x \in var(E)$, $x \notin var(t)$

General resolution with selection:

- Resolution rule (BRS):

$$\frac{C \vee p \qquad \neg p' \vee D}{(C \vee D)\sigma} \; (BR)$$

where $\sigma = \mathrm{mgu}(p, p')$

- Binary positive factoring (BFS):

$$\frac{C \vee p \vee p'}{(C \vee p)\sigma} \; (BF)$$

where $\sigma = \mathrm{mgu}(p, p')$

Ordered resolution with selection:

Extend $\succ$ from order on ground atoms to any order $\succ'$ on (non-ground) atoms:

- requirement (stability under substitutions)
  if $A(\bar{x}) \succ B(\bar{x})$ then for every ground substitution γ :
  $A(\bar{x})\gamma \succ' Q(\bar{x})\gamma$.

General resolution with selection:

- Resolution rule (BRS):

$$\frac{C \vee p \qquad \neg p' \vee D}{(C \vee D)\sigma} \; (BR)$$

  where $\sigma = \mathrm{mgu}(p, p')$

- Binary positive factoring (BFS):

$$\frac{C \vee p \vee p'}{(C \vee p)\sigma} \; (BF)$$

  where $\sigma = \mathrm{mgu}(p, p')$

Ordered resolution with selection:

Extend $\succ$ from order on ground atoms to any order $\succ'$ on (non-ground) atoms:

- requirement (stability under substitutions)
  if $A(\bar{x}) \succ B(\bar{x})$ then for every ground substitution γ :
  $A(\bar{x})\gamma \succ' Q(\bar{x})\gamma$.

General resolution with selection:

- Resolution rule (BRS):

$$\frac{C \vee \underline{p} \qquad \underline{\neg p'} \vee D}{(C \vee D)\sigma} \ (BR)$$

  where $\sigma = \mathrm{mgu}(p, p')$

- Binary positive factoring (BFS):

$$\frac{C \vee \underline{p} \vee \underline{p'}}{(C \vee p)\sigma} \ (BF)$$

  where $\sigma = \mathrm{mgu}(p, p')$

Ordered resolution with selection:

Extend $\succ$ from order on ground atoms to any order $\succ'$ on (non-ground) atoms:

- requirement (stability under substitutions)
  if $A(\bar{x}) \succ B(\bar{x})$ then for every ground substitution γ :
  $A(\bar{x})\gamma \succ' Q(\bar{x})\gamma$.

Completeness of resolution in the general case

Theorem. $\mathbb{BRS}$ with any admissible selection functions is complete for general first-order clauses.

Proof. Consider a set of first-order clauses S.

Need to show: If S is saturated and $\square \notin S$ then S is satisfiable.

Lifting argument: $Gr(S)$ is also saturated and does not contain $\square$.
Indeed for any inference by ground resolution in $Gr(S)$ there is more general non-ground inference in S.

Therefore $Gr(S)$ is satisfiable on a Herbrand model I_S.
Finally $I_S \models S$.

Completeness of resolution in the general case

Theorem. $\mathbb{BRS}$ with any admissible selection functions is complete for general first-order clauses.

Proof. Consider a set of first-order clauses S.
Need to show: If S is saturated and $\square \notin S$ then S is satisfiable.

Lifting argument: $Gr(S)$ is also saturated and does not contain $\square$.
Indeed for any inference by ground resolution in $Gr(S)$ there is more general non-ground inference in S.

Therefore $Gr(S)$ is satisfiable on a Herbrand model I_S.
Finally $I_S \models S$.

Completeness of resolution in the general case

Theorem. $\mathbb{BRS}$ with any admissible selection functions is complete for general first-order clauses.

Proof. Consider a set of first-order clauses S.

Need to show: If S is saturated and $\square \notin S$ then S is satisfiable.

Lifting argument: $Gr(S)$ is also saturated and does not contain $\square$.
Indeed for any inference by ground resolution in $Gr(S)$ there is more general non-ground inference in S.

Therefore $Gr(S)$ is satisfiable on a Herbrand model I_S.
Finally $I_S \models S$.

Completeness of resolution in the general case

Theorem. $\mathbb{BRS}$ with any admissible selection functions is complete for general first-order clauses.

Proof. Consider a set of first-order clauses S.

Need to show: If S is saturated and $\Box \notin S$ then S is satisfiable.

Lifting argument: $Gr(S)$ is also saturated and does not contain $\Box$.
Indeed for any inference by ground resolution in $Gr(S)$ there is more general non-ground inference in S.

Therefore $Gr(S)$ is satisfiable on a Herbrand model I_S.
Finally $I_S \models S$.

Completeness of resolution in the general case

Theorem. $\mathbb{BRS}$ with any admissible selection functions is complete for general first-order clauses.

Proof. Consider a set of first-order clauses S.

Need to show: If S is saturated and $\square \notin S$ then S is satisfiable.

Lifting argument: $Gr(S)$ is also saturated and does not contain $\square$.

Indeed for any inference by ground resolution in $Gr(S)$ there is more general non-ground inference in S.

Therefore $Gr(S)$ is satisfiable on a Herbrand model I_S.

Finally $I_S \models S$.

Completeness of resolution in the general case

Theorem. $\mathbb{BRS}$ with any admissible selection functions is complete for general first-order clauses.

Proof. Consider a set of first-order clauses S.

Need to show: If S is saturated and $\square \notin S$ then S is satisfiable.

Lifting argument: $Gr(S)$ is also saturated and does not contain $\square$.
Indeed for any inference by ground resolution in $Gr(S)$ there is more general non-ground inference in S.

Therefore $Gr(S)$ is satisfiable on a Herbrand model I_S.
Finally $I_S \models S$.

Completeness of resolution in the general case

Theorem. $\mathbb{BRS}$ with any admissible selection functions is complete for general first-order clauses.

Proof. Consider a set of first-order clauses S.

Need to show: If S is saturated and $\square \notin S$ then S is satisfiable.

Lifting argument: $Gr(S)$ is also saturated and does not contain $\square$.

Indeed for any inference by ground resolution in $Gr(S)$ there is more general non-ground inference in S.

Therefore $Gr(S)$ is satisfiable on a Herbrand model I_S.

Finally $I_S \models S$.

Resolution as a decision procedure

Consider a fair saturation process by a sound and complete calculi $\mathcal{C}$

$$S_0 \Rightarrow S_1 \Rightarrow \ldots S_n \Rightarrow \ldots$$

There are three possible outcomes:

1. $\Box$ is derived ($\Box \in S_n$ for some n), then S is unsatisfiable (soundness);
2. no new clauses can be derived from S_i, i. e. $Res(S_i) \subseteq S_i$, for some $0 \leq i < \omega$ and $\Box \notin S$, then S is satisfiable (completeness);
3. S grows ad infinitum, the process does not terminate, in this case S is satisfiable (completeness).

In cases 1) and 2) the procedure terminates.

A sound and complete calculus $\mathcal{C}$ together with a fair saturation strategy is a decision procedure for a fragment Φ if the saturation process terminates for any clause set in Φ.

Resolution as a decision procedure

Consider a fair saturation process by a sound and complete calculi $\mathcal{C}$

$$S_0 \Rightarrow S_1 \Rightarrow \ldots S_n \Rightarrow \ldots$$

There are three possible outcomes:

1. $\Box$ is derived ($\Box \in S_n$ for some n), then S is unsatisfiable (soundness);
2. no new clauses can be derived from S_i, i. e. $Res(S_i) \subseteq S_i$, for some $0 \leq i < \omega$ and $\Box \notin S$, then S is satisfiable (completeness);
3. S grows ad infinitum, the process does not terminate, in this case S is satisfiable (completeness).

In cases 1) and 2) the procedure terminates.
A sound and complete calculus $\mathcal{C}$ together with a fair saturation strategy is a decision procedure for a fragment Φ if the saturation process terminates for any clause set in Φ.

The magic of resolution

Resolution calculus with appropriate simplifications, selection functions and saturation strategies is a decision procedure for many fragments:

- monadic fragment [Bachmair, Ganzinger, Waldmann]
- modal logic translations [Hustadt, Schmidt]
- guarded fragment [Ganzinger, de Nivelle]
- two variable fragment [de Nivelle, Pratt-Hartmann]
- fluted fragment [Hustadt, Schmidt, Georgieva]
- many description logic fragments [Kazakov, Motik, Sattler, ...]
- ...

- Original proofs of decidability for these fragments are based on diverse, complicated, model theoretic arguments.
- One can speculate that the model construction in the proof of completeness of resolution unifies these model theoretic arguments.
- Resolution-based methods provide practical procedures

The magic of resolution

Resolution calculus with appropriate simplifications, selection functions and saturation strategies is a decision procedure for many fragments:

- monadic fragment [Bachmair, Ganzinger, Waldmann]
- modal logic translations [Hustadt, Schmidt]
- guarded fragment [Ganzinger, de Nivelle]
- two variable fragment [de Nivelle, Pratt-Hartmann]
- fluted fragment [Hustadt, Schmidt, Georgieva]
- many description logic fragments [Kazakov, Motik, Sattler, ...]
- ...

- Original proofs of decidability for these fragments are based on diverse, complicated, model theoretic arguments.
- One can speculate that the model construction in the proof of completeness of resolution unifies these model theoretic arguments.
- Resolution-based methods provide practical procedures

The magic of resolution

Resolution calculus with appropriate simplifications, selection functions and saturation strategies is a decision procedure for many fragments:

- monadic fragment [Bachmair, Ganzinger, Waldmann]
- modal logic translations [Hustadt, Schmidt]
- guarded fragment [Ganzinger, de Nivelle]
- two variable fragment [de Nivelle, Pratt-Hartmann]
- fluted fragment [Hustadt, Schmidt, Georgieva]
- many description logic fragments [Kazakov, Motik, Sattler, ...]
- ...

- Original proofs of decidability for these fragments are based on diverse, complicated, model theoretic arguments.
- One can speculate that the model construction in the proof of completeness of resolution unifies these model theoretic arguments.
- Resolution-based methods provide practical procedures

The magic of resolution

Resolution calculus with appropriate simplifications, selection functions and saturation strategies is a decision procedure for many fragments:

- monadic fragment [Bachmair, Ganzinger, Waldmann]
- modal logic translations [Hustadt, Schmidt]
- guarded fragment [Ganzinger, de Nivelle]
- two variable fragment [de Nivelle, Pratt-Hartmann]
- fluted fragment [Hustadt, Schmidt, Georgieva]
- many description logic fragments [Kazakov, Motik, Sattler, ...]
- ...

- Original proofs of decidability for these fragments are based on diverse, complicated, model theoretic arguments.
- One can speculate that the model construction in the proof of completeness of resolution unifies these model theoretic arguments.
- Resolution-based methods provide practical procedures

The magic of resolution

Resolution calculus with appropriate simplifications, selection functions and saturation strategies is a decision procedure for many fragments:

- monadic fragment [Bachmair, Ganzinger, Waldmann]
- modal logic translations [Hustadt, Schmidt]
- guarded fragment [Ganzinger, de Nivelle]
- two variable fragment [de Nivelle, Pratt-Hartmann]
- fluted fragment [Hustadt, Schmidt, Georgieva]
- many description logic fragments [Kazakov, Motik, Sattler, ...]
- ...

- Original proofs of decidability for these fragments are based on diverse, complicated, model theoretic arguments.
- One can speculate that the model construction in the proof of completeness of resolution unifies these model theoretic arguments.
- Resolution-based methods provide practical procedures

The magic of resolution

Resolution calculus with appropriate simplifications, selection functions and saturation strategies is a decision procedure for many fragments:

- monadic fragment [Bachmair, Ganzinger, Waldmann]
- modal logic translations [Hustadt, Schmidt]
- guarded fragment [Ganzinger, de Nivelle]
- two variable fragment [de Nivelle, Pratt-Hartmann]
- fluted fragment [Hustadt, Schmidt, Georgieva]
- many description logic fragments [Kazakov, Motik, Sattler, ...]
- ...

- Original proofs of decidability for these fragments are based on diverse, complicated, model theoretic arguments.
- One can speculate that the model construction in the proof of completeness of resolution unifies these model theoretic arguments.
- Resolution-based methods provide practical procedures

The magic of resolution

Resolution calculus with appropriate simplifications, selection functions and saturation strategies is a decision procedure for many fragments:

- monadic fragment [Bachmair, Ganzinger, Waldmann]
- modal logic translations [Hustadt, Schmidt]
- guarded fragment [Ganzinger, de Nivelle]
- two variable fragment [de Nivelle, Pratt-Hartmann]
- fluted fragment [Hustadt, Schmidt, Georgieva]
- many description logic fragments [Kazakov, Motik, Sattler, . . .]
- . . .

- Original proofs of decidability for these fragments are based on diverse, complicated, model theoretic arguments.
- One can speculate that the model construction in the proof of completeness of resolution unifies these model theoretic arguments.
- Resolution-based methods provide practical procedures

The magic of resolution

Resolution calculus with appropriate simplifications, selection functions and saturation strategies is a decision procedure for many fragments:

- monadic fragment [Bachmair, Ganzinger, Waldmann]
- modal logic translations [Hustadt, Schmidt]
- guarded fragment [Ganzinger, de Nivelle]
- two variable fragment [de Nivelle, Pratt-Hartmann]
- fluted fragment [Hustadt, Schmidt, Georgieva]
- many description logic fragments [Kazakov, Motik, Sattler, ...]
- ...

- Original proofs of decidability for these fragments are based on diverse, complicated, model theoretic arguments.
- One can speculate that the model construction in the proof of completeness of resolution unifies these model theoretic arguments.
- Resolution-based methods provide practical procedures

The magic of resolution

Resolution calculus with appropriate simplifications, selection functions and saturation strategies is a decision procedure for many fragments:

- monadic fragment [Bachmair, Ganzinger, Waldmann]
- modal logic translations [Hustadt, Schmidt]
- guarded fragment [Ganzinger, de Nivelle]
- two variable fragment [de Nivelle, Pratt-Hartmann]
- fluted fragment [Hustadt, Schmidt, Georgieva]
- many description logic fragments [Kazakov, Motik, Sattler, ...]
- ...

- Original proofs of decidability for these fragments are based on diverse, complicated, model theoretic arguments.
- One can speculate that the model construction in the proof of completeness of resolution unifies these model theoretic arguments.
- Resolution-based methods provide practical procedures