Automated Reasoning

Resolution Theorem Proving

Temur Kutsia

RISC, Johannes Kepler University, Linz, Austria
kutsia@risc.jku.at

What is Automated Reasoning

Reasoning: The process of making inferences.

Automated reasoning studies methods to automate the process of reasoning.

Automated reasoning systems: computer programs that implement automated reasoning methods to perform reasoning automatically (or semi-automatically).

Examples of Reasoning

All men are mortal. Socrates is a man. Therefore Socrates is mortal.

Examples of Reasoning

All men are mortal. Socrates is a man. Therefore Socrates is mortal.

Every fruit is tasty if it is not cooked. This apple not tasty. Therefore, it is cooked.

Do You Agree with These Reasonings?

All that glistens is not gold. This pot does not glisten.
Therefore, it is gold.

Do You Agree with These Reasonings?

All that glistens is not gold. This pot does not glisten.
Therefore, it is gold.

All numbers are odd. 2 is not odd. Therefore, 2 is not a number.

Do You Agree with These Reasonings?

All that glistens is not gold. This pot does not glisten. Therefore, it is gold.

All numbers are odd. 2 is not odd. Therefore, 2 is not a number.

All numbers are odd. 2 is even. Therefore, 2 is not a number.

Do You Agree with These Reasonings?

All that glistens is not gold. This pot does not glisten. Therefore, it is gold.

All numbers are odd. 2 is not odd. Therefore, 2 is not a number.

All numbers are odd. 2 is even. Therefore, 2 is not a number.

Some people are geniuses. Einstein is a person. Therefore, Einstein is a genius.

Are These Statements True?

There exists a person with the property that if he (or she) is a genius then everybody is a genius.

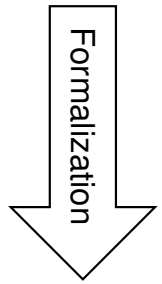
If a group satisfies the identity $x^2 = 1$, then it is commutative.

General Picture

Natural language, mathematical problems, program + specification, ...

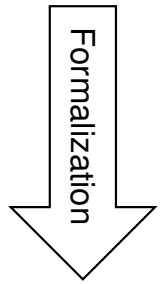
General Picture

Natural language, mathematical problems, program + specification, ...



General Picture

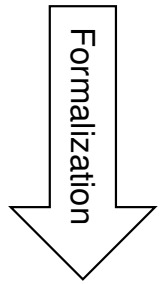
Natural language, mathematical problems, program + specification, ...



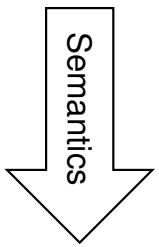
Formal language: FOL, HOL, temporal logic, etc.

General Picture

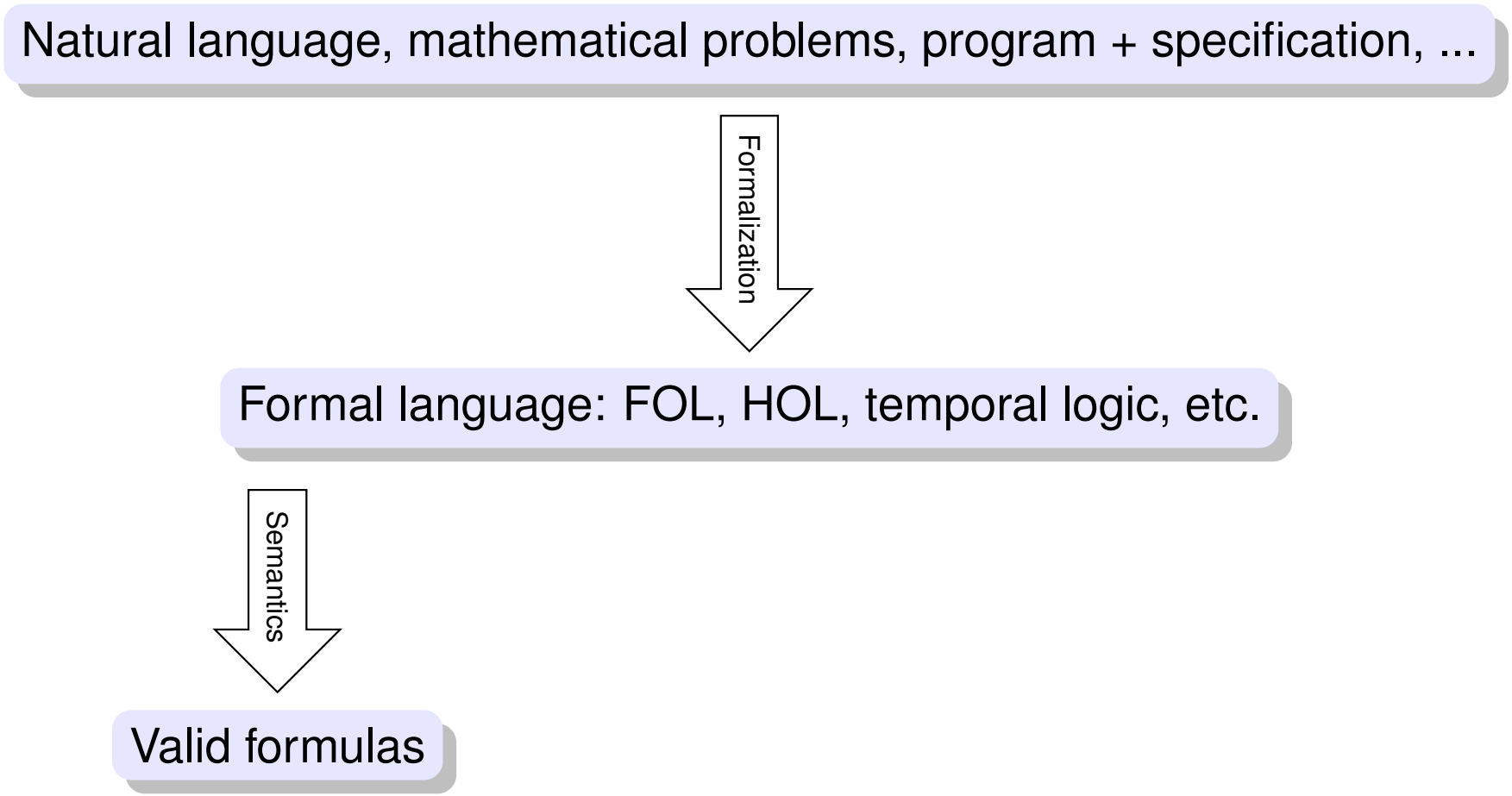
Natural language, mathematical problems, program + specification, ...



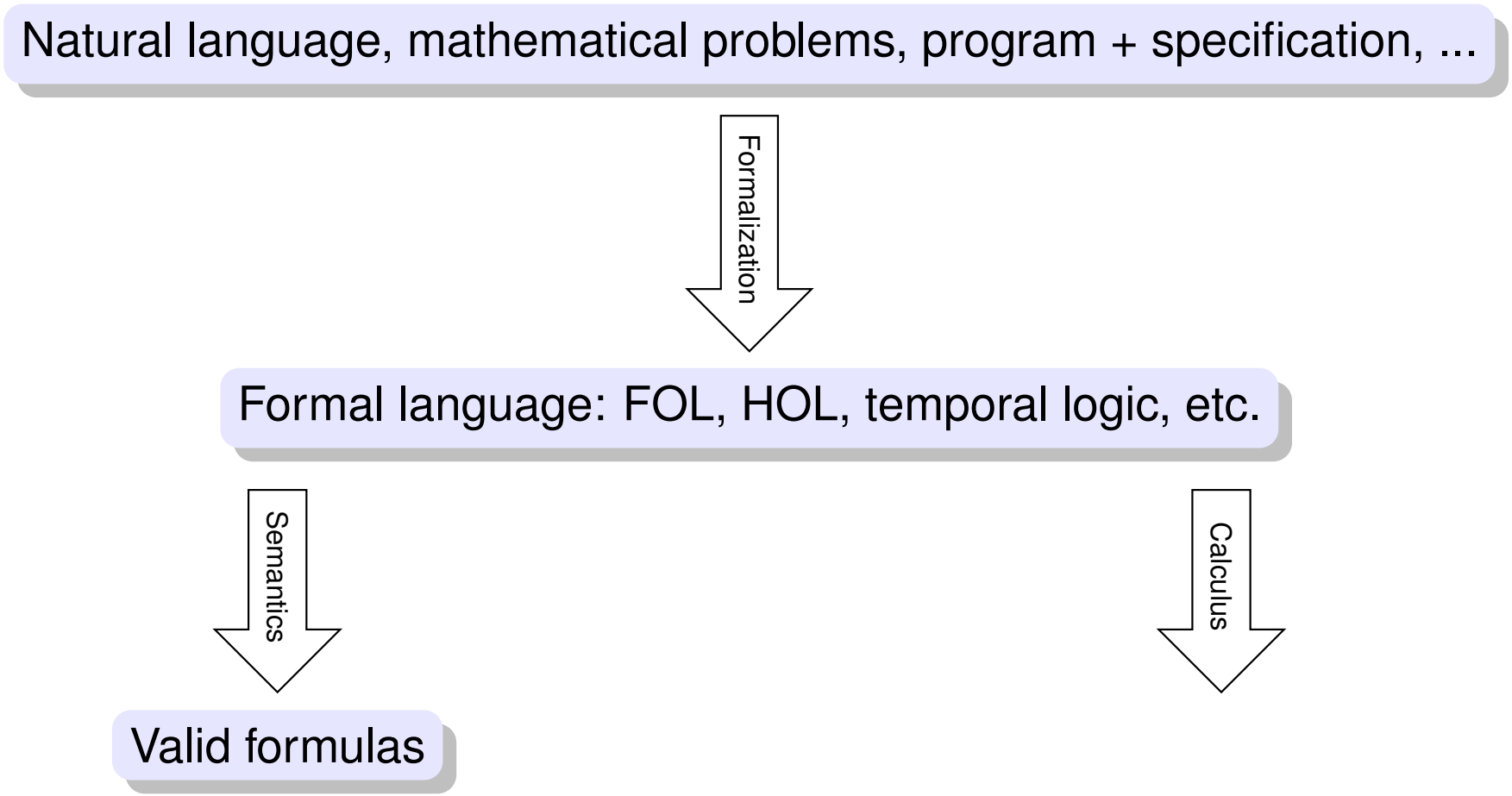
Formal language: FOL, HOL, temporal logic, etc.



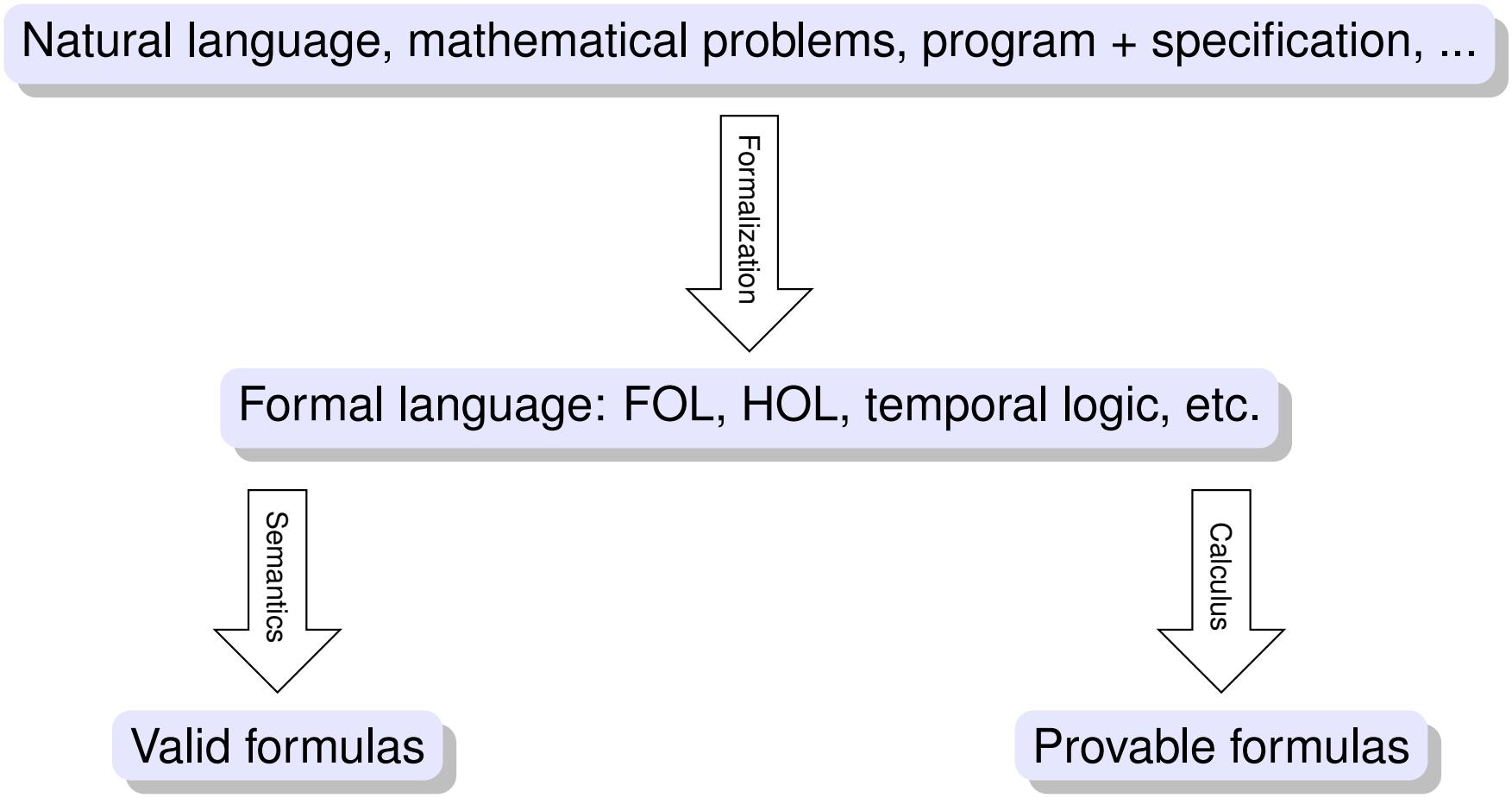
General Picture



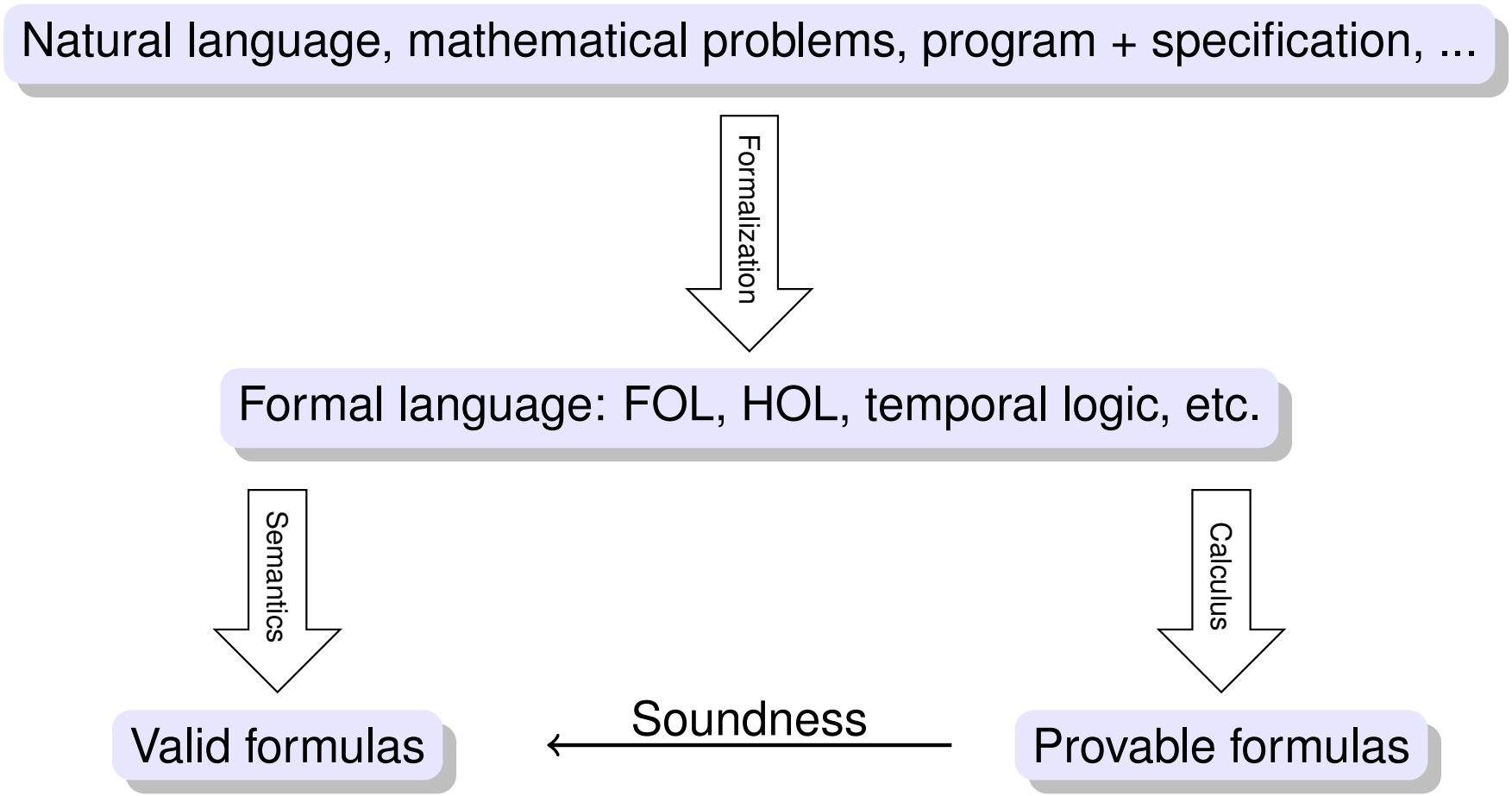
General Picture



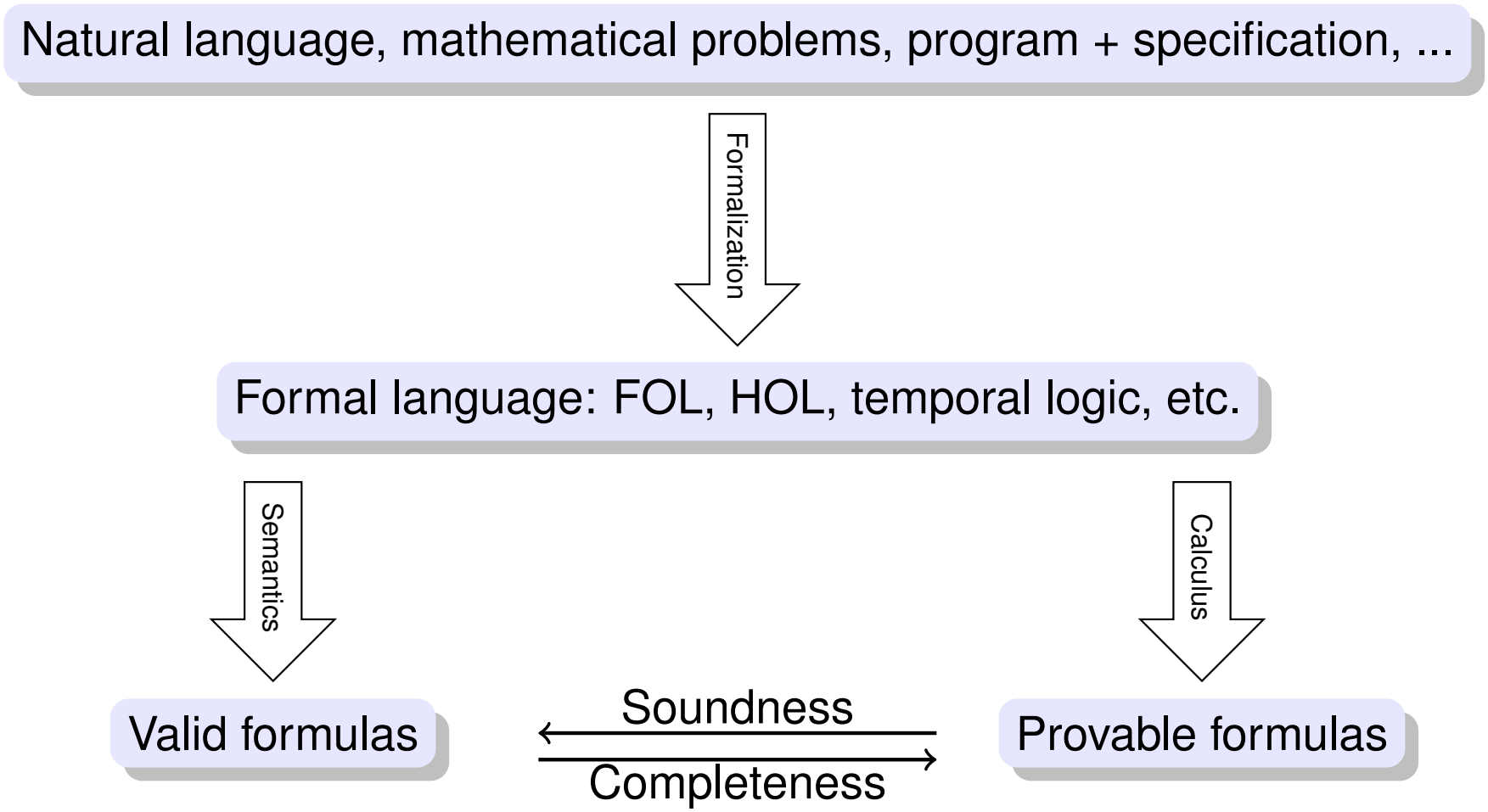
General Picture



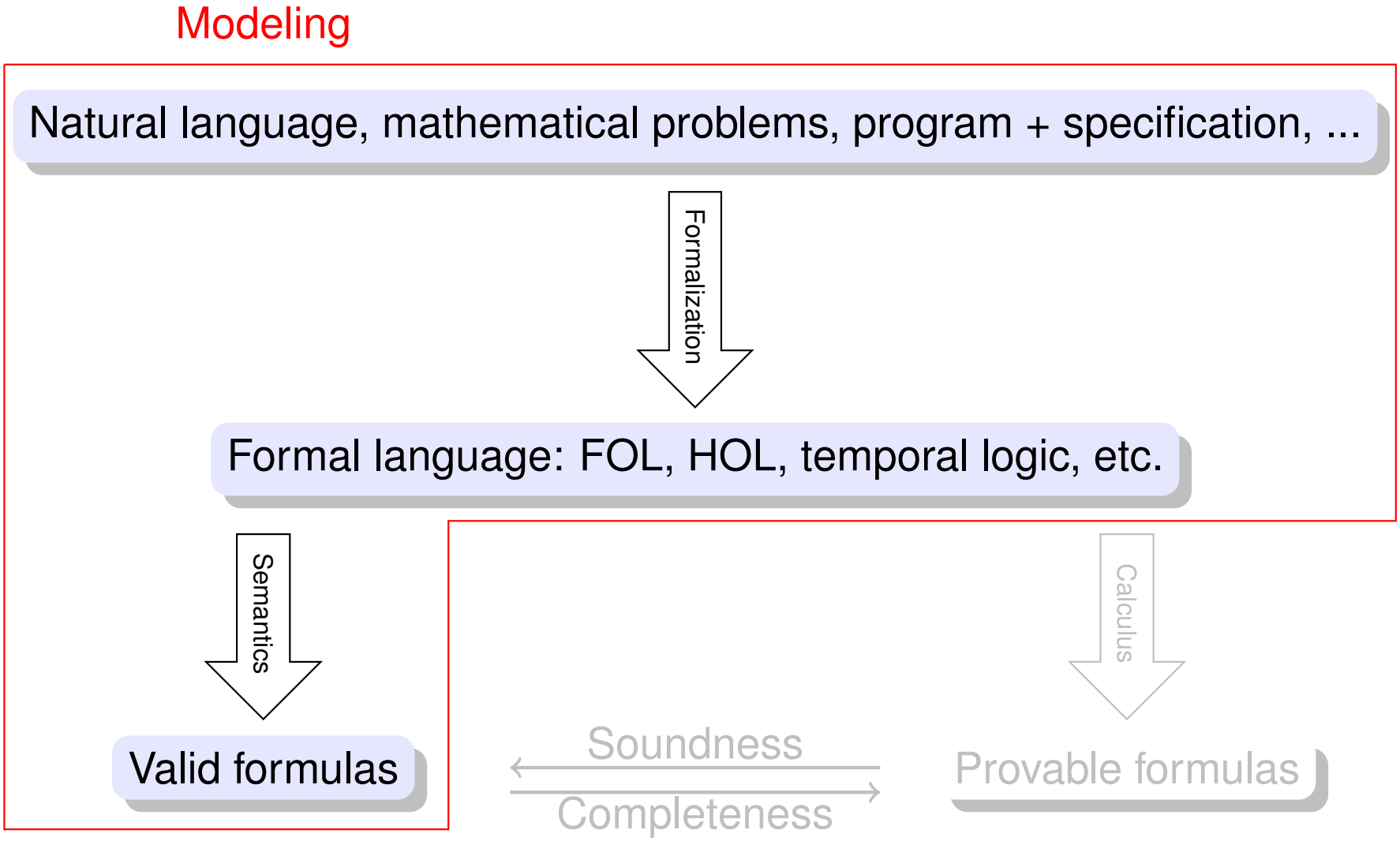
General Picture



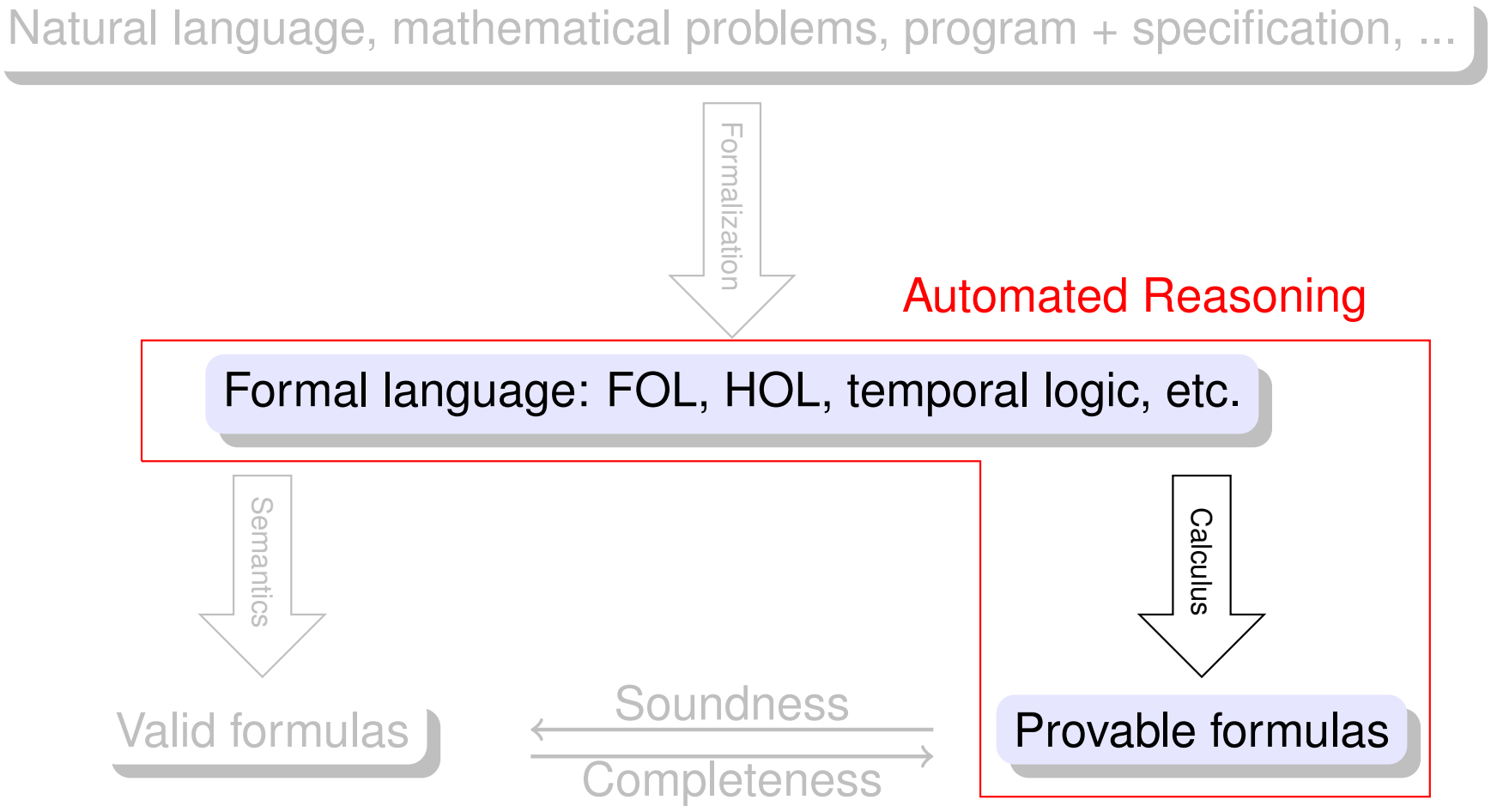
General Picture



General Picture



General Picture



Informal Example

Problem formulation (Chang and Lee, 1973):

> *Suppose that stock prices go down if the prime interest rate goes up. Suppose also that most people are unhappy when stock prices go down. Assume that prime interest rate does go up. Are most people unhappy?*

Informal Example

Problem formulation (Chang and Lee, 1973):

> *Suppose that stock prices go down if the prime interest rate goes up. Suppose also that most people are unhappy when stock prices go down. Assume that prime interest rate does go up. Are most people unhappy?*

Formalization:

- P : prime interest rate goes up.
- S : stock prices go down.
- U : most people are unhappy.
- If the prime interest rate goes up, stock prices go down: $\mathrm{P} \Rightarrow \mathrm{S}$.
- If stock prices go down, most people are unhappy: $\mathrm{S} \Rightarrow \mathrm{U}$.

Show that if $\mathrm{P} \Rightarrow \mathrm{S}$, $\mathrm{S} \Rightarrow \mathrm{U}$, and P hold, then U holds as well.

Informal Example

We should show that if $P \Rightarrow S$, $S \Rightarrow U$, and P hold, then U holds as well.

That means, $((P \Rightarrow S) \wedge (S \Rightarrow U) \wedge P) \Rightarrow U$ is valid.

Denote $((P \Rightarrow S) \wedge (S \Rightarrow U) \wedge P) \Rightarrow U$ by $\mathbf{G}$.

Semantically:

P	S	U	$P \Rightarrow S$	$S \Rightarrow U$	$\mathbf{G}$
true	true	true	true	true	true
true	true	false	true	false	true
true	false	true	false	true	true
true	false	false	false	true	true
false	true	true	true	true	true
false	false	true	true	true	true
false	true	false	true	false	true
false	false	false	true	true	true

Informal Example

In the example we used propositional logic.

Often we need more powerful logics.

For instance, we need first-order logic to express the Socrates example:

Informal Example

In the example we used propositional logic.

Often we need more powerful logics.

For instance, we need first-order logic to express the Socrates example:

- $\forall x.man(x) \Rightarrow mortal(x)$: All men are mortal.
- $man(socrates)$: Socrates is a man.
- $mortal(socrates)$: Socrates is mortal.

First-Order Logic

- Syntax
- Semantics
- Inference system

Syntax

- Alphabet
- Terms
- Formulas

Alphabet

A first-order alphabet consists of the following sets of symbols:

- A countable set of variables $\mathcal{V}$.
- For each $\mathfrak{n} \geqslant 0$, a set of $\mathfrak{n}$-ary function symbols $\mathcal{F}^{\mathfrak{n}}$. Elements of $\mathcal{F}^0$ are called constants.
- For each $\mathfrak{n} \geqslant 0$, a set of $\mathfrak{n}$-ary predicate symbols $\mathcal{P}^{\mathfrak{n}}$.
- Logical connectives $\neg$, $\vee$, $\wedge$, $\Rightarrow$, $\Leftrightarrow$.
- Quantifiers $\exists$, $\forall$.
- Parentheses and comma.

Alphabet

Notation:

- $\mathrm{x}, \mathrm{y}, \mathrm{z}$ for variables.
- f, g for function symbols.
- $\mathrm{a}, \mathrm{b}, \mathrm{c}$ for constants.
- p, q for predicate symbols.

Terms

Definition

- A variable is a term.
- If $\mathrm{t}_1, \ldots, \mathrm{t}_n$ are terms and $\mathrm{f} \in \mathcal{F}^n$, then $\mathrm{f}(\mathrm{t}_1, \ldots, \mathrm{t}_n)$ is a term.

Terms

Definition

- A variable is a term.
- If $\mathrm{t}_1, \ldots, \mathrm{t}_n$ are terms and $\mathrm{f} \in \mathcal{F}^n$, then $\mathrm{f}(\mathrm{t}_1, \ldots, \mathrm{t}_n)$ is a term.

Notation:

- $s, \mathrm{t}, \mathrm{r}$ for terms.

Terms

Definition

- A variable is a term.
- If $\mathrm{t}_1, \ldots, \mathrm{t}_n$ are terms and $\mathrm{f} \in \mathcal{F}^n$, then $\mathrm{f}(\mathrm{t}_1, \ldots, \mathrm{t}_n)$ is a term.

Notation:

- $s, \mathrm{t}, \mathrm{r}$ for terms.

Ground term: a term without variables.

Terms

Example

- $\mathrm{plus}(\mathrm{plus}(\mathrm{x}, 1), \mathrm{x})$ is a non-ground term, if plus is a binary function symbol, 1 is a constant, x is a variable.

Terms

Example

- $\mathrm{plus}(\mathrm{plus}(\mathrm{x}, 1), \mathrm{x})$ is a non-ground term, if plus is a binary function symbol, 1 is a constant, x is a variable.
- $\mathrm{father}(\mathrm{father}(\mathrm{John}))$ is a ground term, if father is a unary function symbol and John is a constant.

Formulas

Definition

- If $\mathrm{t}_1, \ldots, \mathrm{t}_n$ are terms and $\mathrm{p} \in \mathcal{P}^n$, then $\mathrm{p}(\mathrm{t}_1, \ldots, \mathrm{t}_n)$ is a formula. It is called an atomic formula or an atom.
- If A is a formula, $\neg(\mathrm{A})$ is a formula.
- If A and B are formulas, then $(\mathrm{A} \vee \mathrm{B})$, $(\mathrm{A} \wedge \mathrm{B})$, $(\mathrm{A} \Rightarrow \mathrm{B})$, and $(\mathrm{A} \Leftrightarrow \mathrm{B})$ are formulas.
- If A is a formula, then $\exists \mathrm{x}.\mathrm{A}$ and $\forall \mathrm{x}.\mathrm{A}$ are formulas.

Formulas

Definition

- If $t_1, \ldots, t_n$ are terms and $p \in \mathcal{P}^n$, then $p(t_1, \ldots, t_n)$ is a formula. It is called an atomic formula or an atom.
- If A is a formula, $\neg(A)$ is a formula.
- If A and B are formulas, then $(A \vee B)$, $(A \wedge B)$, $(A \Rightarrow B)$, and $(A \Leftrightarrow B)$ are formulas.
- If A is a formula, then $\exists x.A$ and $\forall x.A$ are formulas.

Notation:

- A, B, F, G, H for formulas.

Example

Translating English sentences into first-order logic formulas:

For each natural number there exists exactly one immediate successor natural number.

Assume:

- $succ$: unary function symbol for immediate successor.
- $\doteq$: binary predicate symbol for equality.

Example

Translating English sentences into first-order logic formulas:

For each natural number there exists exactly one immediate successor natural number.

$$\forall \mathrm{x}.(\exists \mathrm{y}.(\mathrm{y} \doteq \mathrm{succ}(\mathrm{x}) \wedge \forall z.(z \doteq \mathrm{succ}(\mathrm{x}) \Rightarrow \mathrm{y} \doteq z)))$$

Assume:

- succ: unary function symbol for immediate successor.
- $\doteq$: binary predicate symbol for equality.

Example

Translating English sentences into first-order logic formulas:

There is no natural number whose immediate successor is 0.

Assume:

- $zero$: constant for 0.
- $succ$: unary function symbol for immediate successor.
- $\doteq$: binary predicate symbol for equality.

Example

Translating English sentences into first-order logic formulas:

There is no natural number whose immediate successor is 0.

$$\neg\exists x.\, zero \doteq succ(x)$$

Assume:

- $zero$: constant for 0.
- $succ$: unary function symbol for immediate successor.
- $\doteq$: binary predicate symbol for equality.

Example

Translating English sentences into first-order logic formulas:

For each nonzero natural number there exists exactly one immediate predecessor natural number.

Assume:

- $zero$: constant for 0.
- $pred$: unary function symbol for predecessor.
- $\doteq$: binary predicate symbol for equality.

Example

Translating English sentences into first-order logic formulas:

For each nonzero natural number there exists exactly one immediate predecessor natural number.

$$\forall x.(\neg(x \doteq 0) \Rightarrow \exists y.(y \doteq \mathrm{pred}(x) \wedge \forall z.(z \doteq \mathrm{pred}(x) \Rightarrow y \doteq z)))$$

Assume:

- $zero$: constant for 0.
- pred: unary function symbol for predecessor.
- $\doteq$: binary predicate symbol for equality.

Free and Bound Variables

A is the scope of a quantifier $\mathrm{Q}x$ in $\mathrm{Q}x.A$, $\mathrm{Q} \in \{\forall, \exists\}$.

An occurrence of a variable x in a formula is **bound**, if it is in the scope of a quantifier $\mathrm{Q}x$.

Any other occurrence of a variable in a formula is **free**.

Free and Bound Variables

A is the scope of a quantifier Qx in $Qx.A$, $Q \in \{\forall, \exists\}$.

An occurrence of a variable x in a formula is **bound**, if it is in the scope of a quantifier Qx.

Any other occurrence of a variable in a formula is **free**.

In $\forall x.p(x, y) \wedge \exists y.q(y)$, the occurrence of x and the second occurrence of y are bound, the first occurrence of y is free.

Free and Bound Variables

A is the scope of a quantifier Qx in $Qx.A$, $Q \in \{\forall, \exists\}$.

An occurrence of a variable x in a formula is **bound**, if it is in the scope of a quantifier Qx.

Any other occurrence of a variable in a formula is **free**.

In $\forall x.p(x, y) \wedge \exists y.q(y)$, the occurrence of x and the second occurrence of y are bound, the first occurrence of y is free.

Formula without free occurrences of variables is called **closed**.

Substitutions

Substitution: A function σ from variables to terms, whose domain

$$Dom(\sigma) := \{x \mid \sigma(x) \neq x\}$$

is finite.

Substitutions

Substitution: A function σ from variables to terms, whose domain

$$Dom(\sigma) := \{x \mid \sigma(x) \neq x\}$$

is finite.

Range of a substitution σ:

$$Ran(\sigma) := \{\sigma(x) \mid x \in Dom(\sigma)\}.$$

Substitutions

Substitution: A function σ from variables to terms, whose domain

$$Dom(\sigma) := \{x \mid \sigma(x) \neq x\}$$

is finite.

Range of a substitution σ:

$$Ran(\sigma) := \{\sigma(x) \mid x \in Dom(\sigma)\}.$$

Variable range of a substitution σ:

$$VRan(\sigma) := Var(Ran(\sigma)).$$

Substitutions

Substitution: A function σ from variables to terms, whose domain

$$Dom(\sigma) := \{x \mid \sigma(x) \neq x\}$$

is finite.

Range of a substitution σ:

$$Ran(\sigma) := \{\sigma(x) \mid x \in Dom(\sigma)\}.$$

Variable range of a substitution σ:

$$VRan(\sigma) := Var(Ran(\sigma)).$$

Notation: lower case Greek letters $\sigma, \vartheta, \varphi, \psi, \ldots$.

Identity substitution: ε.

Substitutions

Notation: If $Dom(\sigma) = \{x_1, \ldots, x_n\}$, then σ can be written as the set

$$\{x_1 \mapsto \sigma(x_1), \ldots, x_n \mapsto \sigma(x_n)\}.$$

Substitutions

Substitutions can be extended to terms:

$\sigma(f(t_1, \ldots, t_n)) = f(\sigma(t_1), \ldots, \sigma(t_n))$.

$\sigma(t)$: an instance of t.

Substitutions

Substitutions can be extended to terms:

$\sigma(f(t_1, \ldots, t_n)) = f(\sigma(t_1), \ldots, \sigma(t_n))$.

$\sigma(t)$: an instance of t.

Example:

$$\sigma = \{x \mapsto i(y), y \mapsto e\}.$$
$$t = f(y, f(x, y))$$
$$\sigma(t) = f(e, f(i(y), e))$$

Substitutions

Substitutions can be extended to terms:

$\sigma(f(t_1, \ldots, t_n)) = f(\sigma(t_1), \ldots, \sigma(t_n))$.

$\sigma(t)$: an instance of t.

Example:

$$\sigma = \{x \mapsto i(y), y \mapsto e\}.$$
$$t = f(y, f(x, y))$$
$$\sigma(t) = f(e, f(i(y), e))$$

Sub : The set of substitutions.

Substitution Composition

Composition of ϑ and σ:

$$(\sigma\vartheta)(x) := \sigma(\vartheta(x)).$$

Composition is associative but not commutative.

Substitution Composition

Algorithm for obtaining a set representation of a composition of two substitutions in a set form.

- Given:

  $$\theta = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$$
  $$\sigma = \{y_1 \mapsto s_1, \ldots, y_m \mapsto s_m\},$$

  the set representation of their composition $\sigma\theta$ is obtained from the set

  $$\{x_1 \mapsto \sigma(t_1), \ldots, x_n \mapsto \sigma(t_n), y_1 \mapsto s_1, \ldots, y_m \mapsto s_m\}$$

  by deleting

  - all $y_i \mapsto s_i$'s with $y_i \in \{x_1, \ldots, x_n\}$,
  - all $x_i \mapsto \sigma(t_i)$'s with $x_i = \sigma(t_i)$.

Substitution Composition

Example (Composition)

$$\theta = \{x \mapsto f(y), y \mapsto z\}.$$
$$\sigma = \{x \mapsto a, y \mapsto b, z \mapsto y\}.$$
$$\sigma\theta = \{x \mapsto f(b), z \mapsto y\}.$$

Substitution Composition

Example (Composition)

$$\theta = \{x \mapsto f(y), y \mapsto z\}.$$
$$\sigma = \{x \mapsto a, y \mapsto b, z \mapsto y\}.$$
$$\sigma\theta = \{x \mapsto f(b), z \mapsto y\}.$$

Let $\sigma = \{x \mapsto y, y \mapsto z, z \mapsto x\}$ and $\vartheta = \{y \mapsto x, z \mapsto y, x \mapsto z\}$

$\sigma\sigma =$

Substitution Composition

Example (Composition)

$$\theta = \{x \mapsto f(y), y \mapsto z\}.$$
$$\sigma = \{x \mapsto a, y \mapsto b, z \mapsto y\}.$$
$$\sigma\theta = \{x \mapsto f(b), z \mapsto y\}.$$

Let $\sigma = \{x \mapsto y, y \mapsto z, z \mapsto x\}$ and $\vartheta = \{y \mapsto x, z \mapsto y, x \mapsto z\}$

$\sigma\sigma = \{x \mapsto z, y \mapsto x, z \mapsto y\}.$

Substitution Composition

Example (Composition)

$$\theta = \{x \mapsto f(y), y \mapsto z\}.$$
$$\sigma = \{x \mapsto a, y \mapsto b, z \mapsto y\}.$$
$$\sigma\theta = \{x \mapsto f(b), z \mapsto y\}.$$

Let $\sigma = \{x \mapsto y, y \mapsto z, z \mapsto x\}$ and $\vartheta = \{y \mapsto x, z \mapsto y, x \mapsto z\}$

$\sigma\sigma = \{x \mapsto z, y \mapsto x, z \mapsto y\}.$

$\vartheta\sigma =$.

Substitution Composition

Example (Composition)

$$\theta = \{x \mapsto f(y), y \mapsto z\}.$$
$$\sigma = \{x \mapsto a, y \mapsto b, z \mapsto y\}.$$
$$\sigma\theta = \{x \mapsto f(b), z \mapsto y\}.$$

Let $\sigma = \{x \mapsto y, y \mapsto z, z \mapsto x\}$ and $\vartheta = \{y \mapsto x, z \mapsto y, x \mapsto z\}$

$\sigma\sigma = \{x \mapsto z, y \mapsto x, z \mapsto y\}.$

$\vartheta\sigma = \varepsilon.$

Semantics: Structure

Structure $S = (\mathrm{D}, \mathrm{I})$.

- D: nonempty domain.
- I: interpretation function.
- Structure fixes interpretation of function and predicate symbols.
- Meaning of variables is determined by a variable assignment.

Semantics: Interpretation Function

The interpretation function assigns

- to each $\mathrm{f} \in \mathcal{F}^{\mathfrak{n}}$ an $\mathfrak{n}$-ary function $\mathrm{f}_{\mathrm{I}} : \mathrm{D}^{\mathfrak{n}} \to \mathrm{D}$, (in particular, $\mathrm{c}_{\mathrm{I}} \in \mathrm{D}$ for each constant c)
- to each $\mathrm{p} \in \mathcal{P}^{\mathfrak{n}}$ (different from $\doteq$), an $\mathfrak{n}$-ary relation p_{I} on D.

Variable Assignment

A structure $S = (\mathrm{D}, \mathrm{I})$ is given.

Variable assignment σ_S maps each $x \in \mathcal{V}$ into an element of D: $\sigma_S(x) \in \mathrm{D}$.

Semantic counterpart of substitutions.

Define:

$$\sigma_S[x \to d](y) := \begin{cases} \sigma_S(y), & \text{if } x \neq y \\ d, & \text{otherwise.} \end{cases}$$

Interpretation of Terms

A structure $S = (D, I)$ and a variable assignment σ_S are given.

Value of a term t under S and σ_S, $Val_{S,\sigma_S}(t)$:

- $Val_{S,\sigma_S}(x) = \sigma_S(x)$.
- $Val_{S,\sigma_S}(f(t_1, \ldots, t_n)) = f_I(Val_{S,\sigma_S}(t_1), \ldots, Val_{S,\sigma_S}(t_n))$.

Interpretation of Formulas

A structure $S = (\mathrm{D}, \mathrm{I})$ and a variable assignment σ_S are given.

The truth value of a formula under S and σ_S is either true or false.

For atomic formulas:

Interpretation of Formulas

A structure $S = (D, I)$ and a variable assignment σ_S are given.

The truth value of a formula under S and σ_S is either $true$ or $false$.

For atomic formulas:

- $Val_{S,\sigma_S}(s \doteq t) = true$ iff $Val_{S,\sigma_S}(s) = Val_{S,\sigma_S}(t)$.

Interpretation of Formulas

A structure $S = (D, I)$ and a variable assignment σ_S are given.

The truth value of a formula under S and σ_S is either $true$ or $false$.

For atomic formulas:

- $Val_{S,\sigma_S}(s \doteq t) = true$ iff $Val_{S,\sigma_S}(s) = Val_{S,\sigma_S}(t)$.
- $Val_{S,\sigma_S}(p(t_1, \ldots, t_n)) = true$ iff
  $(Val_{S,\sigma_S}(t_1), \ldots, Val_{S,\sigma_S}(t_n)) \in p_I$.

Interpretation of Formulas

For compound formulas:

Interpretation of Formulas

For compound formulas:

- $Val_{S,\sigma_S}(\neg A) = \mathtt{true}$ iff $Val_{S,\sigma_S}(A) = \mathtt{false}$.

Interpretation of Formulas

For compound formulas:

- $Val_{S,\sigma_S}(\neg A) = \mathrm{true}$ iff $Val_{S,\sigma_S}(A) = \mathrm{false}$.
- $Val_{S,\sigma_S}(A \vee B) = \mathrm{true}$ iff
  $Val_{S,\sigma_S}(A) = \mathrm{true}$ or $Val_{S,\sigma_S}(B) = \mathrm{true}$.

Interpretation of Formulas

For compound formulas:

- $Val_{S,\sigma_S}(\neg A) = true$ iff $Val_{S,\sigma_S}(A) = false$.
- $Val_{S,\sigma_S}(A \vee B) = true$ iff
  $Val_{S,\sigma_S}(A) = true$ or $Val_{S,\sigma_S}(B) = true$.
- $Val_{S,\sigma_S}(A \wedge B) = true$ iff
  $Val_{S,\sigma_S}(A) = true$ and $Val_{S,\sigma_S}(B) = true$.

Interpretation of Formulas

For compound formulas:

- $Val_{S,\sigma_S}(\neg A) = true$ iff $Val_{S,\sigma_S}(A) = false$.
- $Val_{S,\sigma_S}(A \vee B) = true$ iff
  $Val_{S,\sigma_S}(A) = true$ or $Val_{S,\sigma_S}(B) = true$.
- $Val_{S,\sigma_S}(A \wedge B) = true$ iff
  $Val_{S,\sigma_S}(A) = true$ and $Val_{S,\sigma_S}(B) = true$.
- $Val_{S,\sigma_S}(A \Rightarrow B) = true$ iff
  $Val_{S,\sigma_S}(A) = false$ or $Val_{S,\sigma_S}(B) = true$.

Interpretation of Formulas

For compound formulas:

- $Val_{S,\sigma_S}(\neg A) = true$ iff $Val_{S,\sigma_S}(A) = false$.
- $Val_{S,\sigma_S}(A \vee B) = true$ iff
  $Val_{S,\sigma_S}(A) = true$ or $Val_{S,\sigma_S}(B) = true$.
- $Val_{S,\sigma_S}(A \wedge B) = true$ iff
  $Val_{S,\sigma_S}(A) = true$ and $Val_{S,\sigma_S}(B) = true$.
- $Val_{S,\sigma_S}(A \Rightarrow B) = true$ iff
  $Val_{S,\sigma_S}(A) = false$ or $Val_{S,\sigma_S}(B) = true$.
- $Val_{S,\sigma_S}(A \Leftrightarrow B) = true$ iff $Val_{S,\sigma_S}(A) = Val_{S,\sigma_S}(B)$.

Interpretation of Formulas

For quantified formulas:

- $\mathcal{Val}_{S,\sigma_S}(\exists x.A) = \mathit{true}$ iff
  $\mathcal{Val}_{S,\sigma_S[x\to d]}(A) = \mathit{true}$ for some $d \in D$.
- $\mathcal{Val}_{S,\sigma_S}(\forall x.A) = \mathit{true}$ iff
  $\mathcal{Val}_{S,\sigma_S[x\to d]}(A) = \mathit{true}$ for all $d \in D$.

Interpretation of Formulas

The value of a formula A under S:

- $\mathit{Val}_S(A) = \mathit{true}$ iff $\mathit{Val}_{S,\sigma_S}(A) = \mathit{true}$ for all σ_S.

The value of a closed formula is independent of variable assignment.

Interpretation of Formulas

The value of a formula A under S:

- $Val_S(A) = true$ iff $Val_{S,\sigma_S}(A) = true$ for all σ_S.

The value of a closed formula is independent of variable assignment.

S is called a model of A iff $Val_S(A) = true$.

Written $\models_S A$.

Interpretation of Formulas

The value of a formula A under S:

- $Val_S(A) = true$ iff $Val_{S,\sigma_S}(A) = true$ for all σ_S.

The value of a closed formula is independent of variable assignment.

S is called a model of A iff $Val_S(A) = true$.

Written $\models_S A$.

A is a logical consequence of B iff every model of B is a model of A.

Written $B \models A$.

Example

Formula: $\forall x.(p(x) \Rightarrow q(f(x), a))$

Example

Formula: $\forall x.(p(x) \Rightarrow q(f(x), a))$

Define $S = (D, I)$ as

- $D = \{1, 2\}$,
- $a_I = 1$,
- $f_I(1) = 2$, $f_I(2) = 1$,
- $p_I = \{2\}$,
- $q_I = \{(1, 1), (1, 2), (2, 2)\}$.

Example

Formula: $\forall x.(p(x) \Rightarrow q(f(x), a))$

Define $S = (D, I)$ as

- $D = \{1, 2\}$,
- $a_I = 1$,
- $f_I(1) = 2$, $f_I(2) = 1$,
- $p_I = \{2\}$,
- $q_I = \{(1, 1), (1, 2), (2, 2)\}$.

$Val_S(\forall x.(p(x) \Rightarrow q(f(x), a))) = \text{true}.$

Example

Formula: $\forall x.(p(x) \Rightarrow q(f(x), a))$

Define $S = (D, I)$ as

- $D = \{1, 2\}$,
- $a_I = 1$,
- $f_I(1) = 2$, $f_I(2) = 1$,
- $p_I = \{2\}$,
- $q_I = \{(1, 1), (1, 2), (2, 2)\}$.

$Val_S(\forall x.(p(x) \Rightarrow q(f(x), a))) = \mathrm{true}$.

Hence, $\models_S A$.

Validity, Unsatisfiability

A formula A is valid, if $\models_S A$ for all S.

Written $\models A$.

Validity, Unsatisfiability

A formula A is valid, if $\models_S A$ for all S.

Written $\models A$.

A formula A is unsatisfiable, if $\models_S A$ for no S.

Validity, Unsatisfiability

A formula A is valid, if $\models_S A$ for all S.

Written $\models A$.

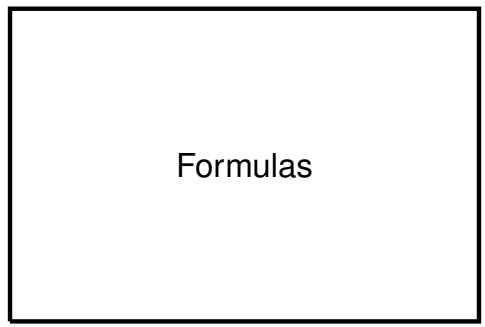
A formula A is unsatisfiable, if $\models_S A$ for no S.

Validity, Unsatisfiability

A formula A is valid, if $\models_S A$ for all S.

Written $\models A$.

A formula A is unsatisfiable, if $\models_S A$ for no S.

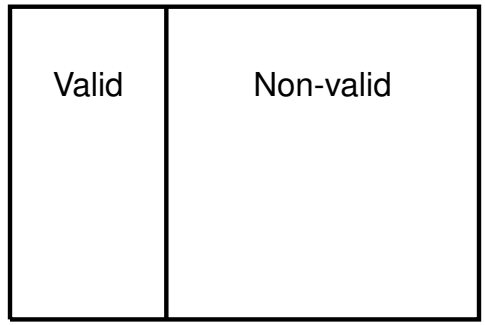


Validity, Unsatisfiability

A formula A is valid, if $\models_S A$ for all S.

Written $\models A$.

A formula A is unsatisfiable, if $\models_S A$ for no S.

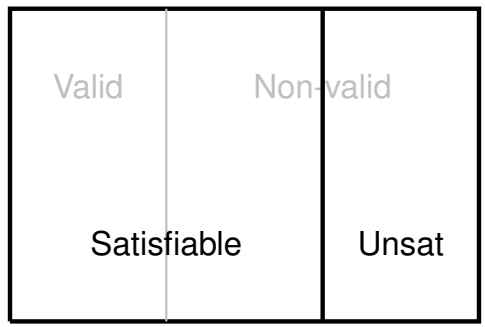


Validity, Unsatisfiability

A formula A is valid, if $\models_S A$ for all S.

Written $\models A$.

A formula A is unsatisfiable, if $\models_S A$ for no S.

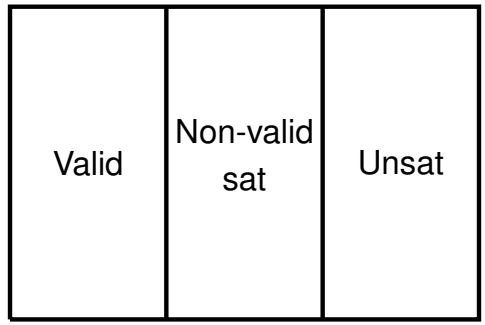


Validity, Unsatisfiability

A formula A is valid, if $\models_S A$ for all S.

Written $\models A$.

A formula A is unsatisfiable, if $\models_S A$ for no S.



Validity, Unsatisfiability

Proposition

Let A and B be formulas and K be a set of formulas. Then

1. A is valid iff $\neg A$ is unsatisfiable.
2. $B \models A$ iff $B \wedge \neg A$ is unsatisfiable.
3. $K \models A$ iff $K \cup \{\neg A\}$ is unsatisfiable.

Inference System

Resolution Calculus

The Resolution Calculus

Operates on the clausal fragment of first-order logic

Clause: A formula of the form $\forall x_1. \cdots . \forall x_n . (L_1 \vee \cdots \vee L_k)$, where

- each L_i is a literal,
- $L_1 \vee \cdots \vee L_k$ contains no variables other than $x_1, \ldots, x_n$.

Every first-order formula can be reduced to a set of clauses.

The reduction preserves unsatisfiability.

Clauses are often written without quantifier prefix: $L_1 \vee \cdots \vee L_k$.

Clausification

Every first-order formula can be reduced to a set of clauses:

Step 1: Transformation into a prenex normal form:

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

where each Q_i is either $\forall$ or $\exists$ and the formula M contains no quantifiers.

Step 2: Skolemization.

Step 3: CNF transformation.

Step 4: Stripping off the quantifiers and transforming the formula in CNF into set of clauses.

Transformation into a Prenex Normal Form

Traditional way.

Rename bound variables, apply the $\rightsquigarrow_{\mathrm{P}}$ rules in any context.

($\overline{\forall} = \exists$, $\overline{\exists} = \forall$, B does not contain x freely.)

$$A_1 \Leftrightarrow A_2 \rightsquigarrow_{\mathrm{P}} (A_1 \Rightarrow A_2) \wedge (A_2 \Rightarrow A_1).$$

$$\neg Qx.A \rightsquigarrow_{\mathrm{P}} \overline{Q}x.\neg A.$$

$$((Qx.A) \star B) \rightsquigarrow_{\mathrm{P}} (Qx.A \star B), \quad \star \in \{\wedge, \vee\}$$

$$((Qx.A) \Rightarrow B) \rightsquigarrow_{\mathrm{P}} (\overline{Q}x.A \Rightarrow B).$$

$$(B \star (Qx.A)) \rightsquigarrow_{\mathrm{P}} Qx.(B \star A), \quad \star \in \{\wedge, \vee, \Rightarrow\}$$

Transformation into a Prenex Normal Form

Traditional way.

Rename bound variables, apply the $\rightsquigarrow_{\mathrm{P}}$ rules in any context.

($\overline{\forall} = \exists$, $\overline{\exists} = \forall$, B does not contain x freely.)

$$A_1 \Leftrightarrow A_2 \rightsquigarrow_{\mathrm{P}} (A_1 \Rightarrow A_2) \wedge (A_2 \Rightarrow A_1).$$
$$\neg Qx.A \rightsquigarrow_{\mathrm{P}} \overline{Q}x.\neg A.$$
$$((Qx.A) \star B) \rightsquigarrow_{\mathrm{P}} (Qx.A \star B), \quad \star \in \{\wedge, \vee\}$$
$$((Qx.A) \Rightarrow B) \rightsquigarrow_{\mathrm{P}} (\overline{Q}x.A \Rightarrow B).$$
$$(B \star (Qx.A)) \rightsquigarrow_{\mathrm{P}} Qx.(B \star A), \quad \star \in \{\wedge, \vee, \Rightarrow\}$$

If $F \rightsquigarrow_{\mathrm{P}}^{*} G$, then G is in prenex normal form.

If F and G are closed, then they are equivalent.

Skolemization

Replace existentially quantified variables by Skolem functions:

- The formula $Q_1 x_1. \cdots Q_n x_n. M$ is in prenex normal form
- Skolemization rule:

$$\forall x_1. \cdots \forall x_n. \exists y. Q_1 z_1. \cdots Q_m z_m.\, M[y] \rightsquigarrow_S \forall x_1. \cdots \forall x_n. Q_1 z_1. \cdots Q_m z_m.\, M[f(x_1, \ldots, x_n)]$$

  where f is a new function symbol of arity n with $n \geqslant 0$.

- Intuition: replace $\exists y$ by a concrete choice function computing y from all the arguments it depends on.

Skolemization

Replace existentially quantified variables by Skolem functions:

- The formula $Q_1 x_1. \cdots Q_n x_n. M$ is in prenex normal form
- Skolemization rule:

$$\forall x_1. \cdots \forall x_n. \exists y. Q_1 z_1. \cdots Q_m z_m.\, M[y] \rightsquigarrow_S \forall x_1. \cdots \forall x_n. Q_1 z_1. \cdots Q_m z_m.\, M[f(x_1, \ldots, x_n)]$$

  where f is a new function symbol of arity n with $n \geqslant 0$.

- Intuition: replace $\exists y$ by a concrete choice function computing y from all the arguments it depends on.

If G is in PNF and $G \rightsquigarrow_S^* H$, then H is in PNF without $\exists$.

$H \models G$ but not the other way around.

G is (un)satisfiable iff H is (un)satisfiable.

Skolemization does not preserve equivalence

$G \rightsquigarrow_S^* H$, $G \not\models H$:

- $G = \exists x.p(x)$, $H = p(a)$.
- $S = (\{1, 2\}, I)$.
- $a_I = 1$.
- $p_I = \{2\}$.
- Then $Val_S(G) = true$ but $Val_S(H) = false$.

Transformation into Clausal Normal Form

$$
\begin{aligned}
F &\rightsquigarrow_P^* Q_1 y_1 \cdots Q_n y_n . A \\
&\rightsquigarrow_S^* \forall x_1 \cdots \forall x_n . B \\
&\rightsquigarrow_{CNF}^* \forall x_1 . \cdots . \forall x_n . \bigwedge_{i=1}^{k} C_i
\end{aligned}
$$

where C_i are clauses.

$\rightsquigarrow_{CNF}^*$ preserves (un)satisfiability.

$\{C_1, \ldots, C_k\}$: clausal normal form of F.

Clausification Example

$$\forall x.\exists y.(\exists z.(p(x,z) \vee p(y,z)) \Rightarrow \exists u.q(x,y,u))$$

Clausification Example

$$\forall x.\exists y.(\exists z.(p(x,z) \vee p(y,z)) \Rightarrow \exists u.q(x,y,u))$$

$$\rightsquigarrow_P \quad \forall x.\exists y.\forall z.(p(x,z) \vee p(y,z) \Rightarrow \exists u.q(x,y,u))$$

Clausification Example

$$
\begin{aligned}
& \forall x.\exists y.(\exists z.(p(x,z) \vee p(y,z)) \Rightarrow \exists u.q(x,y,u)) \\
\rightsquigarrow_P \quad & \forall x.\exists y.\forall z.(p(x,z) \vee p(y,z) \Rightarrow \exists u.q(x,y,u)) \\
\rightsquigarrow_P \quad & \forall x.\exists y.\forall z.\exists u.(p(x,z) \vee p(y,z) \Rightarrow q(x,y,u))
\end{aligned}
$$

Clausification Example

$$
\begin{aligned}
& \forall x.\exists y.(\exists z.(p(x,z) \vee p(y,z)) \Rightarrow \exists u.q(x,y,u)) \\
\rightsquigarrow_P \quad & \forall x.\exists y.\forall z.(p(x,z) \vee p(y,z) \Rightarrow \exists u.q(x,y,u)) \\
\rightsquigarrow_P \quad & \forall x.\exists y.\forall z.\exists u.(p(x,z) \vee p(y,z) \Rightarrow q(x,y,u)) \\
\rightsquigarrow_S \quad & \forall x.\forall z.\exists u.(p(x,z) \vee p(f_1(x),z) \Rightarrow q(x,f_1(x),u))
\end{aligned}
$$

Clausification Example

$$
\begin{aligned}
& \forall x.\exists y.(\exists z.(p(x,z) \vee p(y,z)) \Rightarrow \exists u.q(x,y,u)) \\
\rightsquigarrow_P \quad & \forall x.\exists y.\forall z.(p(x,z) \vee p(y,z) \Rightarrow \exists u.q(x,y,u)) \\
\rightsquigarrow_P \quad & \forall x.\exists y.\forall z.\exists u.(p(x,z) \vee p(y,z) \Rightarrow q(x,y,u)) \\
\rightsquigarrow_S \quad & \forall x.\forall z.\exists u.(p(x,z) \vee p(f_1(x),z) \Rightarrow q(x,f_1(x),u)) \\
\rightsquigarrow_S \quad & \forall x.\forall z.(p(x,z) \vee p(f_1(x),z) \Rightarrow q(x,f_1(x),f_2(x,z)))
\end{aligned}
$$

Clausification Example

$$
\begin{aligned}
& \forall x.\exists y.(\exists z.(p(x,z) \vee p(y,z)) \Rightarrow \exists u.q(x,y,u)) \\
\rightsquigarrow_{P} \quad & \forall x.\exists y.\forall z.(p(x,z) \vee p(y,z) \Rightarrow \exists u.q(x,y,u)) \\
\rightsquigarrow_{P} \quad & \forall x.\exists y.\forall z.\exists u.(p(x,z) \vee p(y,z) \Rightarrow q(x,y,u)) \\
\rightsquigarrow_{S} \quad & \forall x.\forall z.\exists u.(p(x,z) \vee p(f_1(x),z) \Rightarrow q(x,f_1(x),u)) \\
\rightsquigarrow_{S} \quad & \forall x.\forall z.(p(x,z) \vee p(f_1(x),z) \Rightarrow q(x,f_1(x),f_2(x,z))) \\
\rightsquigarrow_{CNF} \quad & \forall x.\forall z.((\neg p(x,z) \vee q(x,f_1(x),f_2(x,z))) \wedge \\
& \qquad (\neg p(f_1(x),z) \vee q(x,f_1(x),f_2(x,z))))
\end{aligned}
$$

Clausification Example

$$
\begin{aligned}
& \forall x.\exists y.(\exists z.(p(x,z) \vee p(y,z)) \Rightarrow \exists u.q(x,y,u)) \\
\rightsquigarrow_{P} \quad & \forall x.\exists y.\forall z.(p(x,z) \vee p(y,z) \Rightarrow \exists u.q(x,y,u)) \\
\rightsquigarrow_{P} \quad & \forall x.\exists y.\forall z.\exists u.(p(x,z) \vee p(y,z) \Rightarrow q(x,y,u)) \\
\rightsquigarrow_{S} \quad & \forall x.\forall z.\exists u.(p(x,z) \vee p(f_1(x),z) \Rightarrow q(x,f_1(x),u)) \\
\rightsquigarrow_{S} \quad & \forall x.\forall z.(p(x,z) \vee p(f_1(x),z) \Rightarrow q(x,f_1(x),f_2(x,z))) \\
\rightsquigarrow_{CNF} \quad & \forall x.\forall z.((\neg p(x,z) \vee q(x,f_1(x),f_2(x,z))) \wedge \\
& \qquad (\neg p(f_1(x),z) \vee q(x,f_1(x),f_2(x,z))))
\end{aligned}
$$

$$\{\neg p(x,z) \vee q(x,f_1(x),f_2(x,z)), \neg p(f_1(x),z) \vee q(x,f_1(x),f_2(x,z))\}$$

What Do We Do?

Given: A set of assumptions $A_1, \ldots, A_n$ and a conjecture B.

Establish validity of $A_1 \wedge \cdots \wedge A_n \Rightarrow B$.

What Do We Do?

Given: A set of assumptions $A_1, \ldots, A_n$ and a conjecture B.

Establish validity of $A_1 \wedge \cdots \wedge A_n \Rightarrow B$.

For this, we negate the conjecture and try to establish unsatisfiability of $A_1 \wedge \cdots \wedge A_n \wedge \neg B$.

What Do We Do?

Given: A set of assumptions $A_1, \ldots, A_n$ and a conjecture B.

Establish validity of $A_1 \wedge \cdots \wedge A_n \Rightarrow B$.

For this, we negate the conjecture and try to establish unsatisfiability of $A_1 \wedge \cdots \wedge A_n \wedge \neg B$.

Inference system (for the fragment without equality): resolution calculus.

Clausification of $A_1 \wedge \cdots \wedge A_n \wedge \neg B$ preserves unsatisfiability.

Resolution works on clauses and tries to derive a contradiction.

Herbrand Interpretation

Structure $\mathcal{H} = (\mathrm{D}, \mathrm{I})$, where

- D is the set of ground terms,
- for each n-ary function symbol f, $\mathrm{f_i}$ maps $(\mathrm{t_1}, \ldots, \mathrm{t_n}) \in \mathrm{D^n}$ into $\mathrm{f}(\mathrm{t_1}, \ldots, \mathrm{t_n}) \in \mathrm{D}$.

Herbrand interpretation $\mathcal{H}$ can be identified with the set of ground atoms which are true in $\mathcal{H}$.

Herbrand's Theorem

Substitutions extend to clauses.

$\sigma(C)$: an instance of a clause C.

$\mathrm{ground}(K)$, where K is a set of clauses: The set of all ground instances of clauses in K.

Theorem

A set of clauses K is satisfiable iff it has a Herbrand model iff $\mathrm{ground}(K)$ has a Herbrand model.

Inference Systems

Inference systems are sets of inferences:

Inference: a tuple $(F_1, \ldots, F_n, F_{n+1})$, $n \geqslant 0$, written as

$$\frac{F_1, \ldots, F_n}{F_{n+1}}$$

$F_1, \ldots, F_n$: premises.

F_{n+1}: conclusion.

Proofs

A proof in an inference system IS of a formula A from a set of assumptions K: A sequence of formulas $F_1, \ldots, F_m$, where

- $F_m = A$,
- for all $1 \leqslant i \leqslant m$, $F_i \in K$ or there exists an inference in IS

$$\frac{F_{i_1}, \ldots, F_{i_k}}{F_i}$$

where $1 \leqslant i_j \leqslant i$ for each $1 \leqslant j \leqslant k$.

Soundness and Completeness

$K \vdash_{IS} A$: There exists a proof of A from K in IS, A is **provable** from K in IS.

Soundness of IS: For each inference $\frac{F_1,\ldots,F_n}{F} \in IS$, $F_1, \ldots, F_n \models F$.

Completeness of IS: If $K \models F$, then $K \vdash_{IS} F$.

Refutational Completeness of IS: If $K \models \square$, then $K \vdash_{IS} \square$, where $\square$ is the empty clause.

Resolution Calculus for Ground Clauses

A: atom, C, D: clauses, L: literal.

- Ground Binary resolution:

$$\frac{A \vee C \qquad \neg A \vee D}{C \vee D}$$

- Ground Factoring:

$$\frac{L \vee L \vee C}{L \vee C}$$

Resolution Calculus for Ground Clauses

Sample refutation:

1. $p(a) \vee q(b)$
2. $p(a) \vee \neg q(b)$
3. $\neg p(a) \vee q(b)$
4. $\neg p(a) \vee \neg q(b)$
5. $p(a) \vee p(a)$ (BR 1,2)
6. $p(a)$ (Factor, 5)
7. $\neg p(a) \vee \neg p(a)$ (BR 3,4)
8. $\neg p(a)$ (Factor, 7)
9. $\square$ (BR 6, 8)

Resolution Calculus for Ground Clauses

Theorem

Resolution calculus for ground clauses is sound.

Theorem

Resolution calculus for ground clauses is refutationally complete: If $K \nvdash_{GRes} \square$, then K has a model.

Proof is based on a construction that builds a Herbrand model for K.

Resolution Calculus for General Case

How to lift the results from propositional to first-order case?

Property: Any model of a clause C is also a model for all instances $\sigma(C)$ of C.

Prove that some instances of clauses from K form an unsatisfiable set. Then K will be unsatisfiable.

Find appropriate instantiations.

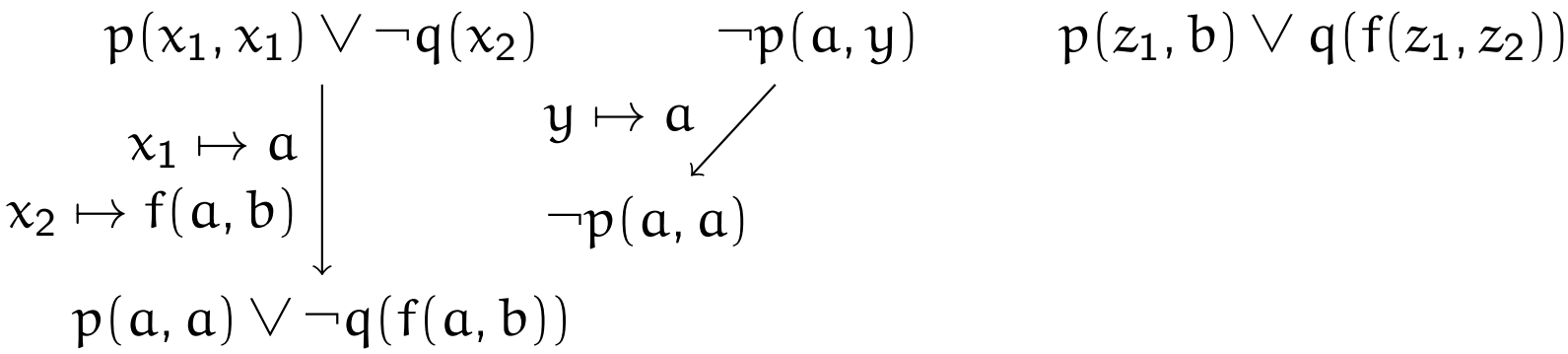
Resolution Calculus for General Case

Appropriate instantiations should create complementary literals.

$$p(x_1, x_1) \vee \neg q(x_2) \qquad \neg p(a, y) \qquad p(z_1, b) \vee q(f(z_1, z_2))$$

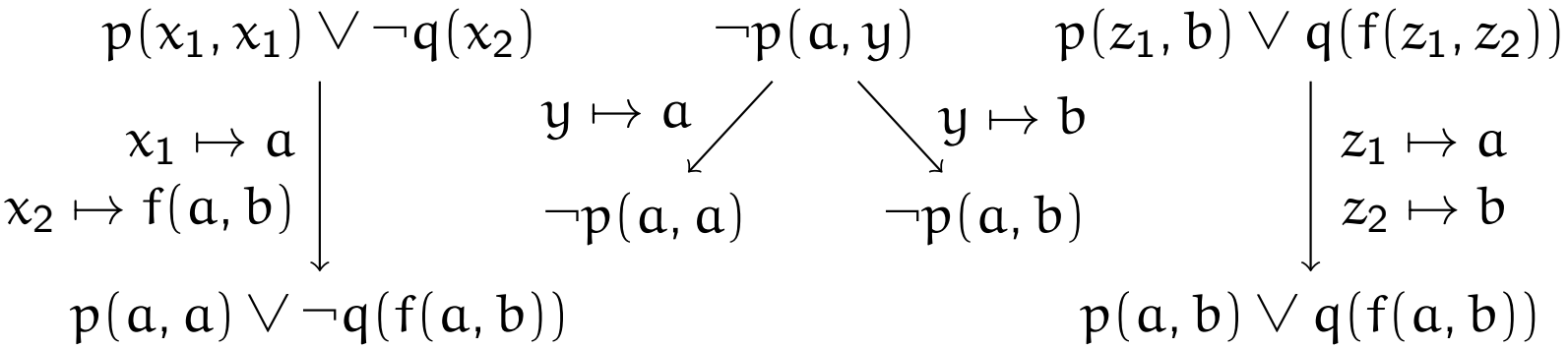
Resolution Calculus for General Case

Appropriate instantiations should create complementary literals.



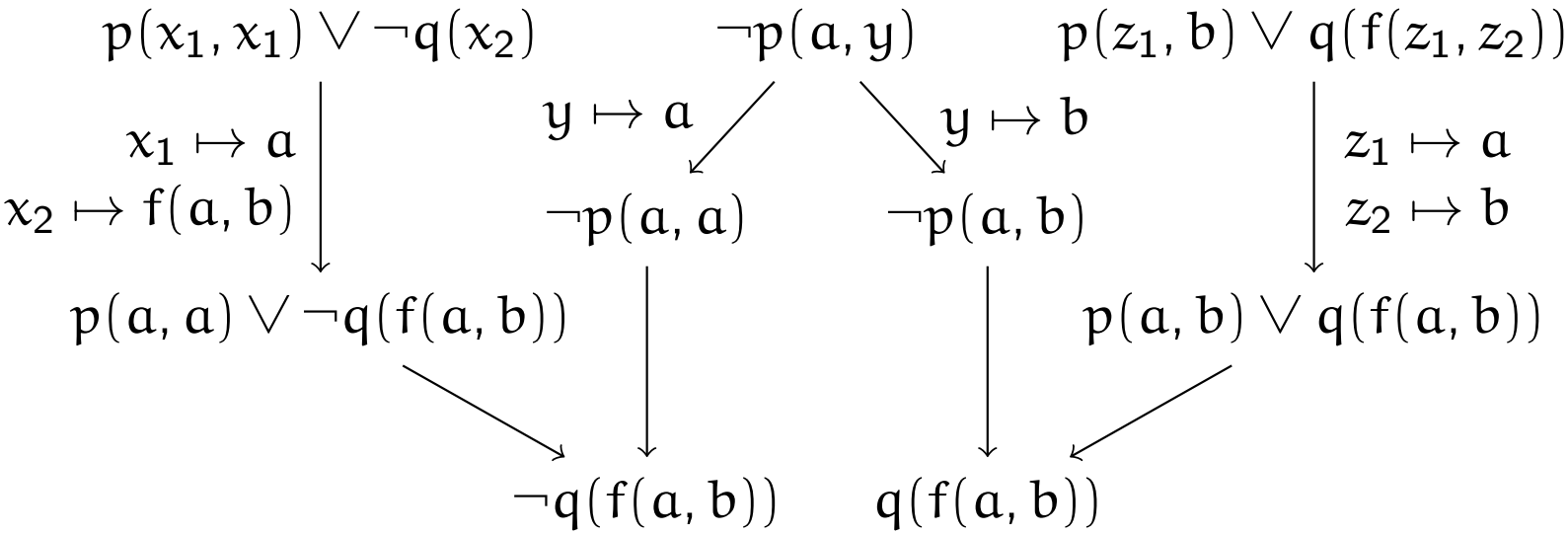
Resolution Calculus for General Case

Appropriate instantiations should create complementary literals.



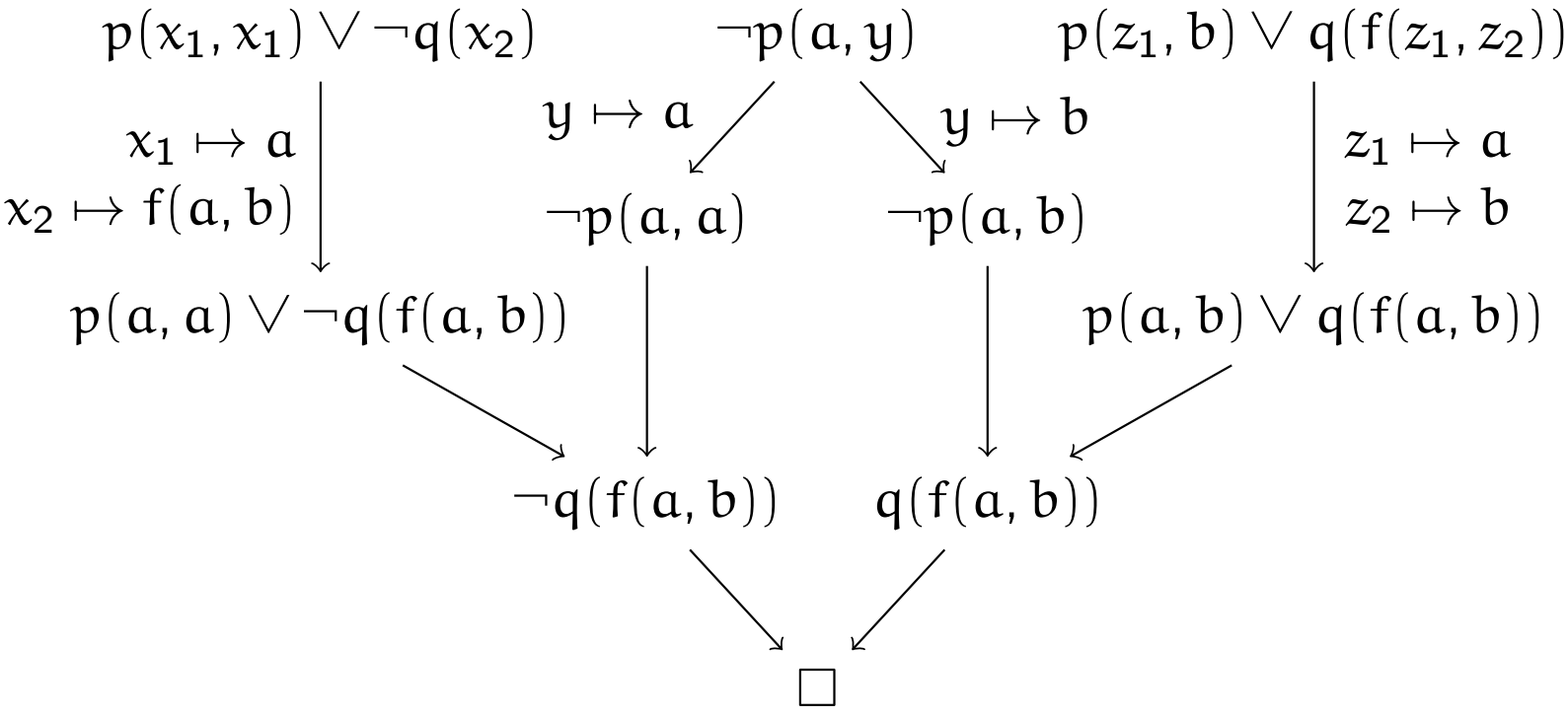
Resolution Calculus for General Case

Appropriate instantiations should create complementary literals.



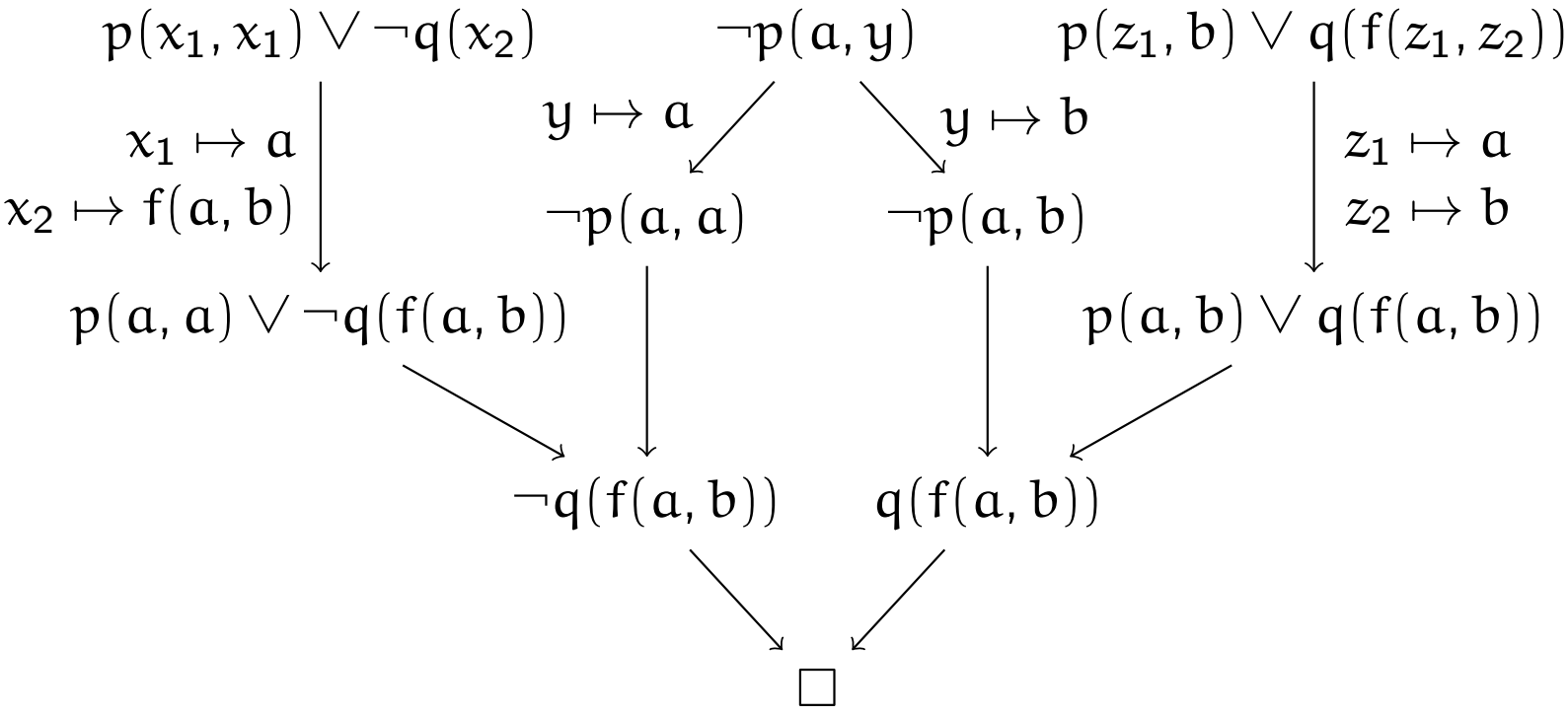
Resolution Calculus for General Case

Appropriate instantiations should create complementary literals.



Resolution Calculus for General Case

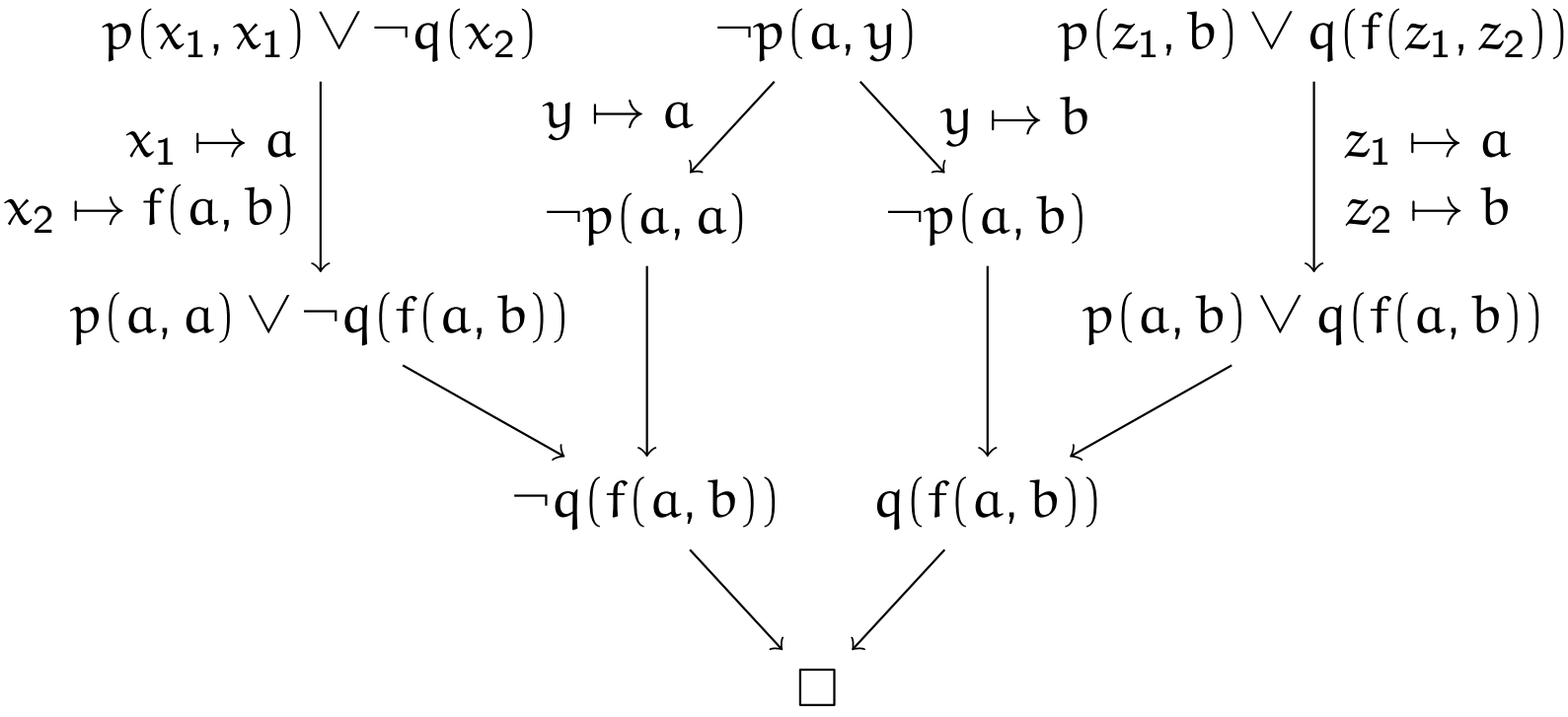
Appropriate instantiations should create complementary literals.



Do only necessary work.

Resolution Calculus for General Case

Appropriate instantiations should create complementary literals.



Do only necessary work.

Unification.

Unification

Syntactic unification:

Given: Two terms s and t.

Find: A substitution σ such that $\sigma(s) = \sigma(t)$.

- σ: a unifier of s and t.
- σ: a solution of the equation $s \doteq^{?} t$.

Example

$x \doteq^{?} f(y)$: infinitely many unifiers

$$\{x \mapsto f(y)\}, \{x \mapsto f(a), y \mapsto a\}, \ldots$$

Some solutions are better than the others: $\{x \mapsto f(y)\}$ is more general than $\{x \mapsto f(a), y \mapsto a\}$

Instantiation Quasi-Ordering

A substitution σ is more general than ϑ, written $\sigma \lesssim \vartheta$, if there exists η such that $\eta\sigma = \vartheta$.

ϑ is called an instance of σ.

The relation $\lesssim$ is reflexive and transitive binary relation, called instantiation quasi-ordering.

$\simeq$ is the equivalence relation corresponding to $\lesssim$, i.e., the relation $\lesssim \cap \gtrsim$.

Instantiation Quasi-Ordering

Example

Let $\sigma = \{x \mapsto y\}$, $\rho = \{x \mapsto a, y \mapsto a\}$, $\vartheta = \{y \mapsto x\}$.

- $\sigma \lesssim \rho$, because $\{y \mapsto a\}\sigma = \rho$.
- $\sigma \lesssim \vartheta$, because $\{y \mapsto x\}\sigma = \vartheta$.
- $\vartheta \lesssim \sigma$, because $\{x \mapsto y\}\vartheta = \sigma$.
- $\sigma \simeq \vartheta$.

Variable Renaming

A substitution $\sigma = \{x_1 \mapsto y_1, x_2 \mapsto y_2, \ldots, x_n \mapsto y_n\}$ is called variable renaming iff $\{x_1, \ldots, x_n\} = \{y_1, \ldots, y_n\}$.

(Permuting the domain variables.)

Example

- $\{x \mapsto y, y \mapsto z, z \mapsto x\}$ is a variable renaming.
- $\{x \mapsto a\}$, $\{x \mapsto y\}$, and $\{x \mapsto z, y \mapsto z, z \mapsto x\}$ are not.

Idempotent Substitutions

Definition

A substitution σ is idempotent iff $\sigma\sigma = \sigma$.

Example

Let $\sigma = \{x \mapsto f(z), y \mapsto z\}$, $\vartheta = \{x \mapsto f(y), y \mapsto z\}$.

- σ is idempotent.
- ϑ is not: $\vartheta\vartheta = \sigma \neq \vartheta$.

Idempotent Substitutions

Definition

A substitution σ is idempotent iff $\sigma\sigma = \sigma$.

Example

Let $\sigma = \{x \mapsto f(z), y \mapsto z\}$, $\vartheta = \{x \mapsto f(y), y \mapsto z\}$.

- σ is idempotent.
- ϑ is not: $\vartheta\vartheta = \sigma \neq \vartheta$.

Theorem

σ is idempotent iff $Dom(\sigma) \cap VRan(\sigma) = \emptyset$.

$\simeq$ and Variable Renaming

Lemma

$\sigma \simeq \vartheta$ iff there exists a variable renaming ρ such that $\rho\sigma = \vartheta$.

$\simeq$ and Variable Renaming

Lemma

$\sigma \simeq \vartheta$ iff there exists a variable renaming ρ such that $\rho\sigma = \vartheta$.

Example

- $\sigma = \{x \mapsto y\}$.
- $\vartheta = \{y \mapsto x\}$.
- $\sigma \simeq \vartheta$.
- $\{x \mapsto y, y \mapsto x\}\sigma = \vartheta$.

Unification Problem, Unifier, MGU

Unification problem:

A finite set of equations $\Gamma = \{s_1 \stackrel{\cdot}{=}^? t_1, \ldots, s_n \stackrel{\cdot}{=}^? t_n\}$.

Unification Problem, Unifier, MGU

Unification problem:
A finite set of equations $\Gamma = \{s_1 \stackrel{\cdot}{=}^? t_1, \ldots, s_n \stackrel{\cdot}{=}^? t_n\}$.

Unifier or solution of Γ:
A substitution σ such that $\sigma(s_i) = \sigma(t_i)$ for all $1 \leqslant i \leqslant n$.

Unification Problem, Unifier, MGU

Unification problem:
A finite set of equations $\Gamma = \{s_1 \stackrel{\cdot}{=}^? t_1, \ldots, s_n \stackrel{\cdot}{=}^? t_n\}$.

Unifier or solution of Γ:
A substitution σ such that $\sigma(s_i) = \sigma(t_i)$ for all $1 \leqslant i \leqslant n$.

$\mathcal{U}(\Gamma)$: The set of all unifiers of Γ. Γ is unifiable iff $\mathcal{U}(\Gamma) \neq \emptyset$.

Unification Problem, Unifier, MGU

Unification problem:
A finite set of equations $\Gamma = \{s_1 \stackrel{\cdot}{=}^? t_1, \ldots, s_n \stackrel{\cdot}{=}^? t_n\}$.

Unifier or solution of Γ:
A substitution σ such that $\sigma(s_i) = \sigma(t_i)$ for all $1 \leqslant i \leqslant n$.

$\mathcal{U}(\Gamma)$: The set of all unifiers of Γ. Γ is unifiable iff $\mathcal{U}(\Gamma) \neq \emptyset$.

σ is a most general unifier (mgu) of Γ iff it is a least element of $\mathcal{U}(\Gamma)$:

- $\sigma \in \mathcal{U}(\Gamma)$, and
- $\sigma \lesssim \vartheta$ for every $\vartheta \in \mathcal{U}(\Gamma)$.

Unifiers: Example

$\sigma := \{x \mapsto y\}$ is an mgu of $x \doteq^? y$.

For any other unifier ϑ of $x \doteq^? y$, $\sigma \lesssim \vartheta$ because

- $\vartheta(x) = \vartheta(y) = \vartheta\sigma(x)$.
- $\vartheta(y) = \vartheta\sigma(y)$.
- $\vartheta(z) = \vartheta\sigma(z)$ for any other variable z.

Unifiers: Example

$\sigma := \{x \mapsto y\}$ is an mgu of $x \doteq^? y$.

For any other unifier ϑ of $x \doteq^? y$, $\sigma \lesssim \vartheta$ because

- $\vartheta(x) = \vartheta(y) = \vartheta\sigma(x)$.
- $\vartheta(y) = \vartheta\sigma(y)$.
- $\vartheta(z) = \vartheta\sigma(z)$ for any other variable z.

$\sigma' := \{x \mapsto z, y \mapsto z\}$ is a unifier but not an mgu of $x \doteq^? y$.

- $\sigma' = \{y \mapsto z\}\sigma$.
- $\{z \mapsto y\}\sigma' = \{x \mapsto y, z \mapsto y\} \neq \sigma$.

Unifiers: Example

$\sigma := \{x \mapsto y\}$ is an mgu of $x \doteq^? y$.

For any other unifier ϑ of $x \doteq^? y$, $\sigma \lesssim \vartheta$ because

- $\vartheta(x) = \vartheta(y) = \vartheta\sigma(x)$.
- $\vartheta(y) = \vartheta\sigma(y)$.
- $\vartheta(z) = \vartheta\sigma(z)$ for any other variable z.

$\sigma' := \{x \mapsto z, y \mapsto z\}$ is a unifier but not an mgu of $x \doteq^? y$.

- $\sigma' = \{y \mapsto z\}\sigma$.
- $\{z \mapsto y\}\sigma' = \{x \mapsto y, z \mapsto y\} \neq \sigma$.

$\sigma'' = \{x \mapsto z_1, y \mapsto z_1, z_1 \mapsto y\}$ is an mgu of $x \doteq^? y$.

- $\sigma = \{y \mapsto z_1, z_1 \mapsto y\}\sigma''$.
- σ'' is not idempotent.

Unifiers: Example

Mgus of $x \doteq^{?} y$?

- $\{x \mapsto y\}$
- $\{y \mapsto x\}$
- $\{x \mapsto z_1,\ y \mapsto z_1,\ z_1 \mapsto y\}$

Unifiers: Example

Mgus of $x \doteq^{?} y$?

- $\{x \mapsto y\}$
- $\{y \mapsto x\}$
- $\{x \mapsto z_1,\ y \mapsto z_1,\ z_1 \mapsto y\}$
- $\{x \mapsto z_1,\ y \mapsto z_1,\ z_1 \mapsto z_2,\ z_2 \mapsto y\}$

Unifiers: Example

Mgus of $x \doteq^{?} y$?

- $\{x \mapsto y\}$
- $\{y \mapsto x\}$
- $\{x \mapsto z_1,\ y \mapsto z_1,\ z_1 \mapsto y\}$
- $\{x \mapsto z_1,\ y \mapsto z_1,\ z_1 \mapsto z_2,\ z_2 \mapsto y\}$
- $\{x \mapsto y,\ z_1 \mapsto z_2,\ z_2 \mapsto z_1\}$
- $\{x \mapsto y,\ z_1 \mapsto z_2,\ z_2 \mapsto z_3,\ z_3 \mapsto z_1\}$
- $\{x \mapsto y,\ z_1 \mapsto z_2,\ z_2 \mapsto z_3,\ z_3 \mapsto z_4,\ z_4 \mapsto z_1\}$

Unifiers: Example

Mgus of $x \doteq^{?} y$?

- $\{x \mapsto y\}$
- $\{y \mapsto x\}$
- $\{x \mapsto z_1,\ y \mapsto z_1,\ z_1 \mapsto y\}$
- $\{x \mapsto z_1,\ y \mapsto z_1,\ z_1 \mapsto z_2,\ z_2 \mapsto y\}$
- $\{x \mapsto y,\ z_1 \mapsto z_2,\ z_2 \mapsto z_1\}$
- $\{x \mapsto y,\ z_1 \mapsto z_2,\ z_2 \mapsto z_3,\ z_3 \mapsto z_1\}$
- $\{x \mapsto y,\ z_1 \mapsto z_2,\ z_2 \mapsto z_3,\ z_3 \mapsto z_4,\ z_4 \mapsto z_1\}$
- $\{y \mapsto x,\ z_1 \mapsto z_2,\ z_2 \mapsto z_1\}$
- $\{y \mapsto x,\ z_1 \mapsto z_2,\ z_2 \mapsto z_3,\ z_3 \mapsto z_1\}$
- ...

Unification

Question: How to compute an mgu of an unification problem?

Unification

Question: How to compute an mgu of an unification problem?

Rule-based unification algorithm.

Repeated transformation of a set of equations.

The Inference System $\mathfrak{U}$

A set of equations in solved form:

$$\{x_1 \approx t_1, \ldots, x_n \approx t_n\}$$

where each x_i occurs exactly once.

For each idempotent substitution there exists exactly one set of equations in solved form.

Notation:

- $[\sigma]$ for the solved form set for an idempotent substitution σ.
- σ_S for the idempotent substitution corresponding to a solved form set S.

The Inference System $\mathfrak{U}$

System: The symbol $\bot$ or a pair $\mathrm{P}; \mathrm{S}$ where

- P is a multiset of unification problems,
- S is a set of equations in solved form.

$\bot$ represents failure.

A unifier (or a solution) of a system $\mathrm{P}; \mathrm{S}$: A substitution that unifies each of the equations in P and S.

$\bot$ has no unifiers.

The Inference System $\mathfrak{U}$

Example

- System: $\{g(a) \doteq^? g(y), g(z) \doteq^? g(g(x))\}; \{x \approx g(y)\}$.
- Its unifier: $\{x \mapsto g(a), y \mapsto a, z \mapsto g(g(a))\}$.

The Inference System $\mathfrak{U}$

Six transformation rules on systems:

Trivial:

$$\{s \stackrel{\cdot}{=}^{?} s\} \uplus P'; S \Leftrightarrow P'; S.$$

Decomposition:

$$\{f(s_1, \ldots, s_n) \stackrel{\cdot}{=}^{?} f(t_1, \ldots, t_n)\} \uplus P'; S \Leftrightarrow$$
$$\{s_1 \stackrel{\cdot}{=}^{?} t_1, \ldots, s_n \stackrel{\cdot}{=}^{?} t_n\} \cup P'; S, \text{ where } n \geqslant 0.$$

Symbol Clash:

$$\{f(s_1, \ldots, s_n) \stackrel{\cdot}{=}^{?} g(t_1, \ldots, t_m)\} \uplus P'; S \Leftrightarrow \bot, \text{ if } f \neq g.$$

The Inference System $\mathfrak{U}$

Orient:

$$\{t \stackrel{\cdot}{=}^{?} x\} \uplus P'; S \Leftrightarrow \{x \stackrel{\cdot}{=}^{?} t\} \cup P'; S, \text{ if } t \notin \mathcal{V}.$$

Occurs Check:

$$\{x \stackrel{\cdot}{=}^{?} t\} \uplus P'; S \Leftrightarrow \bot \text{ if } x \in Var(t) \text{ but } x \neq t.$$

Variable Elimination:

$$\{x \stackrel{\cdot}{=}^{?} t\} \uplus P'; S \Leftrightarrow \{x \mapsto t\}(P'); \{x \mapsto t\}(S) \cup \{x \approx t\},$$

if $x \notin Var(t)$.

Unification with $\mathfrak{U}$

In order to unify s and t:

1. Create an initial system $\{s \doteq^? t\}; \emptyset$.
2. Apply successively rules from $\mathfrak{U}$.

The system $\mathfrak{U}$ is essentially the Herbrand's Unification Algorithm.

Example: Symbol Clash

Example (Failure)

Unify $p(f(a), g(x))$ and $p(y, y)$.

$$\{p(f(a), g(x)) \doteq^? p(y, y)\};\ \emptyset \Longrightarrow_{\mathsf{Dec}}$$
$$\{f(a) \doteq^? y, g(x) \doteq^? y\};\ \emptyset \Longrightarrow_{\mathsf{Or}}$$
$$\{y \doteq^? f(a), g(x) \doteq^? y\};\ \emptyset \Longrightarrow_{\mathsf{VarEl}}$$
$$\{g(x) \doteq^? f(a)\};\ \{y \approx f(a)\} \Longrightarrow_{\mathsf{SymCl}}$$
$$\bot$$

Example: Success

Example

Unify $p(a, x, h(g(z)))$ and $p(z, h(y), h(y))$.

$$\{p(a, x, h(g(z))) \doteq^? p(z, h(y), h(y))\};\ \emptyset \Longrightarrow_{\mathsf{Dec}}$$
$$\{a \doteq^? z, x \doteq^? h(y), h(g(z)) \doteq^? h(y)\};\ \emptyset \Longrightarrow_{\mathsf{Or}}$$
$$\{z \doteq^? a, x \doteq^? h(y), h(g(z)) \doteq^? h(y)\};\ \emptyset \Longrightarrow_{\mathsf{VarEl}}$$
$$\{x \doteq^? h(y), h(g(a)) \doteq^? h(y)\};\ \{z \approx a\} \Longrightarrow_{\mathsf{VarEl}}$$
$$\{h(g(a)) \doteq^? h(y)\};\ \{z \approx a, x \approx h(y)\} \Longrightarrow_{\mathsf{Dec}}$$
$$\{g(a) \doteq^? y\};\ \{z \approx a, x \approx h(y)\} \Longrightarrow_{\mathsf{Or}}$$
$$\{y \doteq^? g(a)\};\ \{z \approx a, x \approx h(y)\} \Longrightarrow_{\mathsf{VarEl}}$$
$$\emptyset;\ \{z \approx a, x \approx h(g(a)), y \approx g(a)\}.$$

Answer: $\{z \mapsto a, x \mapsto h(g(a)), y \mapsto g(a)\}$

Example: Occurrence Check

Example

Unify $p(x, x)$ and $p(y, f(y))$.

$$\{p(x,x) \doteq^? p(y,f(y))\};\ \emptyset \Longrightarrow_{\mathsf{Dec}}$$
$$\{x \doteq^? y, x \doteq^? f(y)\};\ \emptyset \Longrightarrow_{\mathsf{VarEl}}$$
$$\{y \doteq^? f(y)\};\ \{x \approx y\} \Longrightarrow_{\mathsf{OccCh}}$$
$$\bot$$

Properties of $\mathfrak{U}$: Termination

Theorem (Termination)

For any finite set of equations P, every sequence of transformations in $\mathfrak{U}$

$$\mathrm{P}; \emptyset \Leftrightarrow \mathrm{P}_1; \mathrm{S}_1 \Leftrightarrow \mathrm{P}_2; \mathrm{S}_2 \Leftrightarrow \cdots$$

terminates either with $\bot$ or with $\emptyset; S$, with S in solved form.

Corollary

If $\mathrm{P}; \emptyset \Leftrightarrow^{+} \emptyset; S$ then σ_S is idempotent.

Properties of $\mathfrak{U}$: Soundness and Completeness

Theorem (Soundness)

If $\mathrm{P}; \emptyset \Leftrightarrow^{+} \emptyset; \mathrm{S}$, then σ_{S} unifies any equation in P.

Theorem (Completeness)

If ϑ unifies every equation in P, then any maximal sequence of transformations $\mathrm{P}; \emptyset \Leftrightarrow \cdots$ ends in a system $\emptyset; S$ such that $\sigma_{\mathrm{S}} \lesssim \vartheta$.

Properties of $\mathfrak{U}$: Soundness and Completeness

Theorem (Soundness)

If $\mathrm{P}; \emptyset \Leftrightarrow^{+} \emptyset; S$, then σ_S unifies any equation in P.

Theorem (Completeness)

If ϑ unifies every equation in P, then any maximal sequence of transformations $\mathrm{P}; \emptyset \Leftrightarrow \cdots$ ends in a system $\emptyset; S$ such that $\sigma_S \lesssim \vartheta$.

Corollary

If P has no unifiers, then any maximal sequence of transformations from $\mathrm{P}; \emptyset$ must have the form $\mathrm{P}; \emptyset \Leftrightarrow \cdots \Leftrightarrow \bot$.

Observations

$\mathfrak{U}$ computes an idempotent mgu.

The choice of rules in computations via $\mathfrak{U}$ is “don’t care” nondeterminism (the word “any” in Completeness Theorem).

Any control strategy will result to an mgu for unifiable terms, and failure for non-unifiable terms.

Any practical algorithm that proceeds by performing transformations of $\mathfrak{U}$ in any order is

- sound and complete,
- generates mgus for unifiable terms.

Not all transformation sequences have the same length.

Not all transformation sequences end in exactly the same mgu.

Back to Resolution: Calculus for General Case

Two inference rules: Binary resolution and factoring.

A, B: atom, C, D: clauses, L: literal

- Binary resolution:

$$\frac{A \vee C \qquad \neg B \vee D}{\sigma(C \vee D)}$$

  where $\sigma = mgu(A, B)$.

- Factoring:

$$\frac{A \vee B \vee C}{\sigma(A \vee C)}$$

  where $\sigma = mgu(A, B)$.

Resolution: Soundness and Completeness

Theorem

Resolution calculus for general case is sound.

Theorem

Resolution calculus for general case is refutationally complete: If K is a set of clauses saturated wrt Res (i.e., $\mathrm{Res}(\mathrm{K}) \subseteq \mathrm{K}$) and $\mathrm{K} \models \square$, then $\square \in \mathrm{K}$.

Proof is based on the idea that in this case $\mathrm{ground}(\mathrm{K})$ is also saturated, $\mathrm{ground}(\mathrm{K}) \models \square$, and resolution calculus for ground clauses is refutationally complete.

Proving by Resolution

Given a set of clauses K and a hypothesis H, to prove H from K by resolution one should

1. Negate the hypothesis;
2. Add the negated hypothesis to K and start derivation, trying to obtain the contradiction;
3. In the derivation, use binary resolution and factoring rules to generate new clauses, add them to K;
4. If the empty clause appears, stop: contradiction found, H is proved;
5. If no step can be made and the empty clause is not found, then H can not be proved.

Example. Proving by Resolution

Show that the given set of clauses (1-3) is unsatisfiable:

1. $\neg p(x, y) \vee q(x, y)$.
2. $p(x, y) \vee q(y, x)$.
3. $\neg q(a, a) \vee \neg q(b, b)$

Example. Proving by Resolution

Show that the given set of clauses (1-3) is unsatisfiable:

1. $\neg p(x, y) \vee q(x, y)$.
2. $p(x, y) \vee q(y, x)$.
3. $\neg q(a, a) \vee \neg q(b, b)$
4. $q(x_1, y_1) \vee q(y_1, x_1)$. (Resolvent of 1 and 2)

Example. Proving by Resolution

Show that the given set of clauses (1-3) is unsatisfiable:

1. $\neg p(x, y) \vee q(x, y)$.
2. $p(x, y) \vee q(y, x)$.
3. $\neg q(a, a) \vee \neg q(b, b)$
4. $q(x_1, y_1) \vee q(y_1, x_1)$. (Resolvent of 1 and 2)
5. $q(x_1, x_1)$ (Factor of 4)

Example. Proving by Resolution

Show that the given set of clauses (1-3) is unsatisfiable:

1. $\neg p(x, y) \vee q(x, y)$.
2. $p(x, y) \vee q(y, x)$.
3. $\neg q(a, a) \vee \neg q(b, b)$
4. $q(x_1, y_1) \vee q(y_1, x_1)$. (Resolvent of 1 and 2)
5. $q(x_1, x_1)$ (Factor of 4)
6. $\neg q(b, b)$ (Resolvent of 5 and 3)

Example. Proving by Resolution

Show that the given set of clauses (1-3) is unsatisfiable:

1. $\neg p(x, y) \vee q(x, y)$.
2. $p(x, y) \vee q(y, x)$.
3. $\neg q(a, a) \vee \neg q(b, b)$
4. $q(x_1, y_1) \vee q(y_1, x_1)$. (Resolvent of 1 and 2)
5. $q(x_1, x_1)$ (Factor of 4)
6. $\neg q(b, b)$ (Resolvent of 5 and 3)
7. $\square$ (Resolvent of 5 and 6, contradiction found.)

Proving by Resolution

Unrestricted application of the inference rules might lead to search space explosion.

Most of the generated clauses are redundant.

Resolution strategies.

Redundancy elimination.

Ordered Resolution

One of most efficient resolution strategies.

Assumes a partial ordering on terms and literals.

Ordered inference:

- A subset of the literals is marked as maximal
- (If the clause is ground, i.e, without variables, the order is total, and the greatest literal is marked as maximal)
- The inference rules may be restricted in some cases so that they apply only to maximal literals.

Orderings

$\succ$: a well-founded ordering on atoms such that

- $\succ$ is total on ground atoms,
- $\succ$ is stable: $A \succ B$ implies $\sigma(A) \succ \sigma(B)$ for any atoms A and B and all substitutions σ.

Extension on literals:

- $\neg A \succ \neg B$ if $A \succ B$ for any atoms A and B.
- $\neg A \succ A$ for any atom A.

$\succeq \; := \; \succ \cup =$.

$\preceq \; := \; \succeq^{-1}$

Ground Ordered Resolution

A: atom, C, D: clauses, L: literal

- Ground ordered binary resolution:

$$\frac{A \vee C \qquad \neg A \vee D}{C \vee D},$$

  where $A \succ L$ for all L in C and $\neg A \succeq L$ for all L in D.

- Ground ordered positive factoring:

$$\frac{A \vee A \vee C}{A \vee C},$$

  where $A \succeq L$ for all L from C.

Ground Ordered Resolution

A: atom, C, D: clauses, L: literal

- Ground ordered binary resolution:

$$\frac{A \vee C \qquad \neg A \vee D}{C \vee D},$$

  where $A \succ L$ for all L in C and $\neg A \succeq L$ for all L in D.

- Ground ordered positive factoring:

$$\frac{A \vee A \vee C}{A \vee C},$$

  where $A \succeq L$ for all L from C.

The proofs remain correct.

Ordered Resolution: General Case

A, B: atoms, C, D: clauses, L: literal

- Ordered binary resolution:

$$\frac{A \vee C \qquad \neg B \vee D}{\sigma(C \vee D)},$$

where $\sigma = mgu(A, B)$, $\sigma(A) \not\preceq \sigma(L)$ for all L in C, and $\sigma(\neg B) \not\prec \sigma(L)$ for all L in D.

- Ordered positive factoring:

$$\frac{A \vee B \vee C}{\sigma(A \vee C)},$$

where $\sigma = mgu(A, B)$ and $\sigma(A) \not\prec \sigma(L)$ for all L in C.

Selection Function

A selection function is a mapping

$sel : C \longrightarrow$ set of occurrences of negative literals in C.

Will be used to further improve the inference system.

Intuition:

- If a clause has at least one selected literal, compute only inferences that involve a selected literal.
- If a clause has no selected literals, compute only inferences that involve a maximal literal.

Ordered Resolution with Selection: General Case

$Res^{\succ}_{sel}$, parametrized by $\succ$ and sel.

A, B: atoms, C, D: clauses, L: literal

Ordered binary resolution with selection:

$$\frac{A \vee C \qquad \neg B \vee D}{\sigma(C \vee D)},$$

Ordered Resolution with Selection: General Case

$Res^{\succ}_{sel}$, parametrized by $\succ$ and sel.

A, B: atoms, C, D: clauses, L: literal

Ordered binary resolution with selection:

$$\frac{A \vee C \qquad \neg B \vee D}{\sigma(C \vee D)},$$

where

- $\sigma = mgu(A, B)$,
- $\sigma(A) \not\preceq \sigma(L)$ for all L in C,
- $sel(A \vee C) = \emptyset$,
- $\neg B \in sel(\neg B \vee D)$, or
  $sel(\neg B \vee D) = \emptyset$ and $\sigma(\neg B) \not\prec \sigma(L)$ for all L in D.

Ordered Resolution with Selection: General Case

$\mathrm{Res}^{\succ}_{sel}$, parametrized by $\succ$ and sel.

A, B: atoms, C, D: clauses, L: literal

Ordered positive factoring with selection:

$$\frac{\mathrm{A} \vee \mathrm{B} \vee \mathrm{C}}{\sigma(\mathrm{A} \vee \mathrm{C})},$$

Ordered Resolution with Selection: General Case

$\mathrm{Res}^{\succ}_{sel}$, parametrized by $\succ$ and sel.

A, B: atoms, C, D: clauses, L: literal

Ordered positive factoring with selection:

$$\frac{A \vee B \vee C}{\sigma(A \vee C)},$$

where

- $\sigma = \mathrm{mgu}(A, B)$,
- $\sigma(A) \not\prec \sigma(L)$ for all L in C,
- $sel(A \vee B \vee C) = \emptyset$.

Ordered Resolution with Selection: General Case

Ordering and selection restrictions do not affect refutational completeness:

Theorem

Given $\succ$, sel, and a set of clauses K saturated wrt $Res^{\succ}_{sel}$ (i.e., $Res^{\succ}_{sel}(K) \subseteq K$), if $K \models \square$, then $\square \in K$.

Ordered Resolution with Selection

$p(a) \succ q(b)$. Selected literals are underlined. Compare:

1. $p(a) \vee q(b)$
2. $p(a) \vee \neg q(b)$
3. $\neg p(a) \vee q(b)$
4. $\neg p(a) \vee \neg q(b)$
5. $p(a) \vee p(a)$ (BR 1,2)
6. $p(a)$ (Factor, 5)
7. $\neg p(a) \vee \neg p(a)$ (BR 3,4)
8. $\neg p(a)$ (Factor, 7)
9. $\square$ (BR 6, 8)

1. $p(a) \vee q(b)$
2. $p(a) \vee \underline{\neg q(b)}$
3. $\neg p(a) \vee q(b)$
4. $\neg p(a) \vee \underline{\neg q(b)}$
5. $q(b) \vee q(b)$ (OBRS 1,3)
6. $q(b)$ (OPFS, 5)
7. $\neg p(a)$ (OBRS 6,4)
8. $p(a)$ (OBRS 6,2)
9. $\square$ (OBRS 7, 8)

Smaller search space with $Res^{\succ}_{s}el$.

Ordered Resolution with Selection

Smaller search space with $Res^{\succ}_{sel}$.

Rotation redundancies are avoided, e.g., in Res, two derivations of the same clause are possible:

1. $C_1 \vee A$
2. $C_2 \vee \neg A \vee B$
3. $C_3 \vee \neg B$
4. $C_1 \vee C_2 \vee B$ (BR 1,2)
5. $C_1 \vee C_2 \vee C_3$ (BR 3,4)

1. $C_1 \vee A$
2. $C_2 \vee \neg A \vee B$
3. $C_3 \vee \neg B$
4. $C_2 \vee \neg A \vee C_3$ (BR 2,3)
5. $C_1 \vee C_2 \vee C_3$ (BR 3,4)

If $A \succ B$, $Res^{\succ}_{sel}$ forbids the second derivation.

Redundancies

Ordering on clauses.

Treat clauses as multisets.

Multiset extension $\succ_{mul}$ of $\succ$:

$C_1 \succ_{mul} C_2$ iff there exist multisets $D_1 \neq \emptyset$ and D_2 such that

- $D_1 \subseteq C_1$,
- $C_2 = (C_1 - D_1) \cup D_2$,
- for each $d_2 \in D_2$ there is $d_1 \in D_1$ such that $d_1 \succ d_2$.

Redundancies

Ordering on clauses.

Treat clauses as multisets.

Multiset extension $\succ_{mul}$ of $\succ$:

$C_1 \succ_{mul} C_2$ iff there exist multisets $D_1 \neq \emptyset$ and D_2 such that

- $D_1 \subseteq C_1$,
- $C_2 = (C_1 - D_1) \cup D_2$,
- for each $d_2 \in D_2$ there is $d_1 \in D_1$ such that $d_1 \succ d_2$.

$\succ_{mul}$ is used to defined the notion of redundancy.

We reuse $\succ$ for $\succ_{mul}$.

Redundancies

Define for a set of ground clauses K and a ground clause C:

$$K_{\prec C} := \{D \in K \mid D \prec C\}$$

C is redundant wrt K if $K_{\prec C} \models C$.

C is redundant in K if $K_{\prec C} \models C$ and $C \in K$.

A general clause C is redundant wrt a set of general clauses K if all ground instances of C are redundant wrt $\mathrm{ground}(K)$.

Redundancies

Define for a set of ground clauses K and a ground clause C:

$$K_{\prec C} := \{D \in K \mid D \prec C\}$$

C is redundant wrt K if $K_{\prec C} \models C$.

C is redundant in K if $K_{\prec C} \models C$ and $C \in K$.

A general clause C is redundant wrt a set of general clauses K if all ground instances of C are redundant wrt $\mathrm{ground}(K)$.

Examples of redundancy:

- Tautologies: they are redundant wrt any K.
- Subsumption: $\sigma(C) \subset D$. D is redundant wrt $K \cup \{C\}$.

Inference with Redundancy Elimination

Consider an inference process in the inference system IS (called IS-run) with two kinds of step $\mathrm{K_i} \vdash \mathrm{K_{i+1}}$:

1. inference in IS,
2. elimination of redundancy: $\mathrm{K_{i+1}} = \mathrm{K_i} - \{\mathrm{C}\}$, if C is redundant in K.

Inference with Redundancy Elimination

Let $K_0 \vdash K_1 \vdash K_2 \vdash \cdots$ be an IS-run.

A clause C is called persistent in it if there exists i such that for all $j \geqslant i$, $C \in K_j$.

The limit K_ω of the run is the set of all persistent clauses:

$$K_\omega = \bigcup_{i \geqslant 0} \bigcap_{j \geqslant i} K_j.$$

Inference with Redundancy Elimination

Let $K_0 \vdash K_1 \vdash K_2 \vdash \cdots$ be a run.

The run is called IS-fair if every inference with persistent premises in K_ω has been applied, i.e, if

$$\frac{C_1 \ \cdots \ C_n}{C}$$

is an inference step in IS and $\{C_1, \ldots, C_n\} \subseteq K_\omega$, then there exists i such that $C \in K_i$.

$Res^{\succ}_{sel}$ with Redundancy Elimination

$Res^{\succ}_{sel}$ with Redundancy Elimination is refutationally complete:

Theorem

Let $K_0 \vdash K_1 \vdash K_2 \vdash \cdots$ be a $Res^{\succ}_{sel}$-fair run. If K_0 is unsatisfiable then $\square \in K_i$ for some i.

Implementation: Given Clause Algorithm

The clause set is split into two parts: active $\mathcal{A}$ and passive $\mathcal{P}$.

The set $\mathcal{A}$ contains already seen given clauses.

The clauses in $\mathcal{P}$ have not yet been selected as “given”.

From the beginning, $\mathcal{P}$ consists of the initial clauses.

Given Clause Algorithm: Main Loop

At each iteration:

- Select a new given clause C from P and remove is from P.
- Infer new clauses: conclusions of inferences between clauses from $\mathcal{A}$ and C.
- New clauses simplify and get simplified by clauses in active.
- If new clauses contain $\square$, the algorithm returns unsatisfiable.
- Add new clauses to P.
- Add C to $\mathcal{A}$.

Variations of the Algorithm

Otter loop: new clauses simplify and get simplified by passive.

Discount loop: passive clauses do not participate in simplification. Given clause participates in simplification inferences with active.