Proving and Solving in Unranked Theories

Part 1: Proving

Temur Kutsia

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

Plan

First-order logic

From ranked to unranked languages

Resolution and unranked unification

Unranked matching and transformations

Unranked anti-unification and generalization

Plan

First-order logic

From ranked to unranked languages

Resolution and unranked unification

Unranked matching and transformations

Unranked anti-unification and generalization

First-order logic

Warming up: examples of reasoning.

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

First-order logic

Warming up: 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.

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.

Assume a, b, and c are all equal. Assume also c and d are equal. Then a, b, c, and d are all equal.

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.

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

Alphabet

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

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

Ranked alphabet.

Alphabet

Notation:

- x, y, z for variables.
- f, g for function symbols.
- a, b, c for constants.
- p, q for predicate symbols.

Terms

Definition

- A variable is a term.
- If $t_1, \ldots, t_n$ are terms and $f \in \mathcal{F}^n$, then $f(t_1, \ldots, t_n)$ is a term.

Terms

Definition

- A variable is a term.
- If $t_1, \ldots, t_n$ are terms and $f \in \mathcal{F}^n$, then $f(t_1, \ldots, t_n)$ is a term.

Notation:

- s, t, r for terms.

Terms

Definition

- A variable is a term.
- If $t_1, \ldots, t_n$ are terms and $f \in \mathcal{F}^n$, then $f(t_1, \ldots, t_n)$ is a term.

Notation:

- s, t, r for terms.

Ground term: a term without variables.

Terms

Example

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

Terms

Example

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

Formulas

Definition

- If $t_1, \dots, t_n$ are terms and $p \in \mathcal{P}^n$, then $p(t_1, \dots, t_n)$ 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.

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:

- $succ$: binary predicate 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.

$$\forall x.(\exists y.succ(x, y) \wedge \forall z.(succ(x, z) \Rightarrow y \doteq z)))$$

Assume:

- $succ$: binary predicate 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.
- $succ$: binary predicate symbol for immediate successor.

Example

Translating English sentences into first-order logic formulas:

There is no natural number whose immediate successor is 0.

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

Assume:

- $zero$: constant for 0.
- $succ$: binary predicate symbol for immediate successor.

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.**

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: Id.

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:

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

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)

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

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)

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

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

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

$\vartheta\sigma = Id$.

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 $f \in \mathcal{F}^n$ an n-ary function $f_I : D^n \to D$, (in particular, $c_I \in D$ for each constant c)
- to each $p \in \mathcal{P}^n$, an n-ary relation p_I on D.

Semantics: interpretation function

The interpretation function assigns

- to each $f \in \mathcal{F}^n$ an n-ary function $f_I : D^n \to D$, (in particular, $c_I \in D$ for each constant c)
- to each $p \in \mathcal{P}^n$, an n-ary relation p_I on D.

Fixed arity functions and relations.

Variable assignment

A structure $S = (D, I)$ is given.

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

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:

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

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 $Val_{S,\sigma_S}(B) = true$.

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:

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

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.

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

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

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

Example

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

Define $S = (D, I)$ as

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

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

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 $\models A$.

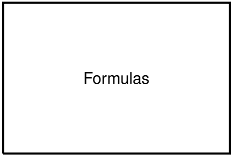
A formula A is unsatisfiable, if $\models_S A$ for no S.

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.

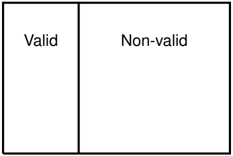


Validity, unsatisfiability

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.

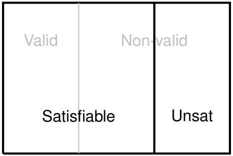


Validity, unsatisfiability

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.

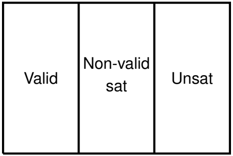


Validity, unsatisfiability

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. Then

1. A is valid iff $\neg A$ is unsatisfiable.
2. $B \models A$ iff $B \wedge \neg A$ 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 (an atomic formula or its negation),
- $L_1 \vee \cdots \vee L_k$ 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: $L_1 \vee \cdots \vee L_k$.

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 $\mathbb{IS}$ 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 $\mathbb{IS}$

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

  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 $K \models \Box$, then $K \vdash_{IS} \Box$, where $\Box$ is the empty clause.

Unification

To define the inferences rules of the resolution calculus, we need a method to solve equations between terms or atoms.

The method is called unification and will be discussed in detail a bit later.

Unification

To define the inferences rules of the resolution calculus, we need a method to solve equations between terms or atoms.

The method is called unification and will be discussed in detail a bit later.

The problem of unification: given two terms (or atoms) s and t, find a substitution σ such that $\sigma(s) = \sigma(t)$.

In other words: how can we replace variables by some terms in s and t to make them equal.

Unification: examples

$\{x \mapsto a, y \mapsto g(b)\}$ unifies $f(x, g(b))$ and $f(a, y)$.

Unification: examples

$\{x \mapsto a, y \mapsto g(b)\}$ unifies $f(x, g(b))$ and $f(a, y)$.

$\{x \mapsto a, y \mapsto g(a)\}$ unifies $f(x, g(x))$ and $f(a, y)$.

Unification: examples

$\{x \mapsto a, y \mapsto g(b)\}$ unifies $f(x, g(b))$ and $f(a, y)$.

$\{x \mapsto a, y \mapsto g(a)\}$ unifies $f(x, g(x))$ and $f(a, y)$.

$f(x, g(x))$ and $h(a, y)$ can not be unified.

Unification: examples

$\{x \mapsto a, y \mapsto g(b)\}$ unifies $f(x, g(b))$ and $f(a, y)$.

$\{x \mapsto a, y \mapsto g(a)\}$ unifies $f(x, g(x))$ and $f(a, y)$.

$f(x, g(x))$ and $h(a, y)$ can not be unified.

$f(x, x)$ and $f(a, b)$ can not be unified.

Unification: examples

$\{x \mapsto a, y \mapsto g(b)\}$ unifies $f(x, g(b))$ and $f(a, y)$.

$\{x \mapsto a, y \mapsto g(a)\}$ unifies $f(x, g(x))$ and $f(a, y)$.

$f(x, g(x))$ and $h(a, y)$ can not be unified.

$f(x, x)$ and $f(a, b)$ can not be unified.

x and $g(x)$ can not be unified.

Unification: examples

$\{x \mapsto a, y \mapsto g(b)\}$ unifies $f(x, g(b))$ and $f(a, y)$.

$\{x \mapsto a, y \mapsto g(a)\}$ unifies $f(x, g(x))$ and $f(a, y)$.

$f(x, g(x))$ and $h(a, y)$ can not be unified.

$f(x, x)$ and $f(a, b)$ can not be unified.

x and $g(x)$ can not be unified.

x and y can be unified by $\{x \mapsto y\}$ and by $\{x \mapsto t, y \mapsto t\}$ for any t, but $\{x \mapsto y\}$ is a better unifier than the others: it is most general.

Unification: examples

$\{x \mapsto a, y \mapsto g(b)\}$ unifies $f(x, g(b))$ and $f(a, y)$.

$\{x \mapsto a, y \mapsto g(a)\}$ unifies $f(x, g(x))$ and $f(a, y)$.

$f(x, g(x))$ and $h(a, y)$ can not be unified.

$f(x, x)$ and $f(a, b)$ can not be unified.

x and $g(x)$ can not be unified.

x and y can be unified by $\{x \mapsto y\}$ and by $\{x \mapsto t, y \mapsto t\}$ for any t, but $\{x \mapsto y\}$ is a better unifier than the others: it is most general.

We will define these notions formally and describe the unification algorithm when we will be discussing unranked logic.

Resolution calculus: inference rules

Two inference rules: Binary resolution and factoring.

A, B: atom, C, D: clauses.

- Binary resolution:

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

  where σ is a most general unifier of A and B.

- Factoring:

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

  where σ is a most general unifier of A and B.

Resolution calculus: inference rules

Two inference rules: Binary resolution and factoring.

A, B: atom, C, D: clauses.

- Binary resolution:

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

  where σ is a most general unifier of A and B.

- Factoring:

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

  where σ is a most general unifier of A and B.

Resolution calculus is sound and 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. $\neg p(x, y) \lor q(x, y)$.
2. $p(x, y) \lor q(y, x)$.
3. $\neg q(a, a) \lor \neg q(b, b)$

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. $\neg p(x, y) \lor q(x, y)$.
2. $p(x, y) \lor q(y, x)$.
3. $\neg q(a, a) \lor \neg q(b, b)$
4. $q(x_1, y_1) \lor q(y_1, x_1)$. (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.)

Ranked vs unranked

The language considered so far was ranked.

All function and predicate symbols had fixed arity, and they were interpreted by fixed arity functions and predicates as well.

Ranked vs unranked

The language considered so far was ranked.

All function and predicate symbols had fixed arity, and they were interpreted by fixed arity functions and predicates as well.

However, in practice often this may not be the case.

The same symbol might appear with different number of arguments in different places: no fixed arity.

Ranked vs unranked

Some examples of unranked symbols:

- symbols originating from different knowledge bases after their integration,
- tags in XML documents,
- names in Common Logic, variadic symbols in KIF,
- functions and constructors implemented in symbolic computation systems (e.g., Mathematica),
- arithmetic operations written in variadic form ($a + b$ and $a + b + c$),
- flexary symbols in OpenMath.

Ranked vs unranked

Some examples of unranked symbols:

- symbols originating from different knowledge bases after their integration,
- tags in XML documents,
- names in Common Logic, variadic symbols in KIF,
- functions and constructors implemented in symbolic computation systems (e.g., Mathematica),
- arithmetic operations written in variadic form ($a + b$ and $a + b + c$),
- flexary symbols in OpenMath.

Other terms for unranked symbols: variadic, flexary, polyadic, multiary, symbols of flexible arity, ...

Unranked alphabet

Unranked alphabets permit unranked function and predicate symbols.

For simplicity, we assume that our unranked alphabet contains only unranked function and predicate symbols (no ranked ones).

Unranked alphabet

Unranked alphabets permit unranked function and predicate symbols.

For simplicity, we assume that our unranked alphabet contains only unranked function and predicate symbols (no ranked ones).

There is one more feature, that helps to really take advantage of the unrankedness: sequence variables.

They stand for finite, possibly empty sequences of terms.

To distinguish, we call the standard variables the individual variables, since they stand for individual terms.

Unranked alphabet

An unranked alphabet consists of the following sets of symbols:

- A countable set of individual variables $\mathcal{V}_{\mathsf{ind}}$.
- A countable set of sequence variables $\mathcal{V}_{\mathsf{seq}}$.
- A set of unranked function symbols $\mathcal{F}$.
- A set of unranked predicate symbols $\mathcal{P}$.
- The logical and auxiliary symbols are the same as in a ranked alphabet.

Unranked terms

Defining terms in an unranked language:

Definition

- An individual variable is a term.
- If each of $s_1, \ldots, s_n$ is a term or a sequence variable, $n \geqslant 0$, and $f \in \mathcal{F}$, then $f(s_1, \ldots, s_n)$ is a term.

Notation: t, r for terms, s, q for terms or sequence variables.

Convention: We omit parentheses, when the argument sequence is empty, writing f instead of $f()$, etc.

Unranked formulas

Defining formulas in an unranked language:

Definition

- If each of $s_1, \ldots, s_n$ is a term or a sequence variable, $n \geqslant 0$, and $p \in \mathcal{P}$, then $p(s_1, \ldots, s_n)$ is an atomic formula (an atom).
- If A is a formula and $\overline{x}$ is a sequence variable, then $\exists\overline{x}.A$ and $\forall\overline{x}.A$ are formulas.
- Other formulas are defined as in the ranked case.

Unranked substitutions

Substitution: a function σ from

- individual variables to terms,
- sequence variables to finite sequences whose elements are terms or sequence variables,

such that the domain of σ is finite.

Unranked substitutions

Substitution: a function σ from

- individual variables to terms,
- sequence variables to finite sequences whose elements are terms or sequence variables,

such that the domain of σ is finite.

Example

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

- individual variable x to a term $f(\overline{x}, a)$,
- sequence variable $\overline{x}$ to a sequence $(f(a), \overline{z}, b)$,
- sequence variable $\overline{y}$ to the empty sequence, denoted by $()$.

Unranked substitutions

Applying $\sigma = \{x \mapsto f(\overline{x}, a),\ \overline{x} \mapsto (f(a), \overline{z}, b),\ \overline{y} \mapsto ()\}$ to some terms:

- $\sigma(g(\overline{x}, \overline{y})) = g(f(a), \overline{z}, b)$.
- $\sigma(f(x, \overline{y}, g(\overline{x}))) = f(f(\overline{x}, a), g(f(a), \overline{z}, b))$.
- $\sigma(x) = f(\overline{x}, a)$.
- $\sigma(f(a, y, b)) = f(a, y, b)$.

Semantics for unranked logic

Structures: $\mathcal{S} = (D, I)$.

$D^* := \cup_{i \geqslant 0} D^n$: the set of all finite sequences over D.

The interpretation function assigns

- to each $f \in \mathcal{F}$: a variadic function $f_I : D^* \to D$,
- to each $p \in \mathcal{P}$: variadic relation p_I on D, i.e., $p_I \subseteq D^*$.

Semantics for unranked logic

Structures: $\mathcal{S} = (\mathrm{D}, \mathrm{I})$.

$\mathrm{D}^* := \cup_{i \geqslant 0} \mathrm{D}^n$: the set of all finite sequences over D.

The interpretation function assigns

- to each $f \in \mathcal{F}$: a variadic function $f_{\mathrm{I}} : \mathrm{D}^* \to \mathrm{D}$,
- to each $p \in \mathcal{P}$: variadic relation p_{I} on D, i.e., $p_{\mathrm{I}} \subseteq \mathrm{D}^*$.

Variadic functions and relations.

Variable assignment

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

Variable assignment σ_S maps

- each $x \in \mathcal{V}_{\mathsf{ind}}$ into an element of D: $\sigma_S(x) \in D$,
- each $\overline{x} \in \mathcal{V}_{\mathsf{seq}}$ into an element of D^*: $\sigma_S(\overline{x}) \in D^*$,

Semantic counterpart of substitutions.

Define:

$$\sigma_S[\overline{x} \to (d_1, \ldots, d_n)](\overline{y}) := \begin{cases} \sigma_S(\overline{y}), & \text{if } \overline{x} \neq \overline{y} \\ (d_1, \ldots, d_n), & \text{otherwise.} \end{cases}$$

Interpretation of terms

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

Interpretations of unranked terms are defined like for their ranked counterparts.

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

Truth value of a formula:

- $Val_{S,\sigma_S}(p(t_1, \ldots, t_n)) = true$ iff
  $(Val_{S,\sigma_S}(t_1), \ldots, Val_{S,\sigma_S}(t_n)) \in p_I$.
- $Val_{S,\sigma_S}(\exists \overline{x}.A) = true$ iff
  $Val_{S,\sigma_S[\overline{x} \to (d_1,\ldots,d_n)]}(A) = true$
  for some $(d_1, \ldots, d_n) \in D^*$.
- $Val_{S,\sigma_S}(\forall \overline{x}.A) = true$ iff
  $Val_{S,\sigma_S[\overline{x} \to (d_1,\ldots,d_n)]}(A) = true$
  for all $(d_1, \ldots, d_n) \in D^*$.
- For other formulas: as in the ranked case.

The resolution calculus, again

The rules for the resolution calculus remain unchanged.

Unification should take into account sequence variables.

Example

Consider an unranked equality relation all_equal, which is true when all its arguments are identical, for any number of arguments. Prove that if $\mathtt{a} = \mathtt{b}$, $\mathtt{a} = \mathtt{c}$, and $\mathtt{a} = \mathtt{d}$, then $all\_equal(\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d})$ holds.

Example

Consider an unranked equality relation all_equal, which is true when all its arguments are identical, for any number of arguments. Prove that if $\mathsf{a} = \mathsf{b}$, $\mathsf{a} = \mathsf{c}$, and $\mathsf{a} = \mathsf{d}$, then $all\_equal(\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d})$ holds.

Axiomatize all_equal with the help of an unranked predicate and a sequence variable:

$all\_equal(\mathsf{x})$.
$all\_equal(\mathsf{x}, \mathsf{y}, \overline{z}) \vee \neg(\mathsf{x} \doteq \mathsf{y}) \vee \neg all\_equal(\mathsf{x}, \overline{z})$.

Example

Consider an unranked equality relation all_equal, which is true when all its arguments are identical, for any number of arguments. Prove that if $a = b$, $a = c$, and $a = d$, then $all\_equal(a, b, c, d)$ holds.

Axiomatize all_equal with the help of an unranked predicate and a sequence variable:

$all\_equal(x)$.
$all\_equal(x, y, \overline{z}) \vee \neg(x \doteq y) \vee \neg all\_equal(x, \overline{z})$.

Assume $a \doteq b$, $a \doteq c$, and $a \doteq d$.

Prove $all\_equal(a, b, c, d)$.

Example

(1) $all\_equal(x)$.

(2) $all\_equal(x, y, \overline{z}) \vee \neg(x \doteq y) \vee \neg all\_equal(x, \overline{z})$.

(3) $a \doteq b$.

(4) $a \doteq c$.

(5) $a \doteq d$.

(6) $\neg all\_equal(a, b, c, d)$. (Negation of the hypothesis)

Example

(1) $all\_equal(x)$.

(2) $all\_equal(x, y, \overline{z}) \vee \neg(x \doteq y) \vee \neg all\_equal(x, \overline{z})$.

(3) $a \doteq b$.

(4) $a \doteq c$.

(5) $a \doteq d$.

(6) $\neg all\_equal(a, b, c, d)$. (Negation of the hypothesis)

(7) $\neg(a \doteq b) \vee \neg all\_equal(a, c, d)$. (Resolvent of (6) and (2))

Example

(1) $\mathit{all\_equal}(x)$.

(2) $\mathit{all\_equal}(x, y, \overline{z}) \vee \neg(x \doteq y) \vee \neg \mathit{all\_equal}(x, \overline{z})$.

(3) $a \doteq b$.

(4) $a \doteq c$.

(5) $a \doteq d$.

(6) $\neg \mathit{all\_equal}(a, b, c, d)$. (Negation of the hypothesis)

(7) $\neg(a \doteq b) \vee \neg \mathit{all\_equal}(a, c, d)$. (Resolvent of (6) and (2))

(8) $\neg \mathit{all\_equal}(a, c, d)$. (Resolvent of (7) and (3))

Example

(1) $all\_equal(x).$

(2) $all\_equal(x, y, \overline{z}) \vee \neg(x \doteq y) \vee \neg all\_equal(x, \overline{z}).$

(3) $a \doteq b.$

(4) $a \doteq c.$

(5) $a \doteq d.$

(6) $\neg all\_equal(a, b, c, d).$ (Negation of the hypothesis)

(7) $\neg(a \doteq b) \vee \neg all\_equal(a, c, d).$ (Resolvent of (6) and (2))

(8) $\neg all\_equal(a, c, d).$ (Resolvent of (7) and (3))

(9) $\neg(a \doteq c) \vee \neg all\_equal(a, d).$ (Resolvent of (8) and (2))

Example

(1) $all\_equal(x).$

(2) $all\_equal(x, y, \overline{z}) \vee \neg(x \doteq y) \vee \neg all\_equal(x, \overline{z}).$

(3) $a \doteq b.$

(4) $a \doteq c.$

(5) $a \doteq d.$

(6) $\neg all\_equal(a, b, c, d).$ (Negation of the hypothesis)

(7) $\neg(a \doteq b) \vee \neg all\_equal(a, c, d).$ (Resolvent of (6) and (2))

(8) $\neg all\_equal(a, c, d).$ (Resolvent of (7) and (3))

(9) $\neg(a \doteq c) \vee \neg all\_equal(a, d).$ (Resolvent of (8) and (2))

(10) $\neg all\_equal(a, d).$ (Resolvent of (9) and (4))

Example

(1) $\mathit{all\_equal}(x)$.

(2) $\mathit{all\_equal}(x, y, \overline{z}) \vee \neg(x \doteq y) \vee \neg \mathit{all\_equal}(x, \overline{z})$.

(3) $a \doteq b$.

(4) $a \doteq c$.

(5) $a \doteq d$.

(6) $\neg \mathit{all\_equal}(a, b, c, d)$. (Negation of the hypothesis)

(7) $\neg(a \doteq b) \vee \neg \mathit{all\_equal}(a, c, d)$. (Resolvent of (6) and (2))

(8) $\neg \mathit{all\_equal}(a, c, d)$. (Resolvent of (7) and (3))

(9) $\neg(a \doteq c) \vee \neg \mathit{all\_equal}(a, d)$. (Resolvent of (8) and (2))

(10) $\neg \mathit{all\_equal}(a, d)$. (Resolvent of (9) and (4))

(11) $\neg(a \doteq d) \vee \neg \mathit{all\_equal}(a)$. (Resolvent of (10) and (2))

Example

(1) $\mathit{all\_equal}(x)$.

(2) $\mathit{all\_equal}(x, y, \overline{z}) \lor \neg(x \doteq y) \lor \neg \mathit{all\_equal}(x, \overline{z})$.

(3) $a \doteq b$.

(4) $a \doteq c$.

(5) $a \doteq d$.

(6) $\neg \mathit{all\_equal}(a, b, c, d)$. (Negation of the hypothesis)

(7) $\neg(a \doteq b) \lor \neg \mathit{all\_equal}(a, c, d)$. (Resolvent of (6) and (2))

(8) $\neg \mathit{all\_equal}(a, c, d)$. (Resolvent of (7) and (3))

(9) $\neg(a \doteq c) \lor \neg \mathit{all\_equal}(a, d)$. (Resolvent of (8) and (2))

(10) $\neg \mathit{all\_equal}(a, d)$. (Resolvent of (9) and (4))

(11) $\neg(a \doteq d) \lor \neg \mathit{all\_equal}(a)$. (Resolvent of (10) and (2))

(12) $\neg \mathit{all\_equal}(a)$. (Resolvent of (11) and (5))

Example

(1) $\mathit{all\_equal}(x)$.

(2) $\mathit{all\_equal}(x, y, \overline{z}) \vee \neg(x \doteq y) \vee \neg \mathit{all\_equal}(x, \overline{z})$.

(3) $a \doteq b$.

(4) $a \doteq c$.

(5) $a \doteq d$.

(6) $\neg \mathit{all\_equal}(a, b, c, d)$. (Negation of the hypothesis)

(7) $\neg(a \doteq b) \vee \neg \mathit{all\_equal}(a, c, d)$. (Resolvent of (6) and (2))

(8) $\neg \mathit{all\_equal}(a, c, d)$. (Resolvent of (7) and (3))

(9) $\neg(a \doteq c) \vee \neg \mathit{all\_equal}(a, d)$. (Resolvent of (8) and (2))

(10) $\neg \mathit{all\_equal}(a, d)$. (Resolvent of (9) and (4))

(11) $\neg(a \doteq d) \vee \neg \mathit{all\_equal}(a)$. (Resolvent of (10) and (2))

(12) $\neg \mathit{all\_equal}(a)$. (Resolvent of (11) and (5))

(13) $\square$ (Resolvent of (12) and (1))

Bad news

Unranked logic, as we defined it, is not compact, in contrast to ranked first-order logic.

Bad news

Unranked logic, as we defined it, is not compact, in contrast to ranked first-order logic.

Counterexample of compactness. An infinite set consisting of:

$$\begin{aligned}
\exists \overline{x}.&\ p(\overline{x}) \\
&\ \neg p \\
\forall x_1.&\ \neg p(x_1) \\
\forall x_1, x_2.&\ \neg p(x_1, x_2) \\
\forall x_1, x_2, x_3.&\ \neg p(x_1, x_2, x_3) \\
\ldots&
\end{aligned}$$

Every finite subset of this set has a model, but the entire set does not.

In this respect, the unranked clausal fragment behaves well.

A problem with the unranked clausal fragment

We have formulated two inference rules for the clausal fragment (both for ranked and unranked cases): binary resolution and factoring.

Let us recall them.

A problem with the unranked clausal fragment

A, B: atom, C, D: clauses.

- Binary resolution:

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

  where σ is a most general unifier of A and B.

- Factoring:

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

  where σ is a most general unifier of A and B.

A problem with the unranked clausal fragment

A, B: atom, C, D: clauses.

- Binary resolution:

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

  where σ is a most general unifier of A and B.

- Factoring:

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

  where σ is a most general unifier of A and B.

In the unranked case, A and B might have infinitely many most general unifiers.

A problem with the unranked clausal fragment

In the unranked case, two terms (two atoms) might have infinitely many most general unifiers.

For instance, $f(a, \overline{x})$ and $f(\overline{x}, a)$ have infinitely many mgus: $\{\overline{x} \mapsto ()\}, \{\overline{x} \mapsto a\}, \{\overline{x} \mapsto (a, a)\}, \ldots$.

It means that the inference rules can be potentially applied in infinitely many different ways for the given two clauses.

We need finitely many alternatives.

We should study unranked unification in more detail and identify so called finitary cases.