Selected Topics from Unification Theory

Temur Kutsia

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

Overview

Syntactic Unification

Equational Unification

What is Unification

- Goal: Identify two symbolic expressions.
- Method: Replace certain subexpressions (variables) by other expressions.

What is Unification

- Goal: Identify two symbolic expressions.
- Method: Replace certain subexpressions (variables) by other expressions.

Example

- Goal: Identify $f(x, a)$ and $f(b, y)$.
- Method: Replace the variable x by b, and the variable y by a. Both initial expressions become $f(b, a)$.

What is Unification

- Goal: Identify two symbolic expressions.
- Method: Replace certain subexpressions (variables) by other expressions.

Example

- Goal: Identify $f(x, a)$ and $f(b, y)$.
- Method: Replace the variable x by b, and the variable y by a. Both initial expressions become $f(b, a)$.
- Of course, one should know what expressions are variables, and what are not.
  (Syntax: variables, function symbols, terms, etc.)

What is Unification

- Goal: Identify two symbolic expressions.
- Method: Replace certain subexpressions (variables) by other expressions.

Example

- Goal: Identify $f(x, a)$ and $f(b, y)$.
- Method: Replace the variable x by b, and the variable y by a. Both initial expressions become $f(b, a)$.
- Of course, one should know what expressions are variables, and what are not.
  (Syntax: variables, function symbols, terms, etc.)
- The *substitution* $\{x \mapsto b, y \mapsto a\}$ *unifies* the *terms* $f(x, a)$ and $f(b, y)$.

What is Unification

- Goal: Identify two symbolic expressions.
- Method: Replace certain subexpressions (variables) by other expressions.

Example

- Goal: Identify $f(x,a)$ and $f(b,y)$.
- Method: Replace the variable x by b, and the variable y by a. Both initial expressions become $f(b,a)$.
- Of course, one should know what expressions are variables, and what are not.
  (Syntax: variables, function symbols, terms, etc.)
- The *substitution* $\{x \mapsto b, y \mapsto a\}$ *unifies* the *terms* $f(x,a)$ and $f(b,y)$.
- Solving the equation $f(x,a) = f(b,y)$ for x and y.

What is Unification

- Goal of unification: Identify two symbolic expressions.
- Method: Replace certain subexpressions (variables) by other expressions.

Depending what is meant under "identify" (syntactic identity or equality modulo some equations) one speaks about *syntactic unification* or *equational unification*.

Example

- The terms $f(x, a)$ and $g(a, x)$ are not syntactically unifiable.
- However, they are unifiable modulo the equation $f(a, a) = g(a, a)$ with the substitution $\{x \mapsto a\}$.

What is Unification

- Goal of unification: Identify two symbolic expressions.
- Method: Replace certain subexpressions (variables) by other expressions.

Depending at which positions the variables are allowed to occur, and which kind of expressions they are allowed to be replaced by, one speaks about *first-order unification* or *higher-order unification*.

Example

- If G and x are variables, the terms $f(x, a)$ and $G(a, x)$ can not be subjected to first-order unification.
- $G(a, x)$ is not a first-order term: G occurs in the top position.
- However, $f(x, a)$ and $G(a, x)$ can be unified by higher-order unification with the substitution $\{x \mapsto a, G \mapsto f\}$.

What is Unification Good For?

- To make an inference step in theorem proving.
- To perform an inference in logic programming.
- To make a rewriting step in term rewriting.
- To generate a critical pair in completion.
- To extract a part from structured or semistructured data.
- For type inference in programming languages.
- For matching in pattern-based languages.
- For program schemas manipulation.
- For various formalisms in computational linguistics.
- etc.

Subject of Today's Lecture

- Syntactic unification and matching.
- Generalizations to the equational case.

Notation

- First-order language.
- $\mathcal{F}$: Set of function symbols.
- $\mathcal{V}$: Set of variables.
- x, y, z: Variables.
- a, b, c: Constants.
- f, g, h: Arbitrary function symbols.
- s, t, r: Terms.
- $\mathcal{T}(\mathcal{F}, \mathcal{V})$: Set of terms over $\mathcal{F}$ and $\mathcal{V}$.
- Equation: a pair of terms, written $s \doteq t$.
- $vars(t)$: The set of variables in t. This notation will be used also for sets of terms, equations, and sets of equations.

Substitutions

Substitution

- A mapping from variables to terms, where all but finitely many variables are mapped to themselves.

Example

A substitution is represented as a set of *bindings*:

- $\{x \mapsto f(a, b), y \mapsto z\}$.
- $\{x \mapsto f(x, y), y \mapsto f(x, y)\}$.

All variables except x and y are mapped to themselves by these substitutions.

Substitutions

Notation

- σ, ϑ, η, ρ denote arbitrary substitutions.
- ε denotes the identity substitution.

Substitutions

Substitution Application

Applying a substitution σ to a term t:

$$t\sigma = \begin{cases} \sigma(x) & \text{if } t = x \\ f(t_1\sigma, \ldots, t_n\sigma) & \text{if } t = f(t_1, \ldots, t_n) \end{cases}$$

Example

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

Substitutions

Domain, Range, Variable Range

For a substitution σ:

- The *domain* is the set of variables:

$$dom(\sigma) = \{x \mid x\sigma \neq x\}.$$

- The *range* is the set of terms:

$$ran(\sigma) = \bigcup_{x \in dom(\sigma)} \{x\sigma\}.$$

- The *variable range* is the set of variables:

$$vran(\sigma) = vars(ran(\sigma)).$$

Substitutions

Example (Domain, Range, Variable Range)

$$dom(\{x \mapsto f(a,y),\ y \mapsto g(z)\}) = \{x, y\}$$
$$ran(\{x \mapsto f(a,y),\ y \mapsto g(z)\}) = \{f(a,y), g(z)\}$$
$$vran(\{x \mapsto f(a,y),\ y \mapsto g(z)\}) = \{y, z\}$$

$$dom(\varepsilon) = ran(\varepsilon) = vran(\varepsilon) = \varnothing$$

Substitutions

Composition of Substitutions

- Written: $\sigma\vartheta$.
- $t(\sigma\vartheta) = (t\sigma)\vartheta$.

Example

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

Composition is associative but not commutative:

$$\vartheta\sigma = \{x \mapsto a, y \mapsto b\} \neq \sigma\vartheta.$$

Substitutions

Instantiation Quasi-Ordering

- A substitution σ is *more general* than ϑ, written $\sigma \preceq \vartheta$, if there exists η such that $\sigma\eta = \vartheta$.
- The relation $\preceq$ is quasi-ordering (reflexive and transitive binary relation), called *instantiation quasi-ordering.*
- $\eqsim$ is the equivalence relation corresponding to $\preceq$.

Example

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

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

Substitutions

Unifier, Most General Unifier, Unification Problem

- A substitution σ is a *unifier* of the terms s and t if $s\sigma = t\sigma$.
- A unifier σ of s and t is a *most general unifier (mgu)* if $\sigma \leq \vartheta$ for every unifier ϑ of s and t.
- A *unification problem* for s and t is represented as $s \doteq^{?} t$.

Substitutions

Example (Unifier, Most General Unifier)

Unification problem: $f(x,z) \doteq^{?} f(y,g(a))$.

- Some of the unifiers:

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

- mgu's: $\{x \mapsto y, z \mapsto g(a)\}$, $\{y \mapsto x, z \mapsto g(a)\}$.
- mgu is unique up to a variable renaming:

$$\{x \mapsto y, z \mapsto g(a)\} \simeq \{y \mapsto x, z \mapsto g(a)\}$$

Outline

Unification Algorithm

- Goal: Design an algorithm that for a given unification problem $s \doteq^{?} t$
  - returns an mgu of s and t if they are unifiable,
  - reports failure otherwise.

Naive Algorithm

Write down two terms and set markers at the beginning of the terms. Then:

1. Move the markers simultaneously, one symbol at a time, until both move off the end of the term (**success**), or until they point to two different symbols;
2. If the two symbols are both non-variables, then **fail**; otherwise, one is a variable (call it x) and the other one is the first symbol of a subterm (call it t):
   - If x occurs in t, then **fail**;
   - Otherwise, replace x everywhere from the marker positions by t (including in the solution), write down "$x \mapsto t$" as a part of the solution, and return to 1.

Naive Algorithm

- Finds disagreements in the two terms to be unified.
- Attempts to repair the disagreements by binding variables to terms.
- Fails when function symbols clash, or when an attempt is made to unify a variable with a term containing that variable.

Example

$$f(x, g(a), g(z))$$

$$\uparrow$$

$$f(g(y), g(y), g(g(x)))$$

$$\uparrow$$

Example

$f(x, g(a), g(z))$

$f(g(y), g(y), g(g(x)))$

Example

$$f(x, g(a), g(z))$$

$$f(g(y), g(y), g(g(x)))$$

Example

$$f(x, g(a), g(z))$$

$$f(g(y), g(y), g(g(x)))$$

Example

$$f(g(y), g(a), g(z))$$

$$f(g(y), g(y), g(g(g(y))))$$

Example

$$f(g(y), g(a), g(z))$$

$$f(g(y), g(y), g(g(g(y))))$$

$$\{x \mapsto g(y)\}$$

Example

$$f(g(y), g(a), g(z))$$

$$f(g(y), g(y), g(g(g(y))))$$

$$\{x \mapsto g(y)\}$$

Example

$$f(g(y), g(a), g(z))$$

$$f(g(y), g(y), g(g(g(y))))$$

$$\{x \mapsto g(y)\}$$

Example

$$f(g(y), g(a), g(z))$$

$$f(g(y), g(a), g(g(g(a))))$$

$$\{x \mapsto g(a)\}$$

Example

$$f(g(y), g(a), g(z))$$

$$f(g(y), g(a), g(g(g(a))))$$

$$\{x \mapsto g(a), y \mapsto a\}$$

Example

$$f(g(y), g(a), g(z))$$

$$f(g(y), g(a), g(g(g(a))))$$

$$\{x \mapsto g(a), y \mapsto a\}$$

Example

$$f(g(y), g(a), g(z))$$

$$f(g(y), g(a), g(g(g(a))))$$

$$\{x \mapsto g(a), y \mapsto a\}$$

Example

$$f(g(y), g(a), g(z))$$

$$f(g(y), g(a), g(g(g(a))))$$

$$\{x \mapsto g(a), y \mapsto a\}$$

Example

$$f(g(y), g(a), g(g(g(a))))$$

$$f(g(y), g(a), g(g(g(a))))$$

$$\{x \mapsto g(a), y \mapsto a\}$$

Example

$$f(g(y), g(a), g(g(g(a))))$$

$$f(g(y), g(a), g(g(g(a))))$$

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

Example

$$f(g(y), g(a), g(g(g(a))))$$

$$f(g(y), g(a), g(g(g(a))))$$

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

Interesting Questions

Correctness:

- Does the algorithm always terminate?
- Does it always produce an mgu for two unifiable terms, and fail for non-unifiable terms?
- Do these answers depend on the order of operations?

Implementation:

- What data structures should be used for terms and substitutions?
- How should application of a substitution be implemented?
- What order should the operations be performed in?

Complexity:

- How much space does this take, and how much time?

Rule-Based Formulation of Unification

- Unification algorithm in a rule-base way.
- Repeated transformation of a set of equations.
- The left-to-right search for disagreements: modeled by term decomposition.

The Inference System $\mathcal{U}$

- A set of equations in *solved form*:

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

  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 $\mathcal{U}$

- *System*: The symbol $\bot$ or a pair $P; S$ where
  - P is a set of unification problems,
  - S is a set of equations in solved form.
- $\bot$ represents failure.
- A unifier (or a solution) of a system $P; S$: A substitution that unifies each of the equations in P and S.
- $\bot$ has no unifiers.

The Inference System $\mathcal{U}$

Example

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

The Inference System $\mathcal{U}$

Six transformation rules on systems:[1]

Trivial:

$$\{s \stackrel{?}{\doteq} s\} \uplus P'; S \Longrightarrow P'; S.$$

Decomposition:

$$\{f(s_1, \ldots, s_n) \stackrel{?}{\doteq} f(t_1, \ldots, t_n)\} \uplus P'; S \Longrightarrow$$

$$\{s_1 \stackrel{?}{\doteq} t_1, \ldots, s_n \stackrel{?}{\doteq} t_n\} \cup P'; S, \qquad \text{where } n \geq 0.$$

Symbol Clash:

$$\{f(s_1, \ldots, s_n) \stackrel{?}{\doteq} g(t_1, \ldots, t_m)\} \uplus P'; S \Longrightarrow \bot, \qquad \text{if } f \neq g.$$

[1] $\uplus$ stands for disjoint union.

The Inference System $\mathcal{U}$

Orient:

$$\{t \stackrel{?}{\doteq} x\} \uplus P'; S \Longrightarrow \{x \stackrel{?}{\doteq} t\} \cup P'; S, \text{ if } t \text{ is not a variable.}$$

Occurs Check:

$$\{x \stackrel{?}{\doteq} t\} \uplus P'; S \Longrightarrow \bot, \qquad \text{if } x \in vars(t) \text{ but } x \neq t.$$

Variable Elimination:

$$\{x \stackrel{?}{\doteq} t\} \uplus P'; S \Longrightarrow P'\{x \mapsto t\}; S\{x \mapsto t\} \cup \{x \doteq t\},$$

if $x \notin vars(t)$.

Unification with $\mathcal{U}$

In order to unify s and t:

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

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

Properties of $\mathcal{U}$: Termination

Lemma

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

$$P; \varnothing \Longrightarrow P_1; S_1 \Longrightarrow P_2; S_2 \Longrightarrow \cdots$$

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

Corollary

If $P; \varnothing \Longrightarrow^+ \varnothing; S$ then σ_S is idempotent.

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

Theorem (Soundness)

If $P; \varnothing \Longrightarrow^{+} \varnothing; S$, then σ_S unifies any equation in P.

Theorem (Completeness)

If ϑ unifies every equation in P, then any maximal sequence of transformations $P; \varnothing \Longrightarrow \cdots$ ends in a system $\varnothing; S$ such that $\sigma_S \leq \vartheta$.

Observations

- $\mathcal{U}$ computes an idempotent mgu.
- The choice of rules in computations via $\mathcal{U}$ 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 $\mathcal{U}$ 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.

Unification via $\mathcal{U}$: Exponential in Time and Space

Example

Unifying s and t, where

$$s = h(x_1, x_2, \ldots, x_n, f(y_0, y_0), f(y_1, y_1), \ldots, f(y_{n-1}, y_{n-1}), y_n)$$
$$t = h(f(x_0, x_0), f(x_1, x_1), \ldots, f(x_{n-1}, x_{n-1}), y_1, y_2, \ldots, y_n, x_n)$$

will create an mgu where each x_i and each y_i is bound to a term with $2^{i+1} - 1$ symbols:

$$\{x_1 \mapsto f(x_0, x_0),\ x_2 \mapsto f(f(x_0, x_0), f(x_0, x_0)),\ \ldots,$$
$$y_0 \mapsto x_0,\ y_1 \mapsto f(x_0, x_0),\ y_2 \mapsto f(f(x_0, x_0), f(x_0, x_0)),\ \ldots\}$$

Efficiency of Unification Algorithms

- There are algorithms with better complexities:
  - quadratic (Corbin & Bidoit),
  - almost linear (Huet, Martelli & Montanari, ...),
  - linear (Patterson & Wegman, ...)
- They require more sophisticated data structures.
- Still, the exponential algorithm is quite popular since it performs well in practice.

Matching

Matcher, Matching Problem

- A substitution σ is a *matcher* of s to t if $s\sigma = t$.
- A matching problem between s and t is represented as $s \ll^{?} t$.

Matching vs Unification

Example

$f(x,y) \ll^? f(g(z),c)$	$f(x,y) \doteq^? f(g(z),c)$
$\{x \mapsto g(z), y \mapsto c\}$	$\{x \mapsto g(z), y \mapsto c\}$
$f(x,y) \ll^? f(g(z),x)$	$f(x,y) \doteq^? f(g(z),x)$
$\{x \mapsto g(z), y \mapsto x\}$	$\{x \mapsto g(z), y \mapsto g(z)\}$
$f(x,a) \ll^? f(b,y)$	$f(x,a) \doteq^? f(b,y)$
No matcher	$\{x \mapsto b, y \mapsto a\}$
$f(x,x) \ll^? f(x,a)$	$f(x,x) \doteq^? f(x,a)$
No matcher	$\{x \mapsto a\}$
$x \ll^? f(x)$	$x \doteq^? f(x)$
$\{x \mapsto f(x)\}$	No unifier

How to Solve Matching Problems

- $s \doteq^? t$ and $s \ll^? t$ coincide, if t is ground.
- When t is not ground in $s \ll^? t$, simply regard all variables in t as constants and use the unification algorithm.
- Alternatively, modify the rules in $\mathcal{U}$ to work directly with the matching problem.

Matched Form

- A set of equations $\{x_1 \ll t_1, \ldots, x_n \ll t_n\}$ is in matched from, if all x's are pairwise distinct.
- The notation σ_S extends to matched forms.
- If S is in matched form, then

$$\sigma_S(x) = \begin{cases} t, & \text{if } x \ll t \in S \\ x, & \text{otherwise} \end{cases}$$

The Inference System $\mathcal{M}$

- *Matching system*: The symbol $\perp$ or a pair $P; S$, where
  - P is set of matching problems.
  - S is set of equations in matched form.
- A matcher (or a solution) of a system $P; S$: A substitution that solves each of the matching equations in P and S.
- $\perp$ has no matchers.

The Inference System $\mathcal{M}$

Five transformation rules on matching systems:[2]

Decomposition:

$$\{f(s_1,\ldots,s_n) \ll^? f(t_1,\ldots,t_n)\} \uplus P'; S \Longrightarrow$$
$$\{s_1 \ll^? t_1,\ldots,s_n \ll^? t_n\} \cup P'; S, \quad \text{where } n \geq 0.$$

Symbol Clash:

$$\{f(s_1,\ldots,s_n) \ll^? g(t_1,\ldots,t_m)\} \uplus P'; S \Longrightarrow \bot, \quad \text{if } f \neq g.$$

[2] $\uplus$ stands for disjoint union.

The Inference System $\mathcal{M}$

Symbol-Variable Clash:

$$\{f(s_1, \ldots, s_n) \ll^? x\} \uplus P'; S \Longrightarrow \bot.$$

Merging Clash:

$$\{x \ll^? t_1\} \uplus P'; \{x \ll t_2\} \uplus S' \Longrightarrow \bot, \qquad \text{if } t_1 \neq t_2.$$

Elimination:

$$\{x \ll^? t\} \uplus P'; S \Longrightarrow P'; \{x \ll t\} \cup S,$$

if S does not contain $x \ll t'$ with $t \neq t'$.

Matching with $\mathcal{M}$

In order to match s to t

1. Create an initial system $\{s \ll^? t\}; \varnothing$.
2. Apply successively the rules from $\mathcal{M}$.

Matching with $\mathcal{M}$

Example

Match $f(x,f(a,x))$ to $f(g(a),f(a,g(a)))$:

$$\{f(x,f(a,x)) \ll^? f(g(a),f(a,g(a)))\};\ \varnothing \Longrightarrow_{\text{Decomposition}}$$

$$\{x \ll^? g(a),\ f(a,x) \ll^? f(a,g(a))\};\ \varnothing \Longrightarrow_{\text{Elimination}}$$

$$\{f(a,x) \ll^? f(a,g(a))\};\ \{x \ll g(a)\} \Longrightarrow_{\text{Decomposition}}$$

$$\{a \ll^? a,\ x \ll^? g(a)\};\ \{x \ll g(a)\} \Longrightarrow_{\text{Decomposition}}$$

$$\{x \ll^? g(a)\};\ \{x \ll g(a)\} \Longrightarrow_{\text{Merge}}$$

$$\varnothing;\ \{x \ll g(a)\}$$

Matcher: $\{x \mapsto g(a)\}$.

Matching with $\mathcal{M}$

Example

Match $f(x,x)$ to $f(x,a)$:

$$\{f(x,x) \ll^? f(x,a)\}; \varnothing \Longrightarrow_{\text{Decomposition}}$$

$$\{x \ll^? x,\ x \ll^? a\}; \varnothing \Longrightarrow_{\text{Elimination}}$$

$$\{x \ll^? a\}; \{x \ll x\} \Longrightarrow_{\text{Merging Clash}}$$

$$\bot$$

No matcher.

Properties of $\mathcal{M}$: Termination

Theorem

For any finite set of matching problems P, every sequence of transformations in $\mathcal{M}$ of the form $P;\varnothing \Longrightarrow P_1;S_1 \Longrightarrow P_2;S_2 \Longrightarrow \cdots$ terminates either with $\bot$ or with $\varnothing;S$, with S in matched form.

Properties of $\mathcal{M}$: Soundness and Completeness

Theorem (Soundness)

If $P; \varnothing \Longrightarrow^{+} \varnothing; S$*, then* σ_S *solves all matching equations in* P*.*

Theorem (Completeness)

If ϑ *is a matcher of* P*, then any maximal sequence of transformations* $P; \varnothing \Longrightarrow \cdots$ *ends in a system* $\varnothing; S$ *such that* $\sigma_S = \vartheta|_{v(P)}$*.*

Notation:

- $\vartheta|_{v(P)}$: The restriction of ϑ to the variables in the left hand sides of P.

Implementation: Matching vs. Unification

- Unlike matching, efficient unification algorithms require sophisticated data structures.
- When efficiency is an issue, matching should be implemented separately from unification.

Outline

Motivation

- Unifications algorithms are essential components for deduction systems.
- Simple integration of axioms that describe the properties of equality often leads to an unacceptable increase of search space.
- Proposed solution: To build equational axioms into inference, replacing syntactic unification with equational unification.

Motivation

Example

Given: AI-theory $\{f(f(x,y),z) \approx f(x,f(y,z)),\ f(x,x) \approx x\}$. Apply idempotence to the term

$$f(x_0,f(x_1,\ldots,f(x_{n-1},f(x_n,f(x_0,\ldots,f(x_{n-1},x_n)\ldots)))\ldots)).$$

Motivation

Example

Given: AI-theory $\{f(f(x,y),z) \approx f(x,f(y,z)),\ f(x,x) \approx x\}$. Apply idempotence to the term

$$f(x_0,f(x_1,\ldots,f(x_{n-1},f(x_n,f(x_0,\ldots,f(x_{n-1},x_n)\ldots)))\ldots)).$$

- Exponentially many ways of rearranging the parentheses with the help of associativity: Very time consuming if the prover has to search for the right one.

Motivation

Example

Given: AI-theory $\{f(f(x,y),z) \approx f(x,f(y,z)),\ f(x,x) \approx x\}$. Apply idempotence to the term

$$f(x_0,f(x_1,\ldots,f(x_{n-1},f(x_n,f(x_0,\ldots,f(x_{n-1},x_n)\ldots)))\ldots)).$$

- Exponentially many ways of rearranging the parentheses with the help of associativity: Very time consuming if the prover has to search for the right one.
- A human mathematician would use words instead of terms, i.e. would work modulo associativity, and apply idempotence $xx = x$ to the word $x_0 \cdots x_n x_0 \cdots x_n$ by unifying x with $x_0 \cdots x_n$.

Motivation

Example

Given: AI-theory $\{f(f(x,y),z) \approx f(x,f(y,z)),\ f(x,x) \approx x\}$. Apply idempotence to the term

$$f(x_0,f(x_1,\ldots,f(x_{n-1},f(x_n,f(x_0,\ldots,f(x_{n-1},x_n)\ldots)))\ldots)).$$

- Exponentially many ways of rearranging the parentheses with the help of associativity: Very time consuming if the prover has to search for the right one.
- A human mathematician would use words instead of terms, i.e. would work modulo associativity, and apply idempotence $xx = x$ to the word $x_0 \cdots x_n x_0 \cdots x_n$ by unifying x with $x_0 \cdots x_n$.
- To adopt this way of proceeding for a prover, we must replace the syntactic unification algorithm in the resolution step by associative unification.

Equational Theory

Equational Theory

- E: a set of equations over $\mathcal{T}(\mathcal{F}, \mathcal{V})$, called identities.
- Equational theory $\doteq_E$ defined by E: The least congruence relation on $\mathcal{T}(\mathcal{F}, \mathcal{V})$ closed under substitution and containing E

Equational Theory

Equational Theory

- E: a set of equations over $\mathcal{T}(\mathcal{F}, \mathcal{V})$, called identities.
- Equational theory $\doteq_E$ defined by E: The least congruence relation on $\mathcal{T}(\mathcal{F}, \mathcal{V})$ closed under substitution and containing E
  i.e., $\doteq_E$ is the least binary relation on $\mathcal{T}(\mathcal{F}, \mathcal{V})$ with the properties:
  - $E \subseteq \ \doteq_E$.
  - Reflexivity: $s \doteq_E s$ for all s.
  - Symmetry: If $s \doteq_E t$ then $t \doteq_E s$ for all s, t.
  - Transitivity: If $s \doteq_E t$ and $t \doteq_E r$ then $s \doteq_E r$ for all s, t, r.
  - Congruence: If $s_1 \doteq_E t_1, \ldots, s_n \doteq_E t_n$ then $f(s_1, \ldots, s_n) \doteq_E f(t_1, \ldots, t_n)$ for all s, t, n and n-ary f.
  - Closure under substitution: If $s \doteq_E t$ then $s\sigma \doteq_E t\sigma$ for all s, t, σ.

Notation, Terminology

- Identities: $s \approx t$.
- $s \doteq_E t$: The term s is equal modulo E to the term t.
- E will be called an equational theory as well (abuse of the terminology).
- $sig(E)$: The set of function symbols that occur in E.

Example

- $C := \{f(x,y) \approx f(y,x)\}$: f is commutative. $sig(C) = f$.
- $f(f(a,b),c) \doteq_C f(c,f(b,a))$.

Notation, Terminology

- Identities: $s \approx t$.
- $s \doteq_E t$: The term s is equal modulo E to the term t.
- E will be called an equational theory as well (abuse of the terminology).
- $sig(E)$: The set of function symbols that occur in E.

Example

- $C := \{f(x,y) \approx f(y,x)\}$: f is commutative. $sig(C) = f$.
- $f(f(a,b),c) \doteq_C f(c,f(b,a))$.
- $AU := \{f(f(x,y),z) \approx f(x,f(y,z)),\ f(x,e) \approx x,\ f(e,x) \approx x\}$: f is associative, e is unit. $sig(AU) = \{f,e\}$
- $f(a,f(x,f(e,a))) \doteq_{AU} f(f(a,x),a)$.

Notation, Terminology

E-Unification Problem, E-Unifier, E-Unifiability

- E: equational theory.
  $\mathcal{F}$: set of function symbols.
  $\mathcal{V}$: countable set of variables.
- E-Unification problem over $\mathcal{F}$: a finite set of equations

$$\Gamma = \{s_1 \stackrel{?}{\doteq}_E t_1, \ldots, s_n \stackrel{?}{\doteq}_E t_n\},$$

  where $s_i, t_i \in \mathcal{T}(\mathcal{F}, \mathcal{V})$.
- E-Unifier of Γ: a substitution σ such that

$$s_1\sigma \doteq_E t_1\sigma, \ldots, s_n\sigma \doteq_E t_n\sigma.$$

- $u_E(\Gamma)$: the set of E-unifiers of Γ.
  Γ is E-unifiable iff $u_E(\Gamma) \neq \varnothing$.

E-Unification vs Syntactic Unification

- Syntactic unification: a special case of E-unif. with $E = \varnothing$.
- Any syntactic unifier of an E-unification problem Γ is also an E-unifier of Γ.
- For $E \neq \varnothing$, $u_E(\Gamma)$ may contain a unifier that is not a syntactic unifier.

E-Unification vs Syntactic Unification

- Syntactic unification: a special case of E-unif. with $E = \varnothing$.
- Any syntactic unifier of an E-unification problem Γ is also an E-unifier of Γ.
- For $E \neq \varnothing$, $u_E(\Gamma)$ may contain a unifier that is not a syntactic unifier.

Example

- Terms $f(a, x)$ and $f(b, y)$:
  - Not syntactically unifiable.
  - Unifiable module commutativity of f.
    C-unifier: $\{x \mapsto b, y \mapsto a\}$

E-Unification vs Syntactic Unification

- Syntactic unification: a special case of E-unif. with $E = \varnothing$.
- Any syntactic unifier of an E-unification problem Γ is also an E-unifier of Γ.
- For $E \neq \varnothing$, $u_E(\Gamma)$ may contain a unifier that is not a syntactic unifier.

Example

- Terms $f(a, x)$ and $f(b, y)$:
  - Not syntactically unifiable.
  - Unifiable module commutativity of f.
    C-unifier: $\{x \mapsto b, y \mapsto a\}$
- Terms $f(a, x)$ and $f(y, b)$:
  - Have the most general syntactic unifier $\{x \mapsto b, y \mapsto a\}$.
  - If f is associative, then $u_A(\{f(a, x) \stackrel{?}{\doteq}_A f(y, b)\})$ contains additional A-unifiers, e.g. $\{x \mapsto f(z, b), y \mapsto f(a, z)\}$.

Notions Adapted

Instantiation Quasi-Ordering (Modified)

- E: equational theory. $\mathcal{X}$: set of variables.
- A substitution σ is *more general modulo* E *on* $\mathcal{X}$ than ϑ, written $\sigma \leq_E^{\mathcal{X}} \vartheta$, if there exists η such that $x\sigma\eta \doteq_E x\vartheta$ for all $x \in \mathcal{X}$.
- ϑ is called an E*-instance* of σ modulo E on $\mathcal{X}$.
- The relation $\leq_E^{\mathcal{X}}$ is quasi-ordering, called *instantiation quasi-ordering.*
- $\doteq_E^{\mathcal{X}}$ is the equivalence relation corresponding to $\leq_E^{\mathcal{X}}$.

No Single MGU

- When comparing unifiers of Γ, the set $\mathcal{X}$ is $vars(\Gamma)$.
- Unifiable E-unification problems might not have an mgu.

Example

- f is commutative.
- $\Gamma = \{f(x,y) \stackrel{?}{\doteq}_C f(a,b)\}$ has two C-unifiers:

$$\sigma_1 = \{x \mapsto a, y \mapsto b\}$$
$$\sigma_2 = \{x \mapsto b, y \mapsto a\}.$$

- On $vars(\Gamma) = \{x,y\}$, any unifier is equal to either σ_1 or σ_2.
- σ_1 and σ_2 are not comparable wrt $\dot{\leq}_C^{\{x,y\}}$.
- Hence, no mgu for Γ.

MCSU vs MGU

In E-unification, the role of mgu is taken on by a complete set of E-unifiers.

Complete and Minimal Complete Sets of E-Unifiers

- Γ: E-unification problem over $\mathcal{F}$.
- $\mathcal{X} = vars(\Gamma)$.
- $\mathcal{C}$ is a *complete set of E-unifiers* of Γ iff
  1. $\mathcal{C} \subseteq u_E(\Gamma)$: $\mathcal{C}$'s elements are E-unifiers of Γ, and
  2. For each $\vartheta \in u_E(\Gamma)$ there exists $\sigma \in \mathcal{C}$ such that $\sigma \leq_E^{\mathcal{X}} \vartheta$.
- $\mathcal{C}$ is a *minimal complete set of E-unifiers* ($mcsu_E$) of Γ if it is a complete set of E-unifiers of Γ and
  3. two distinct elements of $\mathcal{C}$ are not comparable wrt $\leq_E^{\mathcal{X}}$.
- σ is an mgu of Γ iff $mcsu_E(\Gamma) = \{\sigma\}$.

MCSU's

- $mcsu_E(\Gamma) = \varnothing$ if Γ is not E-unifiable.
- Minimal complete sets of unifiers do not always exist.
- When they exist, they may be infinite.
- When they exist, they are unique up to $=_E^{\mathcal{X}}$.

Unification Type

Unification Type of a Problem, Theory.

- E: equational theory.
- Γ: E-unification problem over $\mathcal{F}$.
- Γ has *unification type*
  - *unitary,* if $mcsu(\Gamma)$ has cardinality at most one,
  - *finitary,* if $mcsu(\Gamma)$ has finite cardinality,
  - *infinitary,* if $mcsu(\Gamma)$ has infinite cardinality,
  - *zero,* if $mcsu(\Gamma)$ does not exist.
- Abbreviation: type unitary - 1, finitary - ω, infinitary - ∞, zero - 0.
- Ordering: $1 < \omega < \infty < 0$.
- *Unification type* of E wrt $\mathcal{F}$: the maximal type of an E-unification problem over $\mathcal{F}$.

Unification Type

The unification type of an E-equational problem over $\mathcal{F}$ depends both

- on E, and
- on $\mathcal{F}$.

Examples and more details will follow.

Unification Type

Example (Type Unitary)

Syntactic unification.

- The empty equational theory $\varnothing$: Syntactic unification.
- Unitary wrt any $\mathcal{F}$ because any unifiable syntactic unification problem has an mgu.

Unification Type

Example (Type Finitary)

Commutative unification: $\{f(x,y) \approx f(y,x)\}$

- $\{f(x,y) \stackrel{?}{\doteq}_C f(a,b)\}$ does not have an mgu. C-unification is not unitary.

Unification Type

Example (Type Finitary)

Commutative unification: $\{f(x,y) \approx f(y,x)\}$

- $\{f(x,y) \stackrel{?}{\doteq}_C f(a,b)\}$ does not have an mgu. C-unification is not unitary.
- Show that it is finitary for any $\mathcal{F}$:
  - Let $\Gamma = \{s_1 \stackrel{?}{\doteq}_C t_1, \ldots, s_n \stackrel{?}{\doteq}_C t_n\}$ be a C-unification problem.
  - Consider all possible syntactic unification problems $\Gamma' = \{s'_1 \stackrel{?}{\doteq} t'_1, \ldots, s'_n \stackrel{?}{\doteq} t'_n\}$, where $s'_i \doteq_C s_i$ and $t'_i \doteq_C t_i$ for each $1 \leq i \leq n$.
  - There are only finitely many such Γ's, because the C-equivalence class for a given term t is finite.
  - It can be shown that collection of all mgu's of Γ's is a complete set of C-unifiers of Γ. This set if finite.
  - If this set is not minimal (often the case), it can be minimized by removing redundant C-unifiers.

Unification Type

Example (Type Infinitary)

Associative unification: $\{f(f(x,y),z) \approx f(x,f(y,z))\}$.

- $\{f(x,a) \stackrel{?}{\doteq}_A f(a,x)\}$ has an infinite *mcsu*:
  $\{\{x \mapsto a\}, \{x \mapsto f(a,a)\}, \{x \mapsto f(a,f(a,a))\}, \ldots\}$
- Hence, A-unification can not be unitary or finitary.
- It is not of type zero because any A-unification problem has an *mcsu* that can be enumerated by the procedure from

  

  G. Plotkin.
  Building in equational theories.
  In B. Meltzer and D. Michie, editors, *Machine Intelligence*, volume 7, pages 73–90. Edinburgh University Press, 1972.
- A-unification is infinitary for any $\mathcal{F}$.

Unification Type

Example (Type Zero)

Associative-Idempotent unification:
$\{f(f(x,y),z) \approx f(x,f(y,z)), f(x,x) \approx x\}$.

- $\{f(x,f(y,x)) \stackrel{?}{\doteq}_{AI} f(x,f(z,x))\}$ does not have a minimal complete set of unifiers, see

  

  F. Baader.
  Unification in idempotent semigroups is of type zero.
  J. Automated Reasoning, 2(3):283–286, 1986.

- AI-unification is of type zero.

Unification Type. Signature Matters

Associative-commutative unification with unit:

$$ACU = \{f(f(x,y),z) \approx f(x,f(y,z)), f(x,y) \approx f(y,x), f(x,e) \approx x\}.$$

- Any ACU problem built using only f and variables has an mgu (i.e. is unitary).
- There are ACU problems that contain function symbols other than f and e, which are finitary, not unitary.
  For instance, $mcsu(\{f(x,y) \stackrel{?}{\doteq}_{ACU} f(a,b)\})$ consists of four unifiers (which ones?).

Kinds of E-unification.

Kinds of E-Unification

One may distinguish three kinds of E-unification problems, depending on the function symbols that are allowed to occur in them.

E-Unification Problems: Elementary, with Constants, General.

- E: the set of identities defining an equational theory.
  Γ: an E-unification problem over $\mathcal{F}$.
- Γ is an elementary E-unification problem iff $\mathcal{F} = sig(E)$.
- Γ is an E-unification problem with constants iff $\mathcal{F} \setminus sig(E)$ consists of constants.
- Γ is a general E-unification problem iff $\mathcal{F} \setminus sig(E)$ may contain arbitrary function symbols.

Unification Types of Theories wrt Kinds

- Unification type of E wrt elementary unification: Maximal unification type of E wrt all $\mathcal{F}$ such that $\mathcal{F} = sig(E)$.
- Unification type of E wrt unification with constants: Maximal unification type of E wrt all $\mathcal{F}$ such that $\mathcal{F} \setminus sig(E)$ is a set of constants.
- Unification type of E wrt general unification: Maximal unification type of E wrt all $\mathcal{F}$ such that $\mathcal{F} \setminus sig(E)$ is a set of arbitrary function symbols.

Unification Types of Theories wrt Kinds

The same equational theory can have different unification types for different kinds. Examples:

- ACU (Abelian monoids): Unitary wrt elementary unification, finitary wrt unification with constants and general unification.
- AG (Abelian groups): Unitary wrt elementary unification and unification with constants, finitary wrt general unification.

Decision and Unification Procedures

- Decision procedure for an equational theory E (wrt $\mathcal{F}$): An algorithm that for each E-unification problem Γ (wrt $\mathcal{F}$) returns $success$ if Γ is E-unifiable, and $failure$ otherwise.
- E is decidable if it admits a decision procedure.

Decision and Unification Procedures

- Decision procedure for an equational theory E (wrt $\mathcal{F}$): An algorithm that for each E-unification problem Γ (wrt $\mathcal{F}$) returns $\mathit{success}$ if Γ is E-unifiable, and $\mathit{failure}$ otherwise.
- E is decidable if it admits a decision procedure.
- (Minimal) E-unification algorithm (wrt $\mathcal{F}$): An algorithm that computes a (minimal) finite complete set of E-unifiers for all E-unification problems over $\mathcal{F}$.
- E-unification algorithm yields a decision procedure for E.

Decision and Unification Procedures

- Decision procedure for an equational theory E (wrt $\mathcal{F}$): An algorithm that for each E-unification problem Γ (wrt $\mathcal{F}$) returns $\mathit{success}$ if Γ is E-unifiable, and $\mathit{failure}$ otherwise.
- E is decidable if it admits a decision procedure.
- (Minimal) E-unification algorithm (wrt $\mathcal{F}$): An algorithm that computes a (minimal) finite complete set of E-unifiers for all E-unification problems over $\mathcal{F}$.
- E-unification algorithm yields a decision procedure for E.
- (Minimal) E-unification procedure: A procedure that enumerates a possible infinite (minimal) complete set of E-unifiers.
- E-unification procedure does not yield a decision procedure for E.

Decidability wrt Kinds

Decidability of an equational theory might depend on the kinds of E-unification.

- There exists an equational theory for which elementary unification is decidable, but unification with constants is undecidable:

H.-J. Bürckert.
Some relationships between unification, restricted unification, and matching.
In J. Siekmann, editor, *Proc. 8th Int. Conference on Automated Deduction*, volume 230 of *LNCS*. Springer, 1986.

Decidability wrt Kinds

Decidability of an equational theory might depend on the kinds of E-unification.

- There exists an equational theory for which unification with constants is decidable, but general unification is undecidable:

J. Otop.
E-unification with constants vs. general E-unification.
Journal of Automated Reasoning, 48(3):363–390, 2012.

Single Equation vs Systems of Equations

- In syntactic unification, solving systems of equations can be reduced to solving a single equation.
- For equational unification, the same holds only for general unification.
- For elementary unification and for unification with constants it is not the case.

Single Equation vs Systems of Equations

There exists an equational theory E such that

- all elementary E-unification problems of cardinality 1 (single equations) have minimal complete sets of E-unifiers, but
- E is of type zero wrt to elementary unification: There exists an elementary E-unification problem of cardinality > 1 that does not have a minimal complete set of unifiers.

H.-J. Bürckert, A. Herold, and M. Schmidt-Schauß.
On equational theories, unification, and decidability.
J. Symbolic Computation **8**(3,4), 3–49. 1989.

Single Equation vs Systems of Equations

There exists an equational theory E such that

- unifiability of elementary E-unification problems of cardinality 1 (single equations) is decidable, but
- for elementary problems of larger cardinality it is undecidable.

P. Narendran and H. Otto.
Some results on equational unification.
In M. E. Stickel, editor, *Proc. 10th Int. Conference on Automated Deduction*, volume 449 of *LNAI*. Springer, 1990.

Three Main Questions in Unification Theory

For a given E, unification theory is mainly concerned with finding answers to the following three questions:

Decidability: Is it decidable whether an E-unification problem is solvable? If yes, what is the complexity of this decision problem?

Unification type: What is the unification type of the theory E?

Unification algorithm: How can we obtain an (efficient) E-unification algorithm, or a (preferably minimal) E-unification procedure?

Three Main Questions in Unification Theory

- Decidability depends on
  - equational theory,
  - signature (kinds),
  - cardinality of unification problems.

Three Main Questions in Unification Theory

- Decidability depends on
  - equational theory,
  - signature (kinds),
  - cardinality of unification problems.
- Unification type depends on
  - equational theory,
  - signature (kinds),
  - cardinality of unification problems.

Summary of Results for Specific Theories

General unification:

Theory	Decidability	Type	Algorithm/Procedure
$\varnothing$, BR	Yes	1	Yes
A, AU	Yes	∞	Yes
C, AC, ACU	Yes	ω	Yes
I, CI, ACI	Yes	ω	Yes
AI	Yes	0	?
$\mathsf{D}_{\{f,g\}}\mathsf{A}_g$	No	∞	?
AG	Yes	ω	Yes
CRU	No	? (∞ or 0)	?

BR - Boolean ring, D - distributivity, CRU - commutative ring with unit.

Commutative Unification and Matching

- C-unification inference system $\mathcal{U}_{\mathsf{C}}$ can be obtained from the $\mathcal{U}$ by adding the C-Decomposition rule:

  C-Decomposition:

  $$\{f(s_1, s_2) \stackrel{?}{\doteq}_{\mathsf{C}} f(t_1, t_2)\} \uplus P'; S \Longrightarrow$$
  $$\{s_1 \stackrel{?}{\doteq}_{\mathsf{C}} t_2, s_2 \stackrel{?}{\doteq}_{\mathsf{C}} t_1\} \cup P'; S, \quad \text{if } f \text{ is commutative.}$$

- **C-Decomposition** and **Decomposition** transform the same system in different ways.

Commutative Unification and Matching

- C-unification inference system $\mathcal{U}_{\mathsf{C}}$ can be obtained from the $\mathcal{U}$ by adding the C-Decomposition rule:

  C-Decomposition:

  $$\{f(s_1, s_2) \stackrel{?}{\doteq}_{\mathsf{C}} f(t_1, t_2)\} \uplus P'; S \Longrightarrow$$
  $$\{s_1 \stackrel{?}{\doteq}_{\mathsf{C}} t_2, s_2 \stackrel{?}{\doteq}_{\mathsf{C}} t_1\} \cup P'; S, \quad \text{if } f \text{ is commutative.}$$

- **C-Decomposition** and **Decomposition** transform the same system in different ways.
- C-matching algorithm $\mathcal{M}_{\mathsf{C}}$ is obtained analogously from $\mathcal{M}$.

C-Unification

In order to C-unify s and t:

1. Create an initial system $\{s \doteq^{?}_{\mathsf{C}} t\}; \varnothing$.
2. Apply successively rules from $\mathcal{U}_{\mathsf{C}}$, building a complete tree of derivations. **C-Decomposition** and **Decomposition** rules have to be applied concurrently and form branching points in the derivation tree.

Example. C-Unification

C-unify $g(f(x,y),z)$ and $g(f(f(a,b),f(b,a)),c)$, commutative f.

$$\{g(f(x,y),z) \stackrel{?}{\doteq}_{\mathsf{C}} g(f(f(a,b),f(b,a))),c)\};\varnothing$$

Example. C-Unification

C-unify $g(f(x,y),z)$ and $g(f(f(a,b),f(b,a)),c)$, commutative f.

$$\{g(f(x,y),z) \doteq^{?}_{\mathsf{C}} g(f(f(a,b),f(b,a))),c)\};\varnothing$$
$$\downarrow$$
$$\{f(x,y) \doteq^{?}_{\mathsf{C}} f(f(a,b),f(b,a)), z \doteq^{?}_{\mathsf{C}} c\};\varnothing$$

Example. C-Unification

C-unify $g(f(x,y),z)$ and $g(f(f(a,b),f(b,a)),c)$, commutative f.

$$\{g(f(x,y),z) \doteq^?_{\mathsf{C}} g(f(f(a,b),f(b,a))),c)\};\varnothing$$

$$\downarrow$$

$$\{f(x,y) \doteq^?_{\mathsf{C}} f(f(a,b),f(b,a)), z \doteq^?_{\mathsf{C}} c\};\varnothing$$

$$\swarrow \qquad \searrow$$

$$\{x \doteq^?_{\mathsf{C}} f(a,b), y \doteq^?_{\mathsf{C}} f(b,a), z \doteq^?_{\mathsf{C}} c\};\varnothing \qquad \{x \doteq^?_{\mathsf{C}} f(b,a), y \doteq^?_{\mathsf{C}} f(a,b), z \doteq^?_{\mathsf{C}} c\};\varnothing$$

Example. C-Unification

C-unify $g(f(x,y),z)$ and $g(f(f(a,b),f(b,a)),c)$, commutative f.

$$\{g(f(x,y),z) \doteq^?_{\mathsf{C}} g(f(f(a,b),f(b,a))),c)\};\varnothing$$

$$\downarrow$$

$$\{f(x,y) \doteq^?_{\mathsf{C}} f(f(a,b),f(b,a)), z \doteq^?_{\mathsf{C}} c\};\varnothing$$

$$\swarrow \qquad\qquad \searrow$$

$$\{x \doteq^?_{\mathsf{C}} f(a,b), y \doteq^?_{\mathsf{C}} f(b,a), z \doteq^?_{\mathsf{C}} c\};\varnothing \qquad\qquad \{x \doteq^?_{\mathsf{C}} f(b,a), y \doteq^?_{\mathsf{C}} f(a,b), z \doteq^?_{\mathsf{C}} c\};\varnothing$$

$$\downarrow$$

$$\{y \doteq^?_{\mathsf{C}} f(b,a), z \doteq^?_{\mathsf{C}} c\};\{x \doteq f(a,b)\}$$

Example. C-Unification

C-unify $g(f(x,y),z)$ and $g(f(f(a,b),f(b,a)),c)$, commutative f.

$$\{g(f(x,y),z) \doteq^?_{\mathsf{C}} g(f(f(a,b),f(b,a))),c)\};\varnothing$$

$$\downarrow$$

$$\{f(x,y) \doteq^?_{\mathsf{C}} f(f(a,b),f(b,a)), z \doteq^?_{\mathsf{C}} c\};\varnothing$$

$$\swarrow \qquad\qquad \searrow$$

$$\{x \doteq^?_{\mathsf{C}} f(a,b), y \doteq^?_{\mathsf{C}} f(b,a), z \doteq^?_{\mathsf{C}} c\};\varnothing \qquad\qquad \{x \doteq^?_{\mathsf{C}} f(b,a), y \doteq^?_{\mathsf{C}} f(a,b), z \doteq^?_{\mathsf{C}} c\};\varnothing$$

$$\downarrow$$

$$\{y \doteq^?_{\mathsf{C}} f(b,a), z \doteq^?_{\mathsf{C}} c\};\{x \doteq f(a,b)\}$$

$$\downarrow$$

$$\{z \doteq^?_{\mathsf{C}} c\};\{x \doteq f(a,b), y \doteq f(b,a)\}$$

Example. C-Unification

C-unify $g(f(x,y),z)$ and $g(f(f(a,b),f(b,a)),c)$, commutative f.

$$\{g(f(x,y),z) \doteq^?_{\mathsf{C}} g(f(f(a,b),f(b,a))),c)\};\varnothing$$

$$\downarrow$$

$$\{f(x,y) \doteq^?_{\mathsf{C}} f(f(a,b),f(b,a)), z \doteq^?_{\mathsf{C}} c\};\varnothing$$

$$\swarrow \qquad \searrow$$

$$\{x \doteq^?_{\mathsf{C}} f(a,b), y \doteq^?_{\mathsf{C}} f(b,a), z \doteq^?_{\mathsf{C}} c\};\varnothing \qquad \{x \doteq^?_{\mathsf{C}} f(b,a), y \doteq^?_{\mathsf{C}} f(a,b), z \doteq^?_{\mathsf{C}} c\};\varnothing$$

$$\downarrow$$

$$\{y \doteq^?_{\mathsf{C}} f(b,a), z \doteq^?_{\mathsf{C}} c\};\{x \doteq f(a,b)\}$$

$$\downarrow$$

$$\{z \doteq^?_{\mathsf{C}} c\};\{x \doteq f(a,b), y \doteq f(b,a)\}$$

$$\downarrow$$

$$\varnothing;\{x \doteq f(a,b), y \doteq f(b,a), z \doteq c\}$$

Example. C-Unification

C-unify $g(f(x,y),z)$ and $g(f(f(a,b),f(b,a)),c)$, commutative f.

$$\{g(f(x,y),z) \doteq^?_{\mathsf{C}} g(f(f(a,b),f(b,a))),c)\};\varnothing$$

$$\downarrow$$

$$\{f(x,y) \doteq^?_{\mathsf{C}} f(f(a,b),f(b,a)), z \doteq^?_{\mathsf{C}} c\};\varnothing$$

$$\swarrow \qquad\qquad \searrow$$

$$\{x \doteq^?_{\mathsf{C}} f(a,b), y \doteq^?_{\mathsf{C}} f(b,a), z \doteq^?_{\mathsf{C}} c\};\varnothing \qquad\qquad \{x \doteq^?_{\mathsf{C}} f(b,a), y \doteq^?_{\mathsf{C}} f(a,b), z \doteq^?_{\mathsf{C}} c\};\varnothing$$

$$\downarrow \qquad\qquad\qquad\qquad \downarrow$$

$$\{y \doteq^?_{\mathsf{C}} f(b,a), z \doteq^?_{\mathsf{C}} c\};\{x \doteq f(a,b)\} \qquad\qquad \{y \doteq^?_{\mathsf{C}} f(a,b), z \doteq^?_{\mathsf{C}} c\};\{x \doteq f(b,a)\}$$

$$\downarrow$$

$$\{z \doteq^?_{\mathsf{C}} c\};\{x \doteq f(a,b), y \doteq f(b,a)\}$$

$$\downarrow$$

$$\varnothing;\{x \doteq f(a,b), y \doteq f(b,a), z \doteq c\}$$

Example. C-Unification

C-unify $g(f(x,y),z)$ and $g(f(f(a,b),f(b,a)),c)$, commutative f.

$$\{g(f(x,y),z) \doteq^?_{\mathsf{C}} g(f(f(a,b),f(b,a))),c)\};\varnothing$$

$$\downarrow$$

$$\{f(x,y) \doteq^?_{\mathsf{C}} f(f(a,b),f(b,a)), z \doteq^?_{\mathsf{C}} c\};\varnothing$$

$$\swarrow \qquad\qquad \searrow$$

$$\{x \doteq^?_{\mathsf{C}} f(a,b), y \doteq^?_{\mathsf{C}} f(b,a), z \doteq^?_{\mathsf{C}} c\};\varnothing \qquad \{x \doteq^?_{\mathsf{C}} f(b,a), y \doteq^?_{\mathsf{C}} f(a,b), z \doteq^?_{\mathsf{C}} c\};\varnothing$$

$$\downarrow \qquad\qquad \downarrow$$

$$\{y \doteq^?_{\mathsf{C}} f(b,a), z \doteq^?_{\mathsf{C}} c\};\{x \doteq f(a,b)\} \qquad \{y \doteq^?_{\mathsf{C}} f(a,b), z \doteq^?_{\mathsf{C}} c\};\{x \doteq f(b,a)\}$$

$$\downarrow \qquad\qquad \downarrow$$

$$\{z \doteq^?_{\mathsf{C}} c\};\{x \doteq f(a,b), y \doteq f(b,a)\} \qquad \{z \doteq^?_{\mathsf{C}} c\};\{x \doteq f(b,a), y \doteq f(a,b)\}$$

$$\downarrow$$

$$\varnothing;\{x \doteq f(a,b), y \doteq f(b,a), z \doteq c\}$$

Example. C-Unification

C-unify $g(f(x,y),z)$ and $g(f(f(a,b),f(b,a)),c)$, commutative f.

$$\{g(f(x,y),z) \doteq^?_{\mathsf{C}} g(f(f(a,b),f(b,a))),c)\};\varnothing$$

$$\downarrow$$

$$\{f(x,y) \doteq^?_{\mathsf{C}} f(f(a,b),f(b,a)), z \doteq^?_{\mathsf{C}} c\};\varnothing$$

$$\swarrow \qquad\qquad \searrow$$

$$\{x \doteq^?_{\mathsf{C}} f(a,b), y \doteq^?_{\mathsf{C}} f(b,a), z \doteq^?_{\mathsf{C}} c\};\varnothing \qquad \{x \doteq^?_{\mathsf{C}} f(b,a), y \doteq^?_{\mathsf{C}} f(a,b), z \doteq^?_{\mathsf{C}} c\};\varnothing$$

$$\downarrow \qquad\qquad \downarrow$$

$$\{y \doteq^?_{\mathsf{C}} f(b,a), z \doteq^?_{\mathsf{C}} c\};\{x \doteq f(a,b)\} \qquad \{y \doteq^?_{\mathsf{C}} f(a,b), z \doteq^?_{\mathsf{C}} c\};\{x \doteq f(b,a)\}$$

$$\downarrow \qquad\qquad \downarrow$$

$$\{z \doteq^?_{\mathsf{C}} c\};\{x \doteq f(a,b), y \doteq f(b,a)\} \qquad \{z \doteq^?_{\mathsf{C}} c\};\{x \doteq f(b,a), y \doteq f(a,b)\}$$

$$\downarrow \qquad\qquad \downarrow$$

$$\varnothing;\{x \doteq f(a,b), y \doteq f(b,a), z \doteq c\} \qquad \varnothing;\{x \doteq f(b,a), y \doteq f(a,b), z \doteq c\}$$

Example. C-Unification

C-unify $g(f(x,y),z)$ and $g(f(f(a,b),f(b,a)),c)$, commutative f.

$$\{g(f(x,y),z) \doteq^?_{\mathsf{C}} g(f(f(a,b),f(b,a))),c)\};\varnothing$$

$$\downarrow$$

$$\{f(x,y) \doteq^?_{\mathsf{C}} f(f(a,b),f(b,a)), z \doteq^?_{\mathsf{C}} c\};\varnothing$$

$$\swarrow \qquad\qquad \searrow$$

$$\{x \doteq^?_{\mathsf{C}} f(a,b), y \doteq^?_{\mathsf{C}} f(b,a), z \doteq^?_{\mathsf{C}} c\};\varnothing \qquad \{x \doteq^?_{\mathsf{C}} f(b,a), y \doteq^?_{\mathsf{C}} f(a,b), z \doteq^?_{\mathsf{C}} c\};\varnothing$$

$$\downarrow \qquad\qquad \downarrow$$

$$\{y \doteq^?_{\mathsf{C}} f(b,a), z \doteq^?_{\mathsf{C}} c\};\{x \doteq f(a,b)\} \qquad \{y \doteq^?_{\mathsf{C}} f(a,b), z \doteq^?_{\mathsf{C}} c\};\{x \doteq f(b,a)\}$$

$$\downarrow \qquad\qquad \downarrow$$

$$\{z \doteq^?_{\mathsf{C}} c\};\{x \doteq f(a,b), y \doteq f(b,a)\} \qquad \{z \doteq^?_{\mathsf{C}} c\};\{x \doteq f(b,a), y \doteq f(a,b)\}$$

$$\downarrow \qquad\qquad \downarrow$$

$$\varnothing;\{x \doteq f(a,b), y \doteq f(b,a), z \doteq c\} \qquad \varnothing;\{x \doteq f(b,a), y \doteq f(a,b), z \doteq c\}$$

Not minimal.

C-Unification: Termination, Soundness, Completeness

Theorem

For a C*-unification problem* P*, the* C*-unification algorithm terminates and computes a complete set of* C*-unifiers of* P*.*

MCSUs Can Be Large

Example

- Problem: $f(f(x_1, x_2), f(x_3, x_4)) \doteq^{?}_{\mathsf{C}} f(f(a, b), f(c, d))$.
- *mcsu* contains 4! substitutions.

C-Unification Algorithm Is Not Minimal

- The algorithm, in general, does not return a minimal complete set of C-unifiers.
- The obtained complete set can be further minimized, removing redundant unifiers.
- Not clear how to design a C-unification algorithm that computes a minimal complete set of unifiers directly.

Complexity of C-Unification and Matching

Theorem

The decision problem of C-matching and unification is NP-complete.

ACU-Unification

$$\text{ACU} = \{f(f(x,y),z) \approx f(x,f(y,z)), f(x,y) \approx f(y,x), f(x,e) \approx x\}$$

1. Associativity, commutativity, unit element.
2. f is associative and commutative, e is the unit element.

Example: Elementary ACU-Unification

Elementary ACU-unification problem:

$$\Gamma = \{f(x, f(x, y)) \stackrel{?}{\doteq}_{\mathsf{ACU}} f(z, f(z, z))\}$$

Solving idea:

1. Associate with the equation in Γ a homogeneous linear Diophantine equation $2x + y = 3z$.
2. The equation states that the number of new variables introduced by a unifier σ in both sides of $\Gamma\sigma$ must be the same.

(Continues on the next slide.)

Example. Elementary ACU-Unification (Cont.)

3. Solve $2x + y = 3z$ over nonnegative integers. Three minimal solutions:

$$x = 1,\ y = 1,\ z = 1$$
$$x = 0,\ y = 3,\ z = 1$$
$$x = 3,\ y = 0,\ z = 2$$

Any other solution of the equation can be obtained as a nonnegative linear combination of these three solutions.

(Continues on the next slide.)

Example. Elementary ACU-Unification (Cont.)

4. Introduce new variables v_1, v_2, v_3 for each solution of the Diophantine equation:

	x	y	z
v_1	1	1	1
v_2	0	3	1
v_3	3	0	2

5. Each row corresponds to a unifier of Γ:

$$\sigma_1 = \{x \mapsto v_1,\ y \mapsto v_1,\ z \mapsto v_1\}$$
$$\sigma_2 = \{x \mapsto e,\ y \mapsto f(v_2, f(v_2, v_2)),\ z \mapsto v_2\}$$
$$\sigma_3 = \{x \mapsto f(v_3, f(v_3, v_3)),\ y \mapsto e,\ z \mapsto f(v_3, v_3)\}$$

However, none of them is an mgu.

Example. Elementary ACU-Unification (Cont.)

6. To obtain an mgu, we should combine all three solutions:

	x	y	z
v_1	1	1	1
v_2	0	3	1
v_3	3	0	2

   The columns indicate that the mgu we are looking for should have
   - in the binding for x one v_1, zero v_2, and three v_3's,
   - in the binding for y one v_1, three v_2's, and zero v_3,
   - in the binding for z one v_1, one v_2, and two v_3's

7. Hence, we can construct an mgu:

$$\sigma = \{x \mapsto f(v_1, f(v_3, f(v_3, v_3))),\ y \mapsto f(v_1, f(v_2, f(v_2, v_2))),\ z \mapsto f(v_1, f(v_2, f(v_3, v_3)))\}$$

Example: ACU-Unification with constants

- ACU-unification problem with constants

$$\Gamma = \{f(x,f(x,y)) \stackrel{?}{\doteq}_{\mathsf{ACU}} f(a,f(z,f(z,z)))\}$$

reduces to inhomogeneous linear Diophantine equation

$$S = \{2x + y = 3z + 1\}.$$

- The minimal nontrivial natural solutions of S are $(0,1,0)$ and $(2,0,1)$.

Example: ACU-Unification with constants

- ACU-unification problem with constants

$$\Gamma = \{f(x, f(x, y)) \stackrel{?}{\doteq}_{\mathsf{ACU}} f(a, f(z, f(z, z)))\}$$

reduces to inhomogeneous linear Diophantine equation

$$S = \{2x + y = 3z + 1\}.$$

- Every natural solution of S is obtained as the sum of one of its minimal solutions and a solution of the corresponding homogeneous LDE $2x + y = 3z$.
- One element of the minimal complete set of unifiers of Γ is obtained from the combination of one minimal solution of S with the set of all minimal solutions of $2x + y = 3z$.

Example: ACU-Unification with constants

- ACU-unification problem with constants

$$\Gamma = \{f(x, f(x, y)) \stackrel{?}{\doteq}_{\mathsf{ACU}} f(a, f(z, f(z, z)))\}$$

reduces to inhomogeneous linear Diophantine equation

$$S = \{2x + y = 3z + 1\}.$$

- The minimal complete set of unifiers of Γ is $\{\sigma_1, \sigma_2\}$, where

$$\begin{aligned}
\sigma_1 = \{&x \mapsto f(v_1, f(v_3, f(v_3, v_3))),\\
&y \mapsto f(a, f(v_1, f(v_2, f(v_2, v_2)))),\\
&z \mapsto f(v_1, f(v_2, f(v_3, v_3)))\}\\
\sigma_2 = \{&x \mapsto f(a, f(a, f(v_1, f(v_3, f(v_3, v_3))))),\\
&y \mapsto f(v_1, f(v_2, f(v_2, v_2))),\\
&z \mapsto f(a, f(v_1, f(v_2, f(v_3, v_3))))\}
\end{aligned}$$

ACU-Unification with constants

- If an ACU-unification problem contains more than one constant, solve the corresponding inhomogeneous LDE for each constant.
- The unifiers in the minimal complete set correspond to all possible combinations of the minimal solutions of these inhomogeneous equations.

ACU-Unification with constants

Example

$xxy \doteq^{?}_{\mathsf{ACU}} aabbb$:

- Equation for a: $2x + y = 2$. Minimal solutions: $(1, 0)$ and $(0, 2)$.
- Corresponding unifiers: $\{x \mapsto a, y \mapsto e\}$, $\{x \mapsto e, y \mapsto aa\}$
- Equation for b: $2x + y = 3$. Minimal solutions: $(0, 3)$ and $(1, 1)$.
- Corresponding unifiers: $\{x \mapsto e, y \mapsto bbb\}$, $\{x \mapsto b, y \mapsto b\}$
- Unifiers in the minimal complete set: $\{x \mapsto a, y \mapsto bbb\}$, $\{x \mapsto ab, y \mapsto b\}$, $\{x \mapsto e, y \mapsto aabbb\}$, $\{x \mapsto b, y \mapsto aab\}$.

From ACU to AC

Example

- How to solve $\Gamma_1 = \{f(x, f(x, y)) \stackrel{?}{\doteq}_{\mathsf{AC}} f(z, f(z, z))\}$?
- We "know" how to solve $\Gamma_2 = \{f(x, f(x, y)) \stackrel{?}{\doteq}_{\mathsf{ACU}} f(z, f(z, z))\}$, but its mgu is not an mgu for Γ_1.
- Mgu of Γ_2:

$$\sigma = \{x \mapsto f(v_1, f(v_3, f(v_3, v_3))),\ y \mapsto f(v_1, f(v_2, f(v_2, v_2))),\ z \mapsto f(v_1, f(v_2, f(v_3, v_3)))\}$$

- Unifier of Γ_1: $\vartheta = \{x \mapsto v_1, y \mapsto v_1, z \mapsto v_1\}$.
- σ is not more general modulo AC than ϑ.

From ACU to AC

Example

- Idea: Take the mgu of Γ_2.
- Compose it with all possible erasing substitutions that map a subset of $\{v_1, v_2, v_3\}$ to the unit element.
- Restriction: The result of the composition should not map x, y, and z to the unit element.

From ACU to AC

Example

Minimal complete set of unifiers for Γ_1:

$$\sigma_1 = \{x \mapsto f(v_1, f(v_3, f(v_3, v_3))),\ y \mapsto f(v_1, f(v_2, f(v_2, v_2))),\ z \mapsto f(v_1, f(v_2, f(v_3, v_3)))\}$$

$$\sigma_2 = \{x \mapsto f(v_3, f(v_3, v_3)),\ y \mapsto f(v_2, f(v_2, v_2)),\ z \mapsto f(v_2, f(v_3, v_3))\}$$

$$\sigma_3 = \{x \mapsto f(v_1, f(v_3, f(v_3, v_3))),\ y \mapsto v_1,\ z \mapsto f(v_1, f(v_3, v_3))\}$$

$$\sigma_4 = \{x \mapsto v_1,\ y \mapsto f(v_1, f(v_2, f(v_2, v_2))),\ z \mapsto f(v_1, v_2)\}$$

$$\sigma_5 = \{x \mapsto v_1,\ y \mapsto v_1,\ z \mapsto v_1\}$$

How to Solve Systems of LDEs over Naturals?

Contejean-Devie Algorithm:

Evelyne Contejean and Hervé Devie.
An Efficient Incremental Algorithm for Solving Systems of Linear Diophantine Equations.
Information and Computation 113(1): 143–172 (1994).

How to Solve Systems of LDEs over Naturals?

Contejean-Devie Algorithm:

Evelyne Contejean and Hervé Devie.
An Efficient Incremental Algorithm for Solving Systems of Linear Diophantine Equations.
Information and Computation 113(1): 143–172 (1994).

Generalizes Fortenbacher's Algorithm for solving a single equation:

Michael Clausen and Albrecht Fortenbacher.
Efficient Solution of Linear Diophantine Equations.
J. Symbolic Computation 8(1,2): 201–216 (1989).

Example. E-Unification of Type 0

Example

- Equational theory: $E = \{f(e,x) \approx x,\ g(f(x,y)) \approx g(y)\}$.
- E-unification problem: $\Gamma = \{g(x) \stackrel{?}{\doteq}_E g(e)\}$.

Example. E-Unification of Type 0

Example

- Equational theory: $E = \{f(e,x) \approx x,\ g(f(x,y)) \approx g(y)\}$.
- E-unification problem: $\Gamma = \{g(x) \stackrel{?}{\doteq}_E g(e)\}$.
- Complete (why?) set of solutions:

$$\begin{aligned}
\sigma_0 &= \{x \mapsto e\} \\
\sigma_1 &= \{x \mapsto f(x_0, e)\} \\
\sigma_2 &= \{x \mapsto f(x_1, f(x_0, e))\} \\
&\dots \\
\sigma_n &= \{x \mapsto f(x_{n-1}, x\sigma_{n-1})\} \\
&\dots
\end{aligned}$$

Example. E-Unification of Type 0

Example

- Equational theory: $E = \{f(e,x) \approx x,\ g(f(x,y)) \approx g(y)\}$.
- E-unification problem: $\Gamma = \{g(x) \stackrel{?}{\doteq}_E g(e)\}$.
- Complete (why?) set of solutions:

$$\begin{aligned}
\sigma_0 &= \{x \mapsto e\} \\
\sigma_1 &= \{x \mapsto f(x_0, e)\} \\
\sigma_2 &= \{x \mapsto f(x_1, f(x_0, e))\} \\
&\dots \\
\sigma_n &= \{x \mapsto f(x_{n-1}, x\sigma_{n-1})\} \\
&\dots
\end{aligned}$$

- No *mcsu*. $\sigma_i \doteq_E^{\{x\}} \sigma_{i+1}\{x_i \mapsto e\}$. $\sigma_i \not\leq_E^{\{x\}} \sigma_j$ for $i > j$.
  Infinite descending chain: $\sigma_0 >_E^{\{x\}} \sigma_1 >_E^{\{x\}} \sigma_2 >_E^{\{x\}} \cdots$

Example. E-Unification of Type 0

Example (Cont.)

Why does $\sigma_0 \gtrdot_E^{\{x\}} \sigma_1 \gtrdot_E^{\{x\}} \sigma_2 \gtrdot_E^{\{x\}} \cdots$ imply that there is no $mcsu$?

- Let $S = \{\sigma_0, \sigma_1, \ldots\}$.
- Let S' be an arbitrary complete set of unifiers of Γ.
- Since S is complete, for any $\vartheta \in S'$ there exists $\sigma_i \in S$ such that $\sigma_i \lesssim_E^{\{x\}} \vartheta$.
- Since $\sigma_{i+1} \lessdot_E^{\{x\}} \sigma_i$, we get $\sigma_{i+1} \lessdot_E^{\{x\}} \vartheta$.
- On the other hand, since S' is complete, there exists $\eta \in S'$ such that $\eta \lesssim_E^{\{x\}} \sigma_{i+1}$.
- Hence, $\eta \lessdot_E^{\{x\}} \vartheta$ which implies that S' is not minimal.

Specific vs General Results

For each specific equational theory separately studying

- decidability,
- unification type,
- unification algorithm/procedure.

Can one study these problems for bigger classes of equational theories?

Specific vs General Results

For each specific equational theory separately studying

- decidability,
- unification type,
- unification algorithm/procedure.

Can one study these problems for bigger classes of equational theories?

General Results

In general, unification modulo equational theories

- is undecidable,
- unification type of a given theory is undecidable,
- admits a complete unification procedure (Gallier & Snyder, called an universal E-unification procedure).

General Results: Universal E-Unification Procedure

Universal E-unification procedure $\mathcal{U}_E$.

Rules:

- **Trivial**, **Orient**, **Decomposition**, **Variable Elimination** from $\mathcal{U}$, plus
- **Lazy Paramodulation:**

$$\{e[u]\} \cup P'; S \Longrightarrow \{l \stackrel{?}{\doteq} u,\ e[r]\} \cup P'; S,$$

for a fresh variant of the identity $l \approx r$ from $E \cup E^{-1}$, where

- $e[u]$ is an equation where the term u occurs,
- u is not a variable,
- if l is not a variable, then the top symbol of l and u are the same.

Universal E-Unification Procedure. Control

In order to solve a unification problem Γ modulo a given E:

- Create an initial system $\Gamma; \varnothing$.
- Apply successively rules from $\mathcal{U}_E$, building a complete tree of derivations.
- No other inference rule may be applied to the equation $l \stackrel{?}{\doteq} u$ that is generated by the Lazy Paramodulation rule before it is subjected to a Decomposition step.

Universal E-Unification Procedure. Example

$E = \{f(a,b) \approx a,\ a \approx b\}$.

Unification problem: $\{f(x,x) \doteq^?_E x\}$.

Computing a unifier $\{x \mapsto a\}$ by the universal procedure:

$$
\begin{aligned}
\{f(x,x) \doteq^?_E x\};\varnothing &\Longrightarrow_{LP} \{f(a,b) \doteq^?_E f(x,x), a \doteq^?_E x\};\varnothing \\
&\Longrightarrow_{D} \{a \doteq^?_E x,\ b \doteq^?_E x,\ a \doteq^?_E x\};\varnothing \\
&\Longrightarrow_{O} \{x \doteq^?_E a,\ b \doteq^?_E x,\ a \doteq^?_E x\};\varnothing \\
&\Longrightarrow_{S} \{b \doteq^?_E a,\ a \doteq^?_E a\};\{x \doteq a\} \\
&\Longrightarrow_{LP} \{a \doteq^?_E a,\ b \doteq^?_E b,\ a \doteq^?_E a\};\{x \doteq a\} \\
&\Longrightarrow^{+}_{T} \varnothing;\{x \doteq a\}
\end{aligned}
$$

Universal E-Unification Procedure: Assessment

Pros and cons of the universal procedure:

- Pros: Is sound and complete. Can be used for any E.
- Cons: Very inefficient. Usually does not yield a decision procedure or a (minimal) E-unification algorithm even for unitary or finitary theories with decidable unification.

General Results

More useful results can be obtained by imposing additional restrictions on equational theories:

- Syntactic approaches: Restricting syntactic form of the identities defining equational theories.
- Semantic approaches: Depend on properties of the free algebras defined by the equational theory.