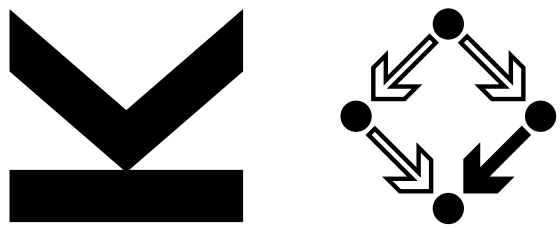
Solving Equational Problems

Unification and Matching

Temur Kutsia

RISC, Johannes Kepler University Linz



ISR 2017, Eindhoven

Overview

Part 1

Syntactic unification and matching

Part 2

Equational unification and matching

Overview

Part 1

Syntactic unification and matching

Part 2

Equational unification and matching

Rewriting Requires Matching

Rewrite $s(0) + s(s(0))$ by the rule $s(x) + y \to s(x + y)$.

Match $s(x) + y$ to $s(0) + s(s(0))$.

Completion Requires Unification

Compute a critical pair between $\mathrm{f}(\mathrm{f}(\mathrm{x}, \mathrm{y}), z) \to \mathrm{f}(\mathrm{x}, \mathrm{f}(\mathrm{y}, z))$ and $\mathrm{h}(\mathrm{f}(\mathrm{x}_1, \mathrm{y}_1)) \to \mathrm{f}(\mathrm{h}(\mathrm{x}_1), \mathrm{h}(\mathrm{y}_1))$.

Unify $\mathrm{f}(\mathrm{f}(\mathrm{x}, \mathrm{y}), z)$ and $\mathrm{f}(\mathrm{x}_1, \mathrm{y}_1)$.

Rewriting Modulo Equalities Requires E-Matching

Rewrite $\mathrm{f}(\mathrm{a}, \mathrm{f}(\mathrm{b}, e))$ by the rule $\mathrm{f}(e, \mathrm{x}) \to \mathrm{x}$.

f is commutative.

Match $\mathrm{f}(e, \mathrm{x})$ to $\mathrm{f}(\mathrm{b}, e)$ modulo commutativity of f.

Completion Modulo Equalities Requires E-Unification

Compute a critical pair between $x \circ (x^{-} \circ z) \to 1$ and $(y_0 \circ x_0)^{-} \to (x_0)^{-} \circ (y_0)^{-}$

$\circ$ is associative and commutative.

AC-unify $x \circ (x^{-} \circ z)$ and $y_0 \circ x_0$.

Solving Term Equations

Unification/matching problems: problems of solving equations between terms.

Used in

- rewriting
- automated reasoning
- logic and functional programming
- type inference
- program transformation
- computational linguistics
- ...

Subject of this Course

Syntactic unification and matching.

Equational unification and matching.

Notation

First-order language.

$\mathcal{F}$: Set of function symbols.

$\mathcal{V}$: Set of variables.

$\mathfrak{x}, \mathfrak{y}, \mathfrak{z}$: Variables.

$\mathfrak{a}, \mathfrak{b}, \mathfrak{c}$: Constants.

$\mathfrak{f}, \mathfrak{g}, \mathfrak{h}$: Arbitrary function symbols.

Notation

$\mathrm{s}, \mathrm{t}, \mathrm{r}$: Terms.

$\mathcal{T}(\mathcal{F}, \mathcal{V})$: Set of terms over $\mathcal{F}$ and $\mathcal{V}$.

Equation: a pair of terms, written $s \doteq \mathrm{t}$.

$\mathcal{V}ars(\mathrm{t})$: 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.

A substitution is represented as a set of bindings, e.g.,

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

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.

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

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

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

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

Substitution Domain, Range, Variable Range

For a substitution σ:

- The domain is the set of variables:

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

- The range is the set of terms:

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

- The variable range is the set of variables:

$$\mathrm{vran}(\sigma) = \mathrm{vars}(\mathrm{ran}(\sigma)).$$

Substitution Domain, Range, Variable Range

Example

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

$$\mathrm{dom}(\varepsilon) = \mathrm{ran}(\varepsilon) = \mathrm{vran}(\varepsilon) = \emptyset$$

Substitution Restriction

Restriction of a substitution σ on a set of variables $\mathcal{X}$:
A substitution $\sigma|_{\mathcal{X}}$ such that for all x

$$x\sigma|_{\mathcal{X}} = \begin{cases} x\sigma & \text{if } x \in \mathcal{X}. \\ x & \text{otherwise.} \end{cases}$$

Example

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

Composition of Substitutions

Composition of σ and ϑ, written $\sigma\vartheta$.

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

Constructing the set representation of the composition $\sigma\vartheta$ (σ and ϑ are given by their set representation):

1. Apply ϑ to every term in $\mathrm{ran}(\sigma)$ and remove any trivial binding $x \mapsto x$ from the obtained set to get σ_1.
2. Remove from ϑ any binding $x \mapsto t$ with $x \in \mathrm{dom}(\sigma)$ to get ϑ_1.
3. Take the union of the sets of bindings σ_1 and ϑ_1.

Composition of Substitutions

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 of Substitutions

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:

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

Variable Renaming Substitutions

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

- y's are distinct variables, and
- $\{x_1, \ldots, x_n\} = \{y_1, \ldots, y_n\}$.

(Permuting the domain variables.)

Example

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

Idempotent Substitutions

σ is called idempotent if $\sigma\sigma = \sigma$.

Theorem

σ is idempotent iff $\mathrm{dom}(\sigma) \cap \mathcal{vran}(\sigma) = \emptyset$.

Instantiation Quasi-Ordering

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

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

$\simeq$ is the equivalence relation corresponding to $\lesssim$.

Instantiation Quasi-Ordering

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

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

$\simeq$ is the equivalence relation corresponding to $\lesssim$.

Example

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

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

Substitution Equivalence and Renaming

Theorem

For any σ and ϑ, $\sigma \simeq \vartheta$ iff there exists a variable renaming substitution η such that $\sigma\eta = \vartheta$.

Example

σ, ϑ from the previous example:

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

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 $\sigma \lesssim \vartheta$ for every unifier ϑ of s and t.

A unification problem for s and t is represented as $s \doteq^{?} t$.

Most General Unifier: Example

Unification problem: $x \doteq^{?} y$.

Most General Unifier: Example

Unification problem: $x \doteq^{?} y$.

Mgus:

$\{x \mapsto y\}$

Most General Unifier: Example

Unification problem: $x \doteq^{?} y$.

Mgus:

$\{x \mapsto y\}$

$\{y \mapsto x\}$

Most General Unifier: Example

Unification problem: $x \doteq^{?} y$.

Mgus:

$\{x \mapsto y\}$

$\{y \mapsto x\}$

Any other?

Unifier, Most General Unifier: Example

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

Unifier, Most General Unifier: Example

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

Some of the unifiers:

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

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

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

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

$\ldots$

Unifier, Most General Unifier: Example

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

Some of the unifiers:

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

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

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

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

$\ldots$

Idempotent mgus: $\{x \mapsto y,\ z \mapsto g(a)\}, \{y \mapsto x,\ z \mapsto g(a)\}$.

Mgus are equivalent:

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

Idempotent Most General Unifiers

Theorem

Let σ be an idempotent unifier for a unification problem. Then ϑ is an unifier the same problem iff $\vartheta = \sigma\vartheta$.

Proof.

$(\Rightarrow)$ ϑ is an unifier. Then $\vartheta = \sigma\varphi$ for some φ. Since σ is idempotent, $\vartheta = \sigma\sigma\varphi = \sigma\vartheta$.

$(\Leftarrow)$ Follows from the definition of mgu. $\square$

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.

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.

Approach:

- Rule-base formulation.
- Repeated transformation of a set of equations.

Solved Form

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.

Solved Form

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.
- $\vec{S}$ for the idempotent substitution corresponding to a solved form set S.

System

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

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

$\bot$ represents failure.

System

System: The symbol $\bot$ or a pair $\mathrm{P}; S$ 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 $\mathrm{P}; S$:
A substitution that unifies each of the equations in P and S.

$\bot$ has no unifiers.

System: Example

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

Its unifier: $\{x \mapsto g(a),\ y \mapsto a,\ z \mapsto g(g(a))\}$.

Unification Rules

Trivial:

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

Decomposition:

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

Symbol Clash:

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

Unification Rules

Orient:

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

Occurrence Check:

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

Elimination:

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

if $x \notin vars(t)$.

The Algorithm

The algorithm UNIFY:

In order to unify s and t:

1. Create an initial system $\{s \doteq^? t\}; \emptyset$.
2. Apply successively the transformation rules.
3. If the final system is $\emptyset; S$, return $\vec{S}$.
4. If the final system is $\bot$, then fail.

This is essentially the Herbrand's Unification Algorithm.

Examples

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

$$\{p(f(a), g(x)) \stackrel{\cdot}{=}^{?} p(y, y)\};\ \emptyset \Longrightarrow \quad \text{(by Decomposition)}$$

$$\{f(a) \stackrel{\cdot}{=}^{?} y,\ g(x) \stackrel{\cdot}{=}^{?} y\};\ \emptyset \Longrightarrow \quad \text{(by Orient)}$$

$$\{y \stackrel{\cdot}{=}^{?} f(a),\ g(x) \stackrel{\cdot}{=}^{?} y\};\ \emptyset \Longrightarrow \quad \text{(by Elimination)}$$

$$\{g(x) \stackrel{\cdot}{=}^{?} f(a)\};\ \{y \stackrel{\cdot}{=} f(a)\} \Longrightarrow \quad \text{(by Symbol Clash)}$$

$$\bot \Longrightarrow$$

Examples

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

$$\{p(a, x, h(g(z))) \stackrel{\cdot}{=}^{?} p(z, h(y), h(y))\};\ \emptyset \Longrightarrow \quad \text{(Dec.)}$$

$$\{a \stackrel{\cdot}{=}^{?} z,\ x \stackrel{\cdot}{=}^{?} h(y),\ h(g(z)) \stackrel{\cdot}{=}^{?} h(y)\};\ \emptyset \Longrightarrow \quad \text{(Or.)}$$

$$\{z \stackrel{\cdot}{=}^{?} a,\ x \stackrel{\cdot}{=}^{?} h(y),\ h(g(z)) \stackrel{\cdot}{=}^{?} h(y)\};\ \emptyset \Longrightarrow \quad \text{(Elim.)}$$

$$\{x \stackrel{\cdot}{=}^{?} h(y),\ h(g(a)) \stackrel{\cdot}{=}^{?} h(y)\};\ \{z \stackrel{\cdot}{=} a\} \Longrightarrow \quad \text{(Elim.)}$$

$$\{h(g(a)) \stackrel{\cdot}{=}^{?} h(y)\};\ \{z \stackrel{\cdot}{=} a,\ x \stackrel{\cdot}{=} h(y)\} \Longrightarrow \quad \text{(Dec.)}$$

$$\{g(a) \stackrel{\cdot}{=}^{?} y\};\ \{z \stackrel{\cdot}{=} a,\ x \stackrel{\cdot}{=} h(y)\} \Longrightarrow \quad \text{(Or.)}$$

$$\{y \stackrel{\cdot}{=}^{?} g(a)\};\ \{z \stackrel{\cdot}{=} a,\ x \stackrel{\cdot}{=} h(y)\} \Longrightarrow \quad \text{(Elim.)}$$

$$\emptyset;\ \{z \stackrel{\cdot}{=} a,\ x \stackrel{\cdot}{=} h(g(a)),\ y \stackrel{\cdot}{=} g(a)\}$$

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

Examples

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

$$\{p(x,x) \stackrel{\cdot}{=}^{?} p(y, f(y))\};\ \emptyset \Longrightarrow \quad \text{(by Decomposition)}$$

$$\{x \stackrel{\cdot}{=}^{?} y, x \stackrel{\cdot}{=}^{?} f(y)\};\ \emptyset \Longrightarrow \quad \text{(by Elimination)}$$

$$\{y \stackrel{\cdot}{=}^{?} f(y)\};\ \{x \stackrel{\cdot}{=} y\} \Longrightarrow \quad \text{(By Occ. Check)}$$

$$\bot$$

Properties of UNIFY: Termination

Lemma

For any finite set of equations P, every sequence of transformations

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

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

Properties of UNIFY: Termination

Proof.

Complexity measure on the set P of equations: $\langle \mathrm{n}_1, \mathrm{n}_2, \mathrm{n}_3 \rangle$, ordered lexicographically on triples of naturals, where

$\mathrm{n}_1 =$ The number of distinct variables in P.

$\mathrm{n}_2 =$ The number of symbols in P.

$\mathrm{n}_3 =$ The number of equations in P of the form $\mathrm{t} \stackrel{\cdot}{=}^{?} \mathrm{x}$ where t is not a variable.

Properties of UNIFY: Termination

Proof [Cont.]

Each rule strictly reduces the complexity measure.

Rule	n_1	n_2	n_3
Trivial	$\geqslant$	$>$	
Decomposition	$=$	$>$	
Orient	$=$	$=$	$>$
Elimination	$>$		

Properties of UNIFY: Termination

Proof [Cont.]

A rule can always be applied to a system with $\mathrm{P} \neq \emptyset$.

Rules do not apply only to $\bot$ and $\emptyset;\ S$.

Whenever an equation is added to S, the variable on the left-hand side is eliminated from the rest of the system, i.e. $S_1, S_2, \ldots$ are in solved form. □

Properties of UNIFY: Termination

Proof [Cont.]

A rule can always be applied to a system with $\mathrm{P} \neq \emptyset$.

Rules do not apply only to $\bot$ and $\emptyset;\ S$.

Whenever an equation is added to S, the variable on the left-hand side is eliminated from the rest of the system, i.e. $S_1, S_2, \ldots$ are in solved form. □

Corollary

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

Properties of UNIFY: Correctness

Notation: Γ for systems.

Lemma

For any transformation $\mathrm{P}; S \Longrightarrow \Gamma$, a substitution ϑ unifies $\mathrm{P}; S$ iff it unifies Γ.

Properties of UNIFY: Correctness

Proof.

Occurrence Check: If $x \in vars(t)$ and $x \neq t$, then

- x contains fewer symbols than t,
- $x\vartheta$ contains fewer symbols than $t\vartheta$ (for any ϑ).

Therefore, $x\vartheta$ and $t\vartheta$ can not be unified.

Elimination: From $x\vartheta = t\vartheta$, by structural induction on u:
$u\vartheta = u\{x \mapsto t\}\vartheta$ for any term, equation, or set of equations u.
Then

$$P'\vartheta = P'\{x \mapsto t\}\vartheta, \qquad S'\vartheta = S'\{x \mapsto t\}\vartheta.$$

□

Properties of UNIFY: Correctness

Theorem (Soundness)

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

Properties of UNIFY: Correctness

Theorem (Soundness)

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

Proof.

By induction on the length of derivation, using the previous lemma and the fact that $\vec{S}$ unifies S. □

Properties of UNIFY: Correctness

Theorem (Completeness)

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

Properties of UNIFY: Correctness

Theorem (Completeness)

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

Proof.

Such a sequence must end in $\emptyset;\ S$ where ϑ unifies S (why?).

For every binding $x \mapsto t$ in $\vec{S}$, $x\vec{S}\vartheta = t\vartheta = x\vartheta$ and for every $x \notin \mathrm{dom}(\vec{S})$, $x\vec{S}\vartheta = x\vartheta$. Hence, $\vartheta = \vec{S}\vartheta$. □

Properties of UNIFY: Correctness

Theorem (Completeness)

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

Proof.

Such a sequence must end in $\emptyset;\ S$ where ϑ unifies S (why?).

For every binding $x \mapsto t$ in $\vec{S}$, $x\vec{S}\vartheta = t\vartheta = x\vartheta$ and for every $x \notin \mathrm{dom}(\vec{S})$, $x\vec{S}\vartheta = x\vartheta$. Hence, $\vartheta = \vec{S}\vartheta$. □

Corollary

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

Observations

The algorithm computes an idempotent mgu.

The choice of rules in computations 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.

Observations

Any practical algorithm that proceeds by performing transformations of UNIFY in any order is sound, complete, and generates mgus for unifiable terms.

Not all transformation sequences have the same length.

Not all transformation sequences end in exactly the same mgu.

Matching

A substitution σ is a matcher of s to t if $s\sigma = t$.

A matching problem between s and t is represented as $s \ll^? t$.

Example: Matching vs Unification

$f(x, y) \ll^{?} f(g(z), c)$	$f(x, y) \doteq^{?} f(g(z), c)$
$\{x \mapsto g(z), y \mapsto c\}$	$\{x \mapsto g(z), y \mapsto c\}$

Example: Matching vs Unification

$f(x,y) \ll^{?} f(g(z),c)$	$f(x,y) \doteq^{?} f(g(z),c)$
$\{x \mapsto g(z), y \mapsto c\}$	$\{x \mapsto g(z), y \mapsto c\}$
$f(x,y) \ll^{?} f(g(z),x)$	$f(x,y) \doteq^{?} f(g(z),x)$
$\{x \mapsto g(z), y \mapsto x\}$	$\{x \mapsto g(z), y \mapsto g(z)\}$

Example: Matching vs Unification

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

Example: Matching vs Unification

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

Example: Matching vs Unification

$f(x,y) \ll^? f(g(z),c)$	$f(x,y) \doteq^? f(g(z),c)$
$\{x \mapsto g(z), y \mapsto c\}$	$\{x \mapsto g(z), y \mapsto c\}$
$f(x,y) \ll^? f(g(z),x)$	$f(x,y) \doteq^? f(g(z),x)$
$\{x \mapsto g(z), y \mapsto x\}$	$\{x \mapsto g(z), y \mapsto g(z)\}$
$f(x,a) \ll^? f(b,y)$	$f(x,a) \doteq^? f(b,y)$
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 unification rules 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 form, if all x's are pairwise distinct.

The notation $\vec{S}$ extends to matched forms.

If S is in matched form, then

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

Matching System

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

- P is set of matching problems.
- S is set of equations in matched form.

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

$\bot$ has no matchers.

Matching Rules

Decomposition:

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

Symbol Clash:

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

Symbol-Variable Clash:

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

Matching Rules

Merging Clash:

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

Elimination:

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

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

The Algorithm

The Algorithm MATCH:

In order to match s to t,

1. Create an initial system $\{s \ll^? t\}; \emptyset$.
2. Apply successively the matching rules.
3. If the final system is $\emptyset; S$, return $\vec{S}$.
4. If the final system is $\bot$, then fail.

Examples

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

$$\{f(x, f(a, x)) \ll^? f(g(a),\ f(a, g(a)))\};\ \emptyset \Longrightarrow \quad \text{(Dec.)}$$

$$\{x \ll^? g(a),\ f(a, x) \ll^? f(a, g(a))\};\ \emptyset \Longrightarrow \quad \text{(Elim.)}$$

$$\{f(a, x) \ll^? f(a, g(a))\};\ \{x \ll g(a)\} \Longrightarrow \quad \text{(Dec.)}$$

$$\{a \ll^? a, x \ll^? g(a)\};\{x \ll g(a)\} \Longrightarrow \quad \text{(Dec.)}$$

$$\{x \ll^? g(a)\};\ \{x \ll g(a)\} \Longrightarrow \quad \text{(Elim.)}$$

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

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

Examples

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

$$\{f(x, x) \ll^? f(x, a)\}; \emptyset \Longrightarrow \quad \text{(Decomposition)}$$
$$\{x \ll^? x, x \ll^? a\}; \emptyset \Longrightarrow \quad \text{(Elimination)}$$
$$\{x \ll^? a\}; \{x \ll x\} \Longrightarrow \quad \text{(Merging Clash)}$$
$$\bot$$

No matcher.

Properties of MATCH: Termination

Theorem

For any finite set of matching problems P, every sequence of matching transformations $\mathrm{P}; \emptyset \Longrightarrow \mathrm{P}_1; \mathrm{S}_1 \Longrightarrow \mathrm{P}_2; \mathrm{S}_2 \Longrightarrow \cdots$ terminates either with $\bot$ or with $\emptyset; \mathrm{S}$, with S in matched form.

Properties of MATCH: Termination

Theorem

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

Proof.

Termination: Every rule strictly decreases the size of P's.

A rule can always be applied to a system with $P \neq \emptyset$.

Rules do not apply only to $\bot$ and $\emptyset; S$.

Whenever $x \ll t$ is added to S, there is no other equation $x \ll t'$ in S. Hence, $S_1, S_2, \ldots$ are in matched form. □

Properties of MATCH: Correctness

Lemma

For any transformation of matching systems $\mathrm{P}; S \Longrightarrow \Gamma$, a substitution ϑ is a matcher for $\mathrm{P}; S$ iff it is a matcher for Γ.

Properties of MATCH: Correctness

Theorem (Soundness)

If $\mathrm{P}; \emptyset \Longrightarrow^{+} \emptyset; \mathrm{S}$, then $\vec{S}$ solves all matching equations in P.

Properties of MATCH: Correctness

Theorem (Soundness)

If $\mathrm{P}; \emptyset \Longrightarrow^{+} \emptyset; \mathrm{S}$, then $\vec{S}$ solves all matching equations in P.

Proof.

By induction on the length of derivations, using the previous lemma and the fact that $\vec{S}$ solves the matching equations in S. □

Properties of MATCH: Correctness

Let $\nu(\{s_1 \ll t_1, \ldots, s_n \ll t_n\})$ be $vars(\{s_1, \ldots, s_n\})$.

Theorem (Completeness)

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

Properties of MATCH: Correctness

Let $\nu(\{s_1 \ll t_1, \ldots, s_n \ll t_n\})$ be $vars(\{s_1, \ldots, s_n\})$.

Theorem (Completeness)

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

Proof.

Such a sequence must end in $\emptyset; S$ where ϑ is a matcher of S.

$\nu(S) = \nu(P)$.

For each $x \ll t \in S$, either $t = x$ or $x \mapsto t \in \vec{S}$.

Therefore, for any such x, $x\vec{S} = t = x\vartheta$.

Hence, $\vec{S} = \vartheta|_{\nu(P)}$. □

Properties of MATCH: Correctness

Corollary

If P has no matchers, then any maximal sequence of matching transformations from $\mathrm{P}; \emptyset$ must end with $\perp$.

Overview

Part 1

Syntactic unification and matching

Part 2

Equational unification and matching

Motivation

Equational matching and unification algorithms are used in

- rewriting and completion modulo equalities,
- automated reasoning,
- programming,
- verification,
- ...

Motivation

Equational unification is a dual problem for the word problem.

$\mathtt{E}$: A given set of equalities.

Word problem:

Does $\forall \overline{x}.\ s \doteq t$ hold in all models of $\mathtt{E}$?

Equational unification:

Does $\exists \overline{x}.\ s \doteq t$ hold in all nonempty models of $\mathtt{E}$?

Motivation

Equational unification generalizes syntactic unification.

$\mathrm{f}(\mathrm{x}, \mathrm{y}) \doteq^{?} \mathrm{f}(\mathrm{a}, \mathrm{b})$ has only one mgu $\{\mathrm{x} \mapsto \mathrm{a}, \mathrm{y} \mapsto \mathrm{b}\}$, if it is a syntactic unification problem.

If f is commutative, then $\{\mathrm{x} \mapsto \mathrm{b}, \mathrm{y} \mapsto \mathrm{a}\}$ is another unifier.

Equational Theory

$\mathtt{E}$: a set of equations over $\mathcal{T}(\mathcal{F}, \mathcal{V})$, called identities.

Equational theory $\doteq_{\mathtt{E}}$ defined by $\mathtt{E}$: The least congruence relation on $\mathcal{T}(\mathcal{F}, \mathcal{V})$ stable under substitution application and containing $\mathtt{E}$.

Equational Theory

$\mathtt{E}$: a set of equations over $\mathcal{T}(\mathcal{F}, \mathcal{V})$, called identities.

Equational theory $\doteq_{\mathtt{E}}$ defined by $\mathtt{E}$: The least congruence relation on $\mathcal{T}(\mathcal{F}, \mathcal{V})$ stable under substitution application and containing $\mathtt{E}$.

I. e., $\doteq_{\mathtt{E}}$ is the least binary relation on $\mathcal{T}(\mathcal{F}, \mathcal{V})$ such that:

- $\mathtt{E} \subseteq \ \doteq_{\mathtt{E}}$.
- Reflexivity: $s \doteq_{\mathtt{E}} s$ for all s.
- Symmetry: If $s \doteq_{\mathtt{E}} t$ then $t \doteq_{\mathtt{E}} s$ for all s, t.
- Transitivity: If $s \doteq_{\mathtt{E}} t$ and $t \doteq_{\mathtt{E}} r$ then $s \doteq_{\mathtt{E}} r$ for all s, t, r.
- Congruence: If $s_1 \doteq_{\mathtt{E}} t_1, \ldots, s_n \doteq_{\mathtt{E}} t_n$ then $f(s_1, \ldots, s_n) \doteq_{\mathtt{E}} f(t_1, \ldots, t_n)$ for all s, t, n and n-ary f.
- Stability: If $s \doteq_{\mathtt{E}} t$ then $s\sigma \doteq_{\mathtt{E}} t\sigma$ for all s, t, σ.

Notation, Terminology

$s \doteq_E t$:

- The pair (s, t) belongs to the equational theory $\doteq_E$.
- The term s is equal modulo E to the term t.

$s \approx t$: Identities.

$sig(E)$: The set of function symbols that occur in E.

Sometimes E is called an equational theory as well.

Notation, Terminology: 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.

$sig(AU) = \{f,e\}$

$f(a,f(x,f(e,a))) \doteq_{AU} f(f(a,x),a)$.

E-Unification Problem, E-Unifier

E: a given set of identities.

E-Unification problem over $\mathcal{F}$: a finite set of equations

$$\Gamma = \{s_1 \stackrel{\cdot}{=}^?_{\mathsf{E}} t_1, \ldots, s_n \stackrel{\cdot}{=}^?_{\mathsf{E}} t_n\},$$

where $s_i, t_i \in \mathcal{T}(\mathcal{F}, \mathcal{V})$.

E-Unifier of Γ: a substitution σ such that

$$s_1\sigma \stackrel{\cdot}{=}_{\mathsf{E}} t_1\sigma, \ldots, s_n\sigma \stackrel{\cdot}{=}_{\mathsf{E}} t_n\sigma.$$

$\mathbb{E}$-Unifiability

$\mathbb{E}$: a given set of identities.

$\mathfrak{u}_{\mathbb{E}}(\Gamma)$: the set of $\mathbb{E}$-unifiers of an $\mathbb{E}$-unification problem Γ.

Γ is $\mathbb{E}$-unifiable iff $\mathfrak{u}_{\mathbb{E}}(\Gamma) \neq \emptyset$.

$\mathbb{E}$-Unification vs Syntactic Unification

Syntactic unification: a special case of $\mathbb{E}$-unification with $\mathbb{E} = \emptyset$.

Any syntactic unifier of an $\mathbb{E}$-unification problem Γ is also an $\mathbb{E}$-unifier of Γ.

For $\mathbb{E} \neq \emptyset$, $\mathfrak{u}_{\mathbb{E}}(\Gamma)$ may contain a unifier that is not a syntactic unifier.

E-Unification vs Syntactic Unification: Example

Terms $\mathtt{f}(\mathtt{a}, \mathtt{x})$ and $\mathtt{f}(\mathtt{b}, \mathtt{y})$:

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

E-Unification vs Syntactic Unification: 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 there are additional unifiers, e.g., $\{x \mapsto f(z, b), y \mapsto f(a, z)\}$.

Instantiation Quasi-Ordering (Modified)

$\mathbb{E}$: equational theory. $\mathcal{X}$: set of variables.

A substitution σ is more general than ϑ modulo $\mathbb{E}$ on $\mathcal{X}$, written $\sigma \lesssim^{\mathcal{X}}_{\mathbb{E}} \vartheta$, if there exists η such that $x\sigma\eta \doteq_{\mathbb{E}} x\vartheta$ for all $x \in \mathcal{X}$.

ϑ is called an $\mathbb{E}$-instance of σ modulo $\mathbb{E}$ on $\mathcal{X}$.

Instantiation Quasi-Ordering (Modified)

E: equational theory. $\mathcal{X}$: set of variables.

A substitution σ is more general than ϑ modulo E on $\mathcal{X}$, written $\sigma \lesssim_{\mathsf{E}}^{\mathcal{X}} \vartheta$, if there exists η such that $x\sigma\eta \doteq_{\mathsf{E}} x\vartheta$ for all $x \in \mathcal{X}$.

ϑ is called an E-instance of σ modulo E on $\mathcal{X}$.

$\lesssim_{\mathsf{E}}^{\mathcal{X}}$: instantiation quasi-ordering.

$\simeq_{\mathsf{E}}^{\mathcal{X}}$: the equivalence relation corresponding to $\lesssim_{\mathsf{E}}^{\mathcal{X}}$.

No MGU

When comparing unifiers of Γ, the set $\mathcal{X}$ is $\mathcal{vars}(\Gamma)$.

Unifiable $\mathbb{E}$-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 $\mathcal{vars}(\Gamma) = \{x, y\}$, any unifier is equal to either σ_1 or σ_2.

σ_1 and σ_2 are not comparable wrt $\lesssim_C^{\{x,y\}}$.

Hence, no mgu for Γ.

Complete Sets of Unifiers

In $\mathbb{E}$-unification, the role of mgu is taken on by a complete set of $\mathbb{E}$-unifiers.

Γ: $\mathbb{E}$-unification problem over $\mathcal{F}$.

$\mathcal{X} = \mathcal{V}\mathrm{ars}(\Gamma)$.

$\mathcal{C}$ is a complete set of $\mathbb{E}$-unifiers of Γ iff

- $\mathcal{C} \subseteq \mathcal{U}_{\mathbb{E}}(\Gamma)$: $\mathcal{C}$'s elements are $\mathbb{E}$-unifiers of Γ, and
- For each $\vartheta \in \mathcal{U}_{\mathbb{E}}(\Gamma)$ there exists $\sigma \in \mathcal{C}$ such that $\sigma \lesssim_{\mathbb{E}}^{\mathcal{X}} \vartheta$.

Minimal Complete Sets of Unifiers

$\mathcal{C}$ is a minimal complete set of $\mathtt{E}$-unifiers ($\mathrm{mcsu}_{\mathtt{E}}$) of Γ if

- it is a complete set of $\mathtt{E}$-unifiers of Γ and
- Two distinct elements of $\mathcal{C}$ are not comparable wrt $\lesssim^{x}_{\mathtt{E}}$.

σ is an mgu of Γ iff $\mathrm{mcsu}_{\mathtt{E}}(\Gamma) = \{\sigma\}$.

Minimal Complete Sets of Unifiers

$\mathfrak{mcsu}_{\mathrm{E}}(\Gamma) = \emptyset$ 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 $\simeq^{\mathcal{X}}_{\mathrm{E}}$.

Unification Type of a Problem

$\mathtt{E}$: equational theory.

Γ: $\mathtt{E}$-unification problem over $\mathcal{F}$.

Γ has unification type

- unitary, if $\mathrm{mcsu}(\Gamma)$ has cardinality at most one,
- finitary, if $\mathrm{mcsu}(\Gamma)$ has finite cardinality,
- infinitary, if $\mathrm{mcsu}(\Gamma)$ has infinite cardinality,
- zero, if $\mathrm{mcsu}(\Gamma)$ does not exist.

Unification Type of a Problem

$\mathbb{E}$: equational theory.

Γ: $\mathbb{E}$-unification problem over $\mathcal{F}$.

Γ has unification type

- unitary, if $\mathrm{mcsu}(\Gamma)$ has cardinality at most one,
- finitary, if $\mathrm{mcsu}(\Gamma)$ has finite cardinality,
- infinitary, if $\mathrm{mcsu}(\Gamma)$ has infinite cardinality,
- zero, if $\mathrm{mcsu}(\Gamma)$ does not exist.

Abbreviation: unitary - 1, finitary - ω, infinitary - ∞, zero - 0.

Ordering: $1 < \omega < \infty < 0$.

Unification Type of a Problem, Theory

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

- on $\mathbb{E}$, and
- on $\mathcal{F}$ (which function symbols are permitted in unification problems).

Unification type of a theory $\mathbb{E}$ wrt $\mathcal{F}$: the maximal type of an $\mathbb{E}$-unification problem over $\mathcal{F}$.

Example of Unitary Unification Type

Syntactic unification.

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

Example of Finitary Unification Type

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

- Not unitary.
- $\{f(x, y) \stackrel{?}{\doteq}_C f(a, b)\}$ has two unifiers $\{x \mapsto a, y \mapsto b\}$ and $\{x \mapsto b, y \mapsto a\}$.
- No mgu.
- C unification is finitary.

Example of Finitary Unification Type

C unification is finitary for any $\mathcal{F}$:

- Let $\Gamma = \{s_1 \stackrel{?}{\doteq}_{\mathsf{C}} t_1, \ldots, s_n \stackrel{?}{\doteq}_{\mathsf{C}} t_n\}$ 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 $s'_i \doteq_{\mathsf{C}} s_i$ and $t'_i \doteq_{\mathsf{C}} t_i$ for each $1 \leqslant i \leqslant n$.
- There are only finitely many such Γ's, because the C-equivalence class for a given term t is finite.
- The collection of all mgus 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.

Example of Infinitary Unification Type

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

- $\{f(x,a) \doteq^{?}_{A} f(a,x)\}$ 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.

  Machine Intelligence, volume 7, pages 73–90. Edinburgh University Press, 1972.
- A-unification is infinitary for any $\mathcal{F}$.

Example of Unification of 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

Unification Type depends on $\mathcal{F}$.

Example

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 is unitary.
- There are ACU problems containing function symbols other than f and e, which are finitary, not unitary.
- For instance, $\mathrm{mcsu}(\{f(x,y) \doteq^{?}_{\mathsf{ACU}} f(a,b)\})$ consists of four unifiers (which ones?).

Kinds of E-unification.

Kinds of $\mathtt{E}$-Unification

Khree kinds of $\mathtt{E}$-unification problems, depending on the function symbols that are allowed to occur in them.

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

Unification Types of Theories wrt Kinds

Unification Types Depending on the Signature

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

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.

Unification Types wrt of Cardinality of Problems

There exists an equational theory $\mathbb{E}$ such that

- all elementary $\mathbb{E}$-unification problems of cardinality 1 (single equations) have minimal complete sets of $\mathbb{E}$-unifiers, but
- $\mathbb{E}$ is of type zero wrt to elementary unification: There exists an elementary $\mathbb{E}$-unification problem of cardinality 2 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.

Decision and Unification Procedures

Decision procedure for an equational theory $\mathbb{E}$ wrt $\mathcal{F}$:
An algorithm that for each $\mathbb{E}$-unification problem Γ over $\mathcal{F}$ returns $success$ if Γ is $\mathbb{E}$-unifiable, and $failure$ otherwise.

Decision and Unification Procedures

Decision procedure for an equational theory $\mathbb{E}$ wrt $\mathcal{F}$:
An algorithm that for each $\mathbb{E}$-unification problem Γ over $\mathcal{F}$ returns $success$ if Γ is $\mathbb{E}$-unifiable, and $failure$ otherwise.

$\mathbb{E}$ is decidable if it admits a decision procedure.

Decision and Unification Procedures

(Minimal) $\mathbb{E}$-unification algorithm wrt $\mathcal{F}$:

- An algorithm that computes a (minimal) finite complete set of $\mathbb{E}$-unifiers for all $\mathbb{E}$-unification problems over $\mathcal{F}$.

Decision and Unification Procedures

(Minimal) $\mathbb{E}$-unification algorithm wrt $\mathcal{F}$:

- An algorithm that computes a (minimal) finite complete set of $\mathbb{E}$-unifiers for all $\mathbb{E}$-unification problems over $\mathcal{F}$.
- Yields a decision procedure for $\mathbb{E}$.

Decision and Unification Procedures

(Minimal) $\mathbb{E}$-unification algorithm wrt $\mathcal{F}$:

- An algorithm that computes a (minimal) finite complete set of $\mathbb{E}$-unifiers for all $\mathbb{E}$-unification problems over $\mathcal{F}$.
- Yields a decision procedure for $\mathbb{E}$.

(Minimal) $\mathbb{E}$-unification procedure wrt $\mathcal{F}$:

- A procedure that enumerates a possible infinite (minimal) complete set of $\mathbb{E}$-unifiers for all $\mathbb{E}$-unification problems over $\mathcal{F}$.

Decision and Unification Procedures

(Minimal) $\mathbb{E}$-unification algorithm wrt $\mathcal{F}$:

- An algorithm that computes a (minimal) finite complete set of $\mathbb{E}$-unifiers for all $\mathbb{E}$-unification problems over $\mathcal{F}$.
- Yields a decision procedure for $\mathbb{E}$.

(Minimal) $\mathbb{E}$-unification procedure wrt $\mathcal{F}$:

- A procedure that enumerates a possible infinite (minimal) complete set of $\mathbb{E}$-unifiers for all $\mathbb{E}$-unification problems over $\mathcal{F}$.
- Does not yield a decision procedure for $\mathbb{E}$.

Decidability wrt Kinds

Decidability of an equational theory might depend on the kinds of $\mathbb{E}$-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 $\mathbb{E}$-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.

Decidability wrt Problem Cardinality

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.

Summary

Unification type depends on

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

Summary

Unification type depends on

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

Decidability depends on

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

Three Main Questions in Unification Theory

Decidability:

Is it decidable whether an $\mathtt{E}$-unification problem is solvable? If yes, what is the complexity of this decision problem?

Unification type:

What is the unification type of the theory $\mathtt{E}$?

Unification algorithm:

How can we obtain an (efficient) $\mathtt{E}$-unification algorithm, or a (preferably minimal) $\mathtt{E}$-unification procedure?

Summary of Results for Specific Theories

General unification:

Theory	Decidability	Type	Algorithm/Procedure
$\emptyset$, BR	Yes	1	Yes
A, AU	Yes	∞	Yes
C, AC, ACU	Yes	ω	Yes
I, CI, ACI	Yes	ω	Yes
AI	Yes	0	?
$D_{\{f,g\}}A_g$	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 algorithm C-UNIFY can be obtained from the UNIFY by adding the C-Decomposition rule:

C-Decomposition:

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

C-Decomposition and Decomposition transform the same system in different ways, causing branching in the derivation tree.

Commutative Unification and Matching

C-unification algorithm C-UNIFY can be obtained from the UNIFY by adding the C-Decomposition rule:

C-Decomposition:

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

C-Decomposition and Decomposition transform the same system in different ways, causing branching in the derivation tree.

C-matching algorithm is obtained analogously from syntactic matching.

Example. C-Unification

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

$$\{g(f(x,y),z) \stackrel{\cdot}{=}^{?}_{C} g(f(f(a,b),f(b,a))),c)\};\emptyset$$

Example. C-Unification

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

$$\{g(f(x,y),z) \stackrel{\cdot}{=}^{?}_{C} g(f(f(a,b),f(b,a))),c)\};\emptyset$$
$$\downarrow$$
$$\{f(x,y) \stackrel{\cdot}{=}^{?}_{C} f(f(a,b),f(b,a)), z \stackrel{\cdot}{=}^{?}_{C} c\};\emptyset$$

Example. C-Unification

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

$$\{g(f(x,y),z) \stackrel{\cdot}{=}^{?}_{C} g(f(f(a,b),f(b,a))),c)\};\emptyset$$

$$\downarrow$$

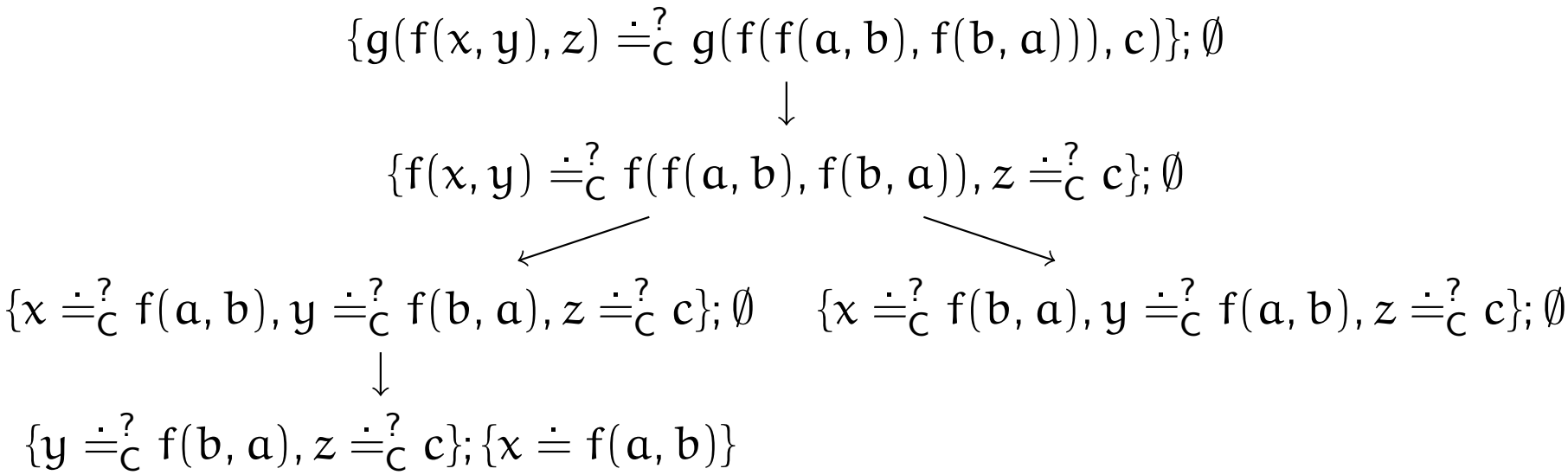
$$\{f(x,y) \stackrel{\cdot}{=}^{?}_{C} f(f(a,b),f(b,a)), z \stackrel{\cdot}{=}^{?}_{C} c\};\emptyset$$

$$\swarrow \qquad \searrow$$

$$\{x \stackrel{\cdot}{=}^{?}_{C} f(a,b), y \stackrel{\cdot}{=}^{?}_{C} f(b,a), z \stackrel{\cdot}{=}^{?}_{C} c\};\emptyset \qquad \{x \stackrel{\cdot}{=}^{?}_{C} f(b,a), y \stackrel{\cdot}{=}^{?}_{C} f(a,b), z \stackrel{\cdot}{=}^{?}_{C} c\};\emptyset$$

Example. C-Unification

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



Example. C-Unification

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

$$\{g(f(x,y),z) \doteq^?_C g(f(f(a,b),f(b,a))),c)\};\emptyset$$

$$\downarrow$$

$$\{f(x,y) \doteq^?_C f(f(a,b),f(b,a)), z \doteq^?_C c\};\emptyset$$

$$\swarrow \qquad \searrow$$

$$\{x \doteq^?_C f(a,b), y \doteq^?_C f(b,a), z \doteq^?_C c\};\emptyset \qquad \{x \doteq^?_C f(b,a), y \doteq^?_C f(a,b), z \doteq^?_C c\};\emptyset$$

$$\downarrow$$

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

$$\downarrow$$

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

Example. C-Unification

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

$$\{g(f(x,y),z) \doteq^?_C g(f(f(a,b),f(b,a))),c)\};\emptyset$$

$$\downarrow$$

$$\{f(x,y) \doteq^?_C f(f(a,b),f(b,a)), z \doteq^?_C c\};\emptyset$$

$$\swarrow \qquad \searrow$$

$$\{x \doteq^?_C f(a,b), y \doteq^?_C f(b,a), z \doteq^?_C c\};\emptyset \qquad \{x \doteq^?_C f(b,a), y \doteq^?_C f(a,b), z \doteq^?_C c\};\emptyset$$

$$\downarrow$$

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

$$\downarrow$$

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

$$\downarrow$$

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

Example. C-Unification

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

$$\{g(f(x,y),z) \doteq^?_C g(f(f(a,b),f(b,a))),c)\};\emptyset$$

$$\downarrow$$

$$\{f(x,y) \doteq^?_C f(f(a,b),f(b,a)), z \doteq^?_C c\};\emptyset$$

$$\swarrow \qquad \searrow$$

$$\{x \doteq^?_C f(a,b), y \doteq^?_C f(b,a), z \doteq^?_C c\};\emptyset \qquad \{x \doteq^?_C f(b,a), y \doteq^?_C f(a,b), z \doteq^?_C c\};\emptyset$$

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

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

$$\downarrow$$

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

$$\downarrow$$

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

Example. C-Unification

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

$$\{g(f(x,y),z) \doteq^?_C g(f(f(a,b),f(b,a))),c)\};\emptyset$$

$$\downarrow$$

$$\{f(x,y) \doteq^?_C f(f(a,b),f(b,a)), z \doteq^?_C c\};\emptyset$$

$$\swarrow \qquad \searrow$$

$$\{x \doteq^?_C f(a,b), y \doteq^?_C f(b,a), z \doteq^?_C c\};\emptyset \qquad \{x \doteq^?_C f(b,a), y \doteq^?_C f(a,b), z \doteq^?_C c\};\emptyset$$

$$\downarrow \qquad \downarrow$$

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

$$\downarrow \qquad \downarrow$$

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

$$\downarrow$$

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

Example. C-Unification

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

$$\{g(f(x,y),z) \doteq^?_C g(f(f(a,b),f(b,a))),c)\};\emptyset$$

$$\downarrow$$

$$\{f(x,y) \doteq^?_C f(f(a,b),f(b,a)), z \doteq^?_C c\};\emptyset$$

$$\swarrow \qquad \searrow$$

$$\{x \doteq^?_C f(a,b), y \doteq^?_C f(b,a), z \doteq^?_C c\};\emptyset \qquad \{x \doteq^?_C f(b,a), y \doteq^?_C f(a,b), z \doteq^?_C c\};\emptyset$$

$$\downarrow \qquad \downarrow$$

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

$$\downarrow \qquad \downarrow$$

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

$$\downarrow \qquad \downarrow$$

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

Example. C-Unification

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

$$\{g(f(x,y),z) \doteq^?_C g(f(f(a,b),f(b,a))),c)\};\emptyset$$

$$\downarrow$$

$$\{f(x,y) \doteq^?_C f(f(a,b),f(b,a)), z \doteq^?_C c\};\emptyset$$

$$\swarrow \qquad \searrow$$

$$\{x \doteq^?_C f(a,b), y \doteq^?_C f(b,a), z \doteq^?_C c\};\emptyset \qquad \{x \doteq^?_C f(b,a), y \doteq^?_C f(a,b), z \doteq^?_C c\};\emptyset$$

$$\downarrow \qquad \downarrow$$

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

$$\downarrow \qquad \downarrow$$

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

$$\downarrow \qquad \downarrow$$

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

Not minimal.

Properties of the C-Unification Algorithm

Theorem

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

Properties of the C-Unification Algorithm

Theorem

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

Proof.

Termination is proved using the same measure as for syntactic unification.

Completeness is based on the following two facts:

- If Γ is transformed by only one rule into Γ', then $\mathfrak{u}_{\mathsf{C}}(\Gamma) = \mathfrak{u}_{\mathsf{C}}(\Gamma')$.
- If Γ is transformed by two rules into Γ_1 and Γ_2, then $\mathfrak{u}_{\mathsf{C}}(\Gamma) = \mathfrak{u}_{\mathsf{C}}(\Gamma_1) \cup \mathfrak{u}_{\mathsf{C}}(\Gamma_2)$.

MCSU for C-Unification Problems Can Be Large

Problem: $\mathrm{f}(\mathrm{f}(x_1, x_2), \mathrm{f}(x_3, x_4)) \stackrel{?}{\doteq}_{\mathsf{C}} \mathrm{f}(\mathrm{f}(\mathrm{a}, \mathrm{b}), \mathrm{f}(\mathrm{c}, \mathrm{d}))$.

mcsu contains 4! substitutions.

MCSU for C-Unification Problems Can Be Large

Problem: $f(f(x_1, x_2), f(x_3, x_4)) \doteq^?_C f(f(a, b), f(c, d))$.

mcsu contains 4! substitutions.

The problem is, actually, a matching problem.

Properties of the C-Unification Algorithm

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.

Properties of the C-Unification Algorithm

Theorem

The decision problem of both C-matching and unification is NP-complete, if there are at least two constants in the alphabet.

Proof

Idea for hardness: reduction from positive 1-in-3 SAT.

$S = \{C_1, \ldots, C_n\}$: an instance of positive 1-in-3 SAT.

x_p: a variable for each propositional variable p from S.

y_i: a variable for each clause $C_i \in S$.

a and b: constants, corresponding to true and false.

Properties of the C-Unification Algorithm

Theorem

The decision problem of both C-matching and unification is NP-complete, if there are at least two constants in the alphabet.

Proof [Cont.]

Translating a clause $C_i = p_i \vee q_i \vee r_i$:

$$g(g(g(x_{p_i}, x_{q_i}), x_{r_i}), y_i) \stackrel{?}{=}_C g(g(g(b, b), a), g(g(b, a), b)).$$

Unifier/evaluation correspondence:

$\{x_{r_i} \mapsto a, x_{p_i} \mapsto b, x_{q_i} \mapsto b, \ldots\}$ (r_i true, p_i false, q_i false).
$\{x_{q_i} \mapsto a, x_{p_i} \mapsto b, x_{r_i} \mapsto b, \ldots\}$ (q_i true, p_i false, r_i false).
$\{x_{p_i} \mapsto a, x_{q_i} \mapsto b, x_{r_i} \mapsto b, \ldots\}$ (p_i true, q_i false, r_i false).

ACU-Unification

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

Associativity, commutativity, unit element.

f is associative and commutative, e is the unit element.

Example: Elementary ACU-Unification

Elementary ACU-unification problem:

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

Example: Elementary ACU-Unification

Elementary ACU-unification problem:

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

Solving idea:

1. Associate with the equation in Γ a homogeneous linear Diophantine equation $2x + y = 3z$.
2. It 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 ν_1, ν_2, ν_3 for each solution of the Diophantine equation:

	x	y	z
ν_1	1	1	1
ν_2	0	3	1
ν_3	3	0	2

5. Each row corresponds to a unifier of Γ:

$$\sigma_1 = \{x \mapsto \nu_1,\ y \mapsto \nu_1,\ z \mapsto \nu_1\}$$
$$\sigma_2 = \{x \mapsto e,\ y \mapsto f(\nu_2, f(\nu_2, \nu_2)),\ z \mapsto \nu_2\}$$
$$\sigma_3 = \{x \mapsto f(\nu_3, f(\nu_3, \nu_3)),\ y \mapsto e,\ z \mapsto f(\nu_3, \nu_3)\}$$

However, none of them is an mgu.

Example. Elementary ACU-Unification (Fin.)

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

	x	y	z
ν_1	1	1	1
ν_2	0	3	1
ν_3	3	0	2

The columns indicate that the mgu should have

- in the binding for x one ν_1, zero ν_2, and three ν_3's,
- in the binding for y one ν_1, three ν_2's, and zero ν_3,
- in the binding for z one ν_1, one ν_2, and two ν_3's

Example. Elementary ACU-Unification (Fin.)

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 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:

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

Example: ACU-Unification with Constants

ACU-unification problem with constants

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

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

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

reduces to inhomogeneous linear Diophantine equation

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

Example: ACU-Unification with Constants

ACU-unification problem with constants

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

reduces to inhomogeneous linear Diophantine equation

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

Every natural solution of S is obtained as the sum of one of the minimal solution of S and a solution of the corresponding homogeneous LDE $2x + y = 3z$.

Example: ACU-Unification with Constants

ACU-unification problem with constants

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

reduces to inhomogeneous linear Diophantine equation

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

Every natural solution of S is obtained as the sum of one of the minimal solution of S 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

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

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 More than One Constant

$xxy \stackrel{?}{\doteq}_{\mathsf{ACU}} aabbb.$

Equation for a: $2x + y = 2$.

- Minimal solutions: $(1, 0)$ and $(0, 2)$.
- Corresponding unifiers: $\{x \mapsto a, y \mapsto e\}$, $\{x \mapsto e, y \mapsto aa\}$.

ACU-Unification with More than One Constant

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

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: $\{x \mapsto e, y \mapsto bbb\}$, $\{x \mapsto b, y \mapsto b\}$.

ACU-Unification with More than One Constant

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

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: $\{x \mapsto e, y \mapsto bbb\}$, $\{x \mapsto b, y \mapsto b\}$.

($2x + y = 0$ has only trivial solutions.) mcsu:

- $\{x \mapsto a, y \mapsto bbb\}$, $\{x \mapsto ab, y \mapsto b\}$,
  $\{x \mapsto e, y \mapsto aabbb\}$, $\{x \mapsto b, y \mapsto aab\}$.

From ACU to AC

How to solve $\Gamma_1 = \{f(x, f(x, y)) \stackrel{?}{\doteq}_{AC} f(z, f(z, z))\}$?

We "know" how to solve $\Gamma_2 = \{f(x, f(x, y)) \stackrel{?}{\doteq}_{ACU} f(z, f(z, z))\}$, but its mgu is not an mgu for Γ_1.

Mgu of Γ_2:

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

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

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

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).

Homogeneous Case

Homogeneous linear Diophantine system with m equations and n variables:

$$\begin{cases} a_{11}x_1 & +\cdots+ & a_{1n}x_n & = & 0 \\ \vdots & & \vdots & & \vdots \\ a_{m1}x_1 & +\cdots+ & a_{mn}x_n & = & 0 \end{cases}$$

- a_{ij}'s are integers.
- Looking for nontrivial natural solutions.

Homogeneous Case: Example

$$\begin{cases} -x_1 + x_2 + 2x_3 - 3x_4 = 0 \\ -x_1 + 3x_2 - 2x_3 - x_4 = 0 \end{cases}$$

Nontrivial solutions:

- $s_1 = (0, 1, 1, 1)$
- $s_2 = (4, 2, 1, 0)$
- $s_3 = (0, 2, 2, 2)$
- $s_4 = (8, 4, 2, 0)$
- $s_5 = (4, 3, 2, 1)$
- $s_6 = (8, 5, 3, 1)$
- $\ldots$

Homogeneous Case: Example

$$\begin{cases} -x_1 + x_2 + 2x_3 - 3x_4 = 0 \\ -x_1 + 3x_2 - 2x_3 - x_4 = 0 \end{cases}$$

Nontrivial solutions:

- $s_1 = (0, 1, 1, 1)$
- $s_2 = (4, 2, 1, 0)$
- $s_3 = (0, 2, 2, 2) = 2s_1$
- $s_4 = (8, 4, 2, 0) = 2s_2$
- $s_5 = (4, 3, 2, 1) = s_1 + s_2$
- $s_6 = (8, 5, 3, 1) = s_1 + 2s_2$
- $\ldots$

Homogeneous Case

Homogeneous linear Diophantine system with m equations and n variables:

$$\begin{cases} a_{11}x_1 & +\cdots+ & a_{1n}x_n & = & 0 \\ \vdots & & \vdots & & \vdots \\ a_{m1}x_1 & +\cdots+ & a_{mn}x_n & = & 0 \end{cases}$$

- a_{ij}'s are integers.
- Looking for a basis in the set of nontrivial natural solutions.

Homogeneous Case

Homogeneous linear Diophantine system with m equations and n variables:

$$\begin{cases} a_{11}x_1 & +\cdots+ & a_{1n}x_n & = & 0 \\ \vdots & & \vdots & & \vdots \\ a_{m1}x_1 & +\cdots+ & a_{mn}x_n & = & 0 \end{cases}$$

- a_{ij}'s are integers.
- Looking for a basis in the set of nontrivial natural solutions.
- Does it exist?

Homogeneous Case

The basis in the set S of nontrivial natural solutions of a homogeneous LDS is the set of $\gg$-minimal elements S.

$\gg$ is the ordering on tuples of natural numbers:

$$(x_1, \ldots, x_n) \gg (y_1, \ldots, y_n)$$

if and only if

- $x_i \geqslant y_i$ for all $1 \leqslant i \leqslant n$ and
- $x_i > y_i$ for some $1 \leqslant i \leqslant n$.

Jump to the example

Matrix Form

Homogeneous linear Diophantine system with m equations and n variables:

$$A x\downarrow = 0\downarrow,$$

where

$$A := \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix} \quad x\downarrow := \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \quad 0\downarrow := \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$$

Matrix Form

Canonical basis in $\mathbb{N}^n$: $(e_1\downarrow, \ldots, e_n\downarrow)$.

$$e_j\downarrow = \begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix}, \text{ with } 1 \text{ in } j\text{'s row.}$$

Then $Ax\downarrow = x_1 Ae_1\downarrow + \cdots + x_n Ae_n\downarrow$.

Matrix Form

a: The linear mapping associated to A.

$$a(x\downarrow) = \begin{pmatrix} a_{11}x_1 & +\cdots+ & a_{1n}x_n \\ \vdots & & \vdots \\ a_{m1}x_1 & +\cdots+ & a_{mn}x_n \end{pmatrix}$$

$$= \quad x_1 a(e_1\downarrow) + \cdots + x_n a(e_n\downarrow).$$

Single Equation: Idea

Case $m = 1$: Single homogeneous LDE $a_1x_1 + \cdots + a_nx_n = 0$.

Fortenbacher's idea:

- Search minimal solutions starting from the elements in the canonical basis of $\mathbb{N}^n$.
- Suppose the current vector $v\downarrow$ is not a solution.
- It can be nondeterministically increased, component by component, until it becomes a solution or greater than a solution.
- To decrease the search space, the following restrictions can be imposed:
  - If $a(v\downarrow) > 0$, then increase by one some v_j with $a_j < 0$.
  - If $a(v\downarrow) < 0$, then increase by one some v_j with $a_j > 0$.

Single Equation: Idea

Case $m = 1$: Single homogeneous LDE $a_1x_1 + \cdots + a_nx_n = 0$.

Fortenbacher's idea:

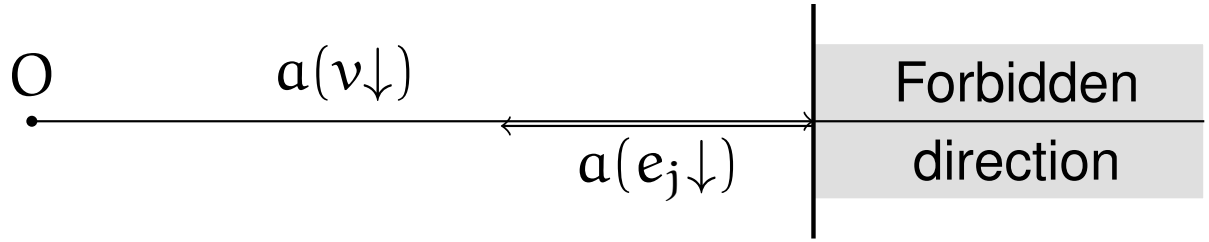
- Search minimal solutions starting from the elements in the canonical basis of $\mathbb{N}^n$.
- Suppose the current vector $v\downarrow$ is not a solution.
- It can be nondeterministically increased, component by component, until it becomes a solution or greater than a solution.
- To decrease the search space, the following restrictions can be imposed:
  - If $a(v\downarrow) > 0$, then increase by one some v_j with $a_j < 0$.
  - If $a(v\downarrow) < 0$, then increase by one some v_j with $a_j > 0$.
  - (If $a(v\downarrow)a(e_j\downarrow) < 0$ for some j, increase v_j by one.)

Single Equation: Geometric Interpretation of the Idea

Fortenbacher's condition: If $a(v\downarrow)a(e_j\downarrow) < 0$ for some j, increase v_j by one.

Increasing v_j by one: $a(v\downarrow + e_j\downarrow) = a(v\downarrow) + a(e_j\downarrow)$.

Going to the "right direction", towards the origin.



Single Equation: Algorithm

Case $m = 1$: Single homogeneous LDE $a_1x_1 + \cdots + a_nx_n = 0$.

Fortenbacher's algorithm:

- Start with the pair P, M of the set of potential solutions $P = \{e_1\downarrow, \ldots, e_n\downarrow\}$ and the set of minimal nontrivial solutions $M = \emptyset$.

Single Equation: Algorithm

Case $m = 1$: Single homogeneous LDE $a_1x_1 + \cdots + a_nx_n = 0$.

Fortenbacher's algorithm:

- Start with the pair P, M of the set of potential solutions $P = \{e_1\downarrow, \ldots, e_n\downarrow\}$ and the set of minimal nontrivial solutions $M = \emptyset$.
- Apply repeatedly the rules:

1. $\{v\downarrow\} \cup P', M \Longrightarrow P', M$,
   if $v\downarrow \gg u\downarrow$ for some $u\downarrow \in M$.

2. $\{v\downarrow\} \cup P', M \Longrightarrow P', \{v\downarrow\} \cup M$,
   if $a(v\downarrow) = 0$ and rule 1 is not applicable.

3. $P, M \Longrightarrow \{v\downarrow + e_j\downarrow \mid v\downarrow \in P,\ a(v\downarrow)a(e_j\downarrow) < 0,\ j \in 1..n\}, M$,
   if rules 1 and 2 are not applicable.

Single Equation: Algorithm

Case $m = 1$: Single homogeneous LDE $a_1x_1 + \cdots + a_nx_n = 0$.

Fortenbacher's algorithm:

- Start with the pair P, M of the set of potential solutions $P = \{e_1\downarrow, \ldots, e_n\downarrow\}$ and the set of minimal nontrivial solutions $M = \emptyset$.
- Apply repeatedly the rules:

  1. $\{v\downarrow\} \cup P', M \Longrightarrow P', M$,
     if $v\downarrow \gg u\downarrow$ for some $u\downarrow \in M$.

  2. $\{v\downarrow\} \cup P', M \Longrightarrow P', \{v\downarrow\} \cup M$,
     if $a(v\downarrow) = 0$ and rule 1 is not applicable.

  3. $P, M \Longrightarrow \{v\downarrow + e_j\downarrow \mid v\downarrow \in P,\ a(v\downarrow)a(e_j\downarrow) < 0,\ j \in 1..n\}, M$,
     if rules 1 and 2 are not applicable.

- If $\emptyset, M$ is reached, return M.

System of Equations: Idea

General case: System of homogeneous LDEs.

$\mathfrak{a}(\mathfrak{x}\downarrow) = 0\downarrow$.

Generalizing Fortenbacher's idea:

- Search minimal solutions starting from the elements in the canonical basis of $\mathbb{N}^{\mathfrak{n}}$.
- Suppose the current vector $\nu\downarrow$ is not a solution.
- It can be nondeterministically increased, component by component, until it becomes a solution or greater than a solution.
- To decrease the search space, increase only those components that lead to the "right direction".

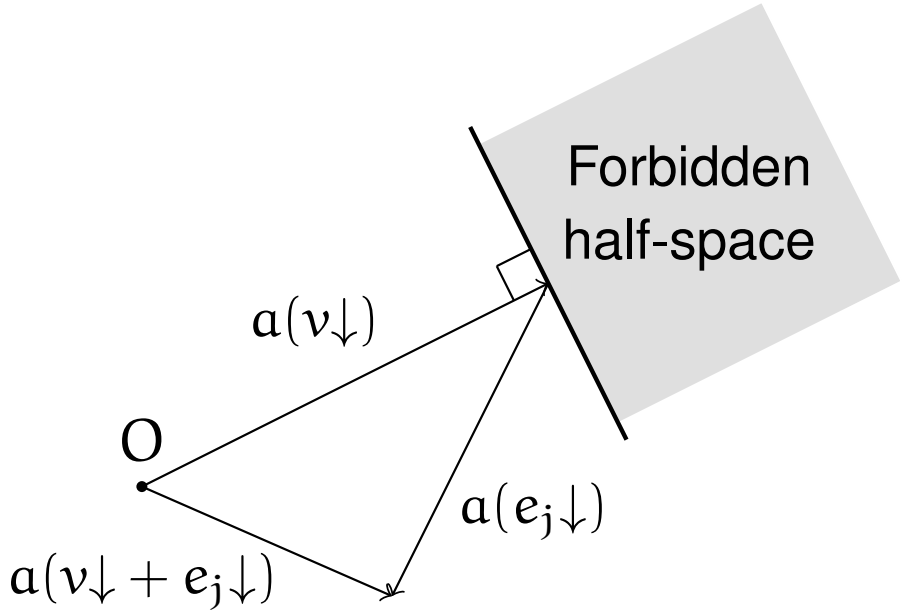
System of Equations: How to Restrict

"Right direction": Towards the origin.

If $a(v\downarrow) \neq 0\downarrow$, then do $a(v\downarrow + e_j\downarrow) = a(v\downarrow) + a(e_j\downarrow)$.

$a(v\downarrow) + a(e_j\downarrow)$ should lie in the half-space containing O.

Contejean-Devie condition: If $a(v\downarrow) \cdot a(e_j\downarrow) < 0$ for some j, increase v_j by one. ($\cdot$ is the scalar product.)



How to Restrict: Comparison

Fortenbacher's condition:

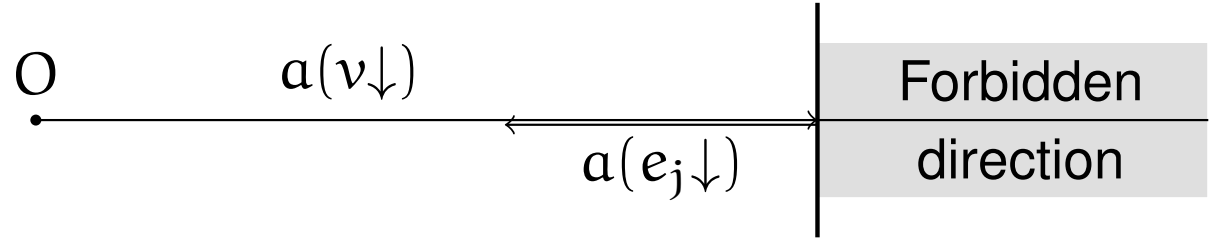
If $a(\nu\downarrow)a(e_j\downarrow) < 0$ for some j, increase ν_j by one.

Contejean-Devie condition:

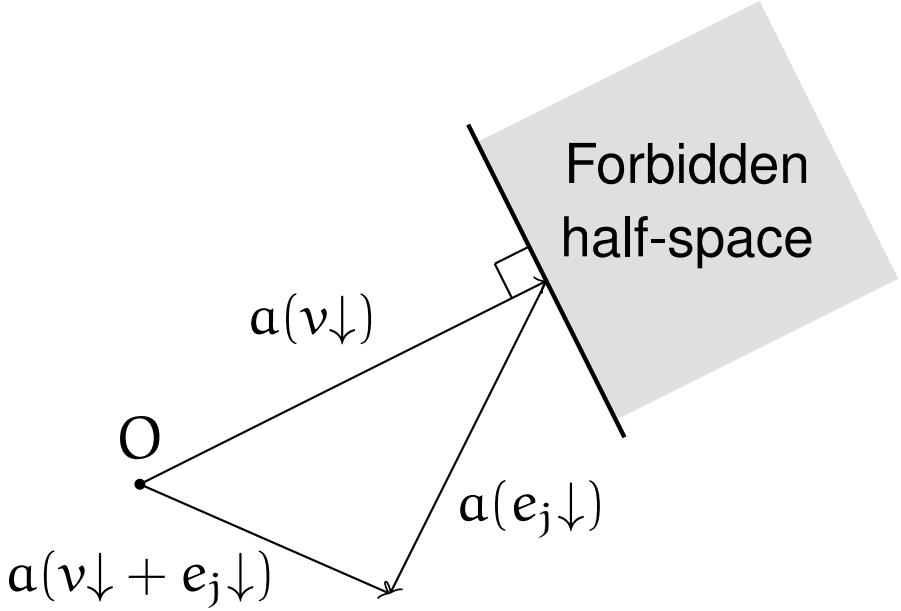
If $a(\nu\downarrow) \cdot a(e_j\downarrow) < 0$ for some j, increase ν_j by one.

How to Restrict: Comparison

Fortenbacher's condition:



Contejean-Devie condition:



System of Equations: Algorithm

System of homogeneous LDEs: $a(x\downarrow) = 0\downarrow$.

Contejean-Devie algorithm:

- Start with the pair P, M where
  - $P = \{e_1\downarrow, \ldots, e_n\downarrow\}$ is the set of potential solutions,
  - $M = \emptyset$ is the set of minimal nontrivial solutions.
- Apply repeatedly the rules:

1. $\{v\downarrow\} \cup P', M \Longrightarrow P', M$,
   if $v\downarrow \gg u\downarrow$ for some $u\downarrow \in M$.

2. $\{v\downarrow\} \cup P', M \Longrightarrow P', \{v\downarrow\} \cup M$,
   if $a(v\downarrow) = 0\downarrow$ and rule 1 is not applicable.

3. $P, M \Longrightarrow \{v\downarrow + e_j\downarrow \mid v\downarrow \in P,\ a(v\downarrow) \cdot a(e_j\downarrow) < 0,\ j \in 1..n\}, M$,
   if rules 1 and 2 are not applicable.

- If $\emptyset, M$ is reached, return M.

Contejean-Devie Algorithm on an Example

$$\begin{cases} -x_1 + x_2 + 2x_3 - 3x_4 = 0 \\ -x_1 + 3x_2 - 2x_3 - x_4 = 0 \end{cases}$$

$$e_1\downarrow = (1,0,0,0)^T \quad e_2\downarrow = (0,1,0,0)^T$$
$$e_3\downarrow = (0,0,1,0)^T \quad e_4\downarrow = (0,0,0,1)^T$$

Start: $\{e_1\downarrow, \ldots, e_4\downarrow\}, \emptyset$.

1. $\{v\downarrow\} \cup P', M \Longrightarrow P', M$,
   if $v\downarrow \gg u\downarrow$ for some $u\downarrow \in M$.
2. $\{v\downarrow\} \cup P', M \Longrightarrow P', \{v\downarrow\} \cup M$,
   if $a(v\downarrow) = 0\downarrow$ and rule 1 is not applicable.
3. $P, M \Longrightarrow \{v\downarrow + e_j\downarrow \mid v\downarrow \in P,$
   $a(v\downarrow) \cdot a(e_j\downarrow) < 0,\ j \in 1..n\}, M$,
   if rules 1 and 2 are not applicable.

Contejean-Devie Algorithm on an Example

$$\begin{cases} -x_1 + x_2 + 2x_3 - 3x_4 = 0 \\ -x_1 + 3x_2 - 2x_3 - x_4 = 0 \end{cases}$$

$$e_1\downarrow = (1,0,0,0)^T \quad e_2\downarrow = (0,1,0,0)^T$$
$$e_3\downarrow = (0,0,1,0)^T \quad e_4\downarrow = (0,0,0,1)^T$$

Start: $\{e_1\downarrow, \ldots, e_4\downarrow\}, \emptyset$.

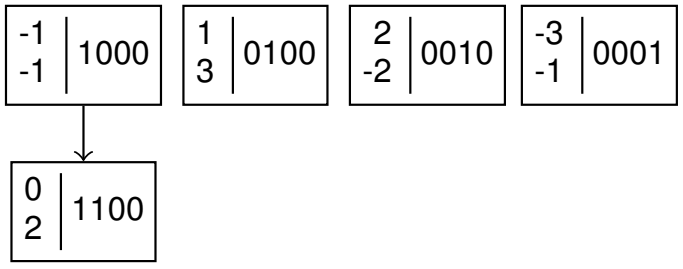
1. $\{v\downarrow\} \cup P', M \Longrightarrow P', M$,
   if $v\downarrow \gg u\downarrow$ for some $u\downarrow \in M$.
2. $\{v\downarrow\} \cup P', M \Longrightarrow P', \{v\downarrow\} \cup M$,
   if $a(v\downarrow) = 0\downarrow$ and rule 1 is not applicable.
3. $P, M \Longrightarrow \{v\downarrow + e_j\downarrow \mid v\downarrow \in P,$
   $a(v\downarrow) \cdot a(e_j\downarrow) < 0,\ j \in 1..n\}, M$,
   if rules 1 and 2 are not applicable.

-1 -1 | 1000	1 3 | 0100	2 -2 | 0010	-3 -1 | 0001

Contejean-Devie Algorithm on an Example

$$\begin{cases} -x_1 + x_2 + 2x_3 - 3x_4 = 0 \\ -x_1 + 3x_2 - 2x_3 - x_4 = 0 \end{cases}$$

$$e_1\downarrow = (1,0,0,0)^T \quad e_2\downarrow = (0,1,0,0)^T$$
$$e_3\downarrow = (0,0,1,0)^T \quad e_4\downarrow = (0,0,0,1)^T$$



Start: $\{e_1\downarrow, \ldots, e_4\downarrow\}, \emptyset$.

1. $\{v\downarrow\} \cup P', M \Longrightarrow P', M$,
   if $v\downarrow \gg u\downarrow$ for some $u\downarrow \in M$.
2. $\{v\downarrow\} \cup P', M \Longrightarrow P', \{v\downarrow\} \cup M$,
   if $a(v\downarrow) = 0\downarrow$ and rule 1 is not applicable.
3. $P, M \Longrightarrow \{v\downarrow + e_j\downarrow \mid v\downarrow \in P,$
   $a(v\downarrow) \cdot a(e_j\downarrow) < 0,\ j \in 1..n\}, M$,
   if rules 1 and 2 are not applicable.

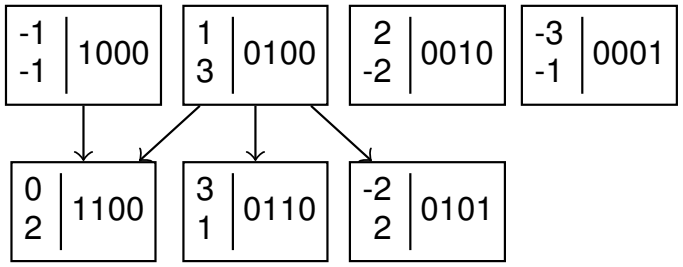
Contejean-Devie Algorithm on an Example

$$\begin{cases} -x_1 + x_2 + 2x_3 - 3x_4 = 0 \\ -x_1 + 3x_2 - 2x_3 - x_4 = 0 \end{cases}$$

$$e_1{\downarrow} = (1,0,0,0)^T \quad e_2{\downarrow} = (0,1,0,0)^T$$
$$e_3{\downarrow} = (0,0,1,0)^T \quad e_4{\downarrow} = (0,0,0,1)^T$$

Start: $\{e_1{\downarrow}, \ldots, e_4{\downarrow}\}, \emptyset$.

1. $\{v{\downarrow}\} \cup P', M \Longrightarrow P', M$,
   if $v{\downarrow} \gg u{\downarrow}$ for some $u{\downarrow} \in M$.
2. $\{v{\downarrow}\} \cup P', M \Longrightarrow P', \{v{\downarrow}\} \cup M$,
   if $a(v{\downarrow}) = 0{\downarrow}$ and rule 1 is not applicable.
3. $P, M \Longrightarrow \{v{\downarrow} + e_j{\downarrow} \mid v{\downarrow} \in P, a(v{\downarrow}) \cdot a(e_j{\downarrow}) < 0,\ j \in 1..n\}, M$,
   if rules 1 and 2 are not applicable.



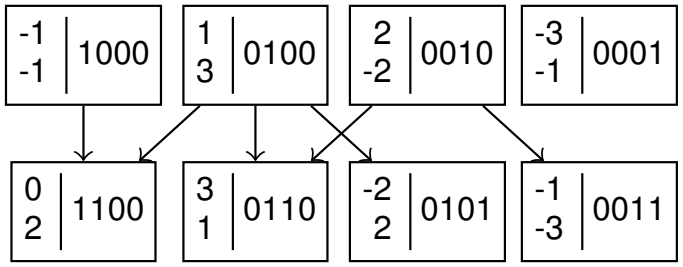
Contejean-Devie Algorithm on an Example

$$\begin{cases} -x_1 + x_2 + 2x_3 - 3x_4 = 0 \\ -x_1 + 3x_2 - 2x_3 - x_4 = 0 \end{cases}$$

$$e_1\downarrow = (1,0,0,0)^T \quad e_2\downarrow = (0,1,0,0)^T$$
$$e_3\downarrow = (0,0,1,0)^T \quad e_4\downarrow = (0,0,0,1)^T$$

Start: $\{e_1\downarrow, \ldots, e_4\downarrow\}, \emptyset$.

1. $\{v\downarrow\} \cup P', M \Longrightarrow P', M$,
   if $v\downarrow \gg u\downarrow$ for some $u\downarrow \in M$.
2. $\{v\downarrow\} \cup P', M \Longrightarrow P', \{v\downarrow\} \cup M$,
   if $a(v\downarrow) = 0\downarrow$ and rule 1 is not applicable.
3. $P, M \Longrightarrow \{v\downarrow + e_j\downarrow \mid v\downarrow \in P, a(v\downarrow) \cdot a(e_j\downarrow) < 0,\ j \in 1..n\}, M$,
   if rules 1 and 2 are not applicable.



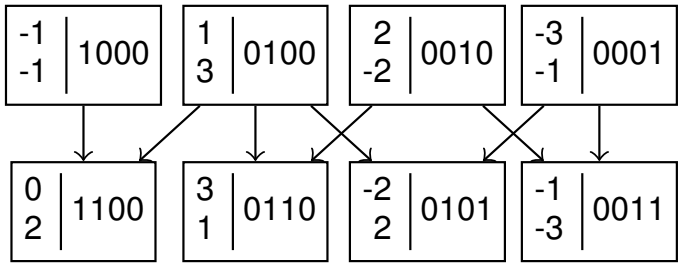
Contejean-Devie Algorithm on an Example

$$\begin{cases} -x_1 + x_2 + 2x_3 - 3x_4 = 0 \\ -x_1 + 3x_2 - 2x_3 - x_4 = 0 \end{cases}$$

$$e_1\downarrow = (1,0,0,0)^T \quad e_2\downarrow = (0,1,0,0)^T$$
$$e_3\downarrow = (0,0,1,0)^T \quad e_4\downarrow = (0,0,0,1)^T$$

Start: $\{e_1\downarrow, \ldots, e_4\downarrow\}, \emptyset$.

1. $\{v\downarrow\} \cup P', M \Longrightarrow P', M$,
   if $v\downarrow \gg u\downarrow$ for some $u\downarrow \in M$.
2. $\{v\downarrow\} \cup P', M \Longrightarrow P', \{v\downarrow\} \cup M$,
   if $a(v\downarrow) = 0\downarrow$ and rule 1 is not applicable.
3. $P, M \Longrightarrow \{v\downarrow + e_j\downarrow \mid v\downarrow \in P, a(v\downarrow) \cdot a(e_j\downarrow) < 0,\ j \in 1..n\}, M$,
   if rules 1 and 2 are not applicable.



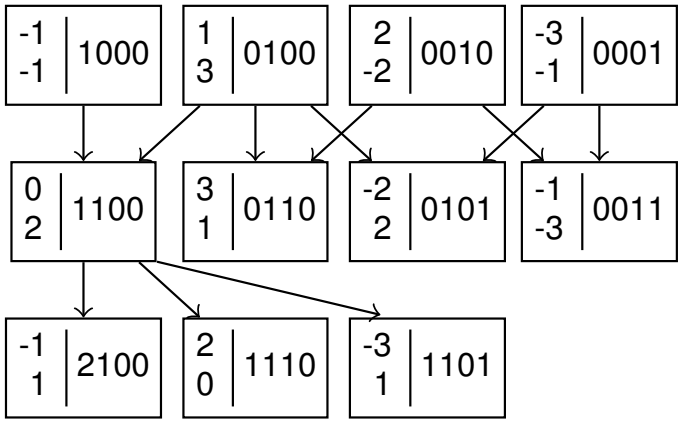
Contejean-Devie Algorithm on an Example

$$\begin{cases} -x_1 + x_2 + 2x_3 - 3x_4 = 0 \\ -x_1 + 3x_2 - 2x_3 - x_4 = 0 \end{cases}$$

$$e_1\downarrow = (1,0,0,0)^T \quad e_2\downarrow = (0,1,0,0)^T$$
$$e_3\downarrow = (0,0,1,0)^T \quad e_4\downarrow = (0,0,0,1)^T$$

Start: $\{e_1\downarrow, \ldots, e_4\downarrow\}, \emptyset$.

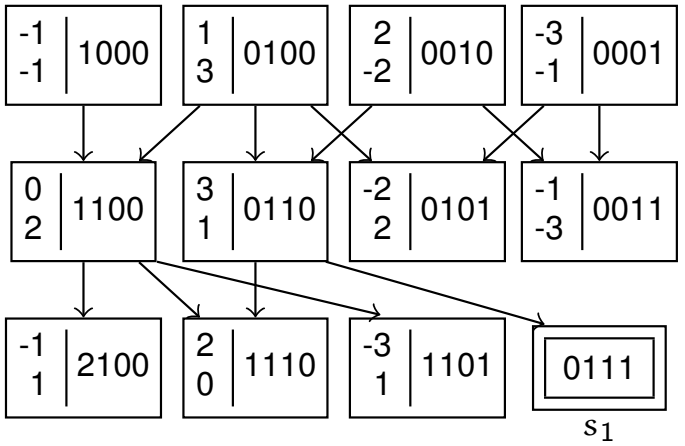
1. $\{v\downarrow\} \cup P', M \Longrightarrow P', M$,
   if $v\downarrow \gg u\downarrow$ for some $u\downarrow \in M$.
2. $\{v\downarrow\} \cup P', M \Longrightarrow P', \{v\downarrow\} \cup M$,
   if $a(v\downarrow) = 0\downarrow$ and rule 1 is not applicable.
3. $P, M \Longrightarrow \{v\downarrow + e_j\downarrow \mid v\downarrow \in P, a(v\downarrow) \cdot a(e_j\downarrow) < 0,\ j \in 1..n\}, M$,
   if rules 1 and 2 are not applicable.



Contejean-Devie Algorithm on an Example

$$\begin{cases} -x_1 + x_2 + 2x_3 - 3x_4 = 0 \\ -x_1 + 3x_2 - 2x_3 - x_4 = 0 \end{cases}$$

$$e_1\downarrow = (1,0,0,0)^T \quad e_2\downarrow = (0,1,0,0)^T$$
$$e_3\downarrow = (0,0,1,0)^T \quad e_4\downarrow = (0,0,0,1)^T$$



Start: $\{e_1\downarrow, \ldots, e_4\downarrow\}, \emptyset$.

1. $\{v\downarrow\} \cup P', M \Longrightarrow P', M$,
   if $v\downarrow \gg u\downarrow$ for some $u\downarrow \in M$.
2. $\{v\downarrow\} \cup P', M \Longrightarrow P', \{v\downarrow\} \cup M$,
   if $a(v\downarrow) = 0\downarrow$ and rule 1 is not applicable.
3. $P, M \Longrightarrow \{v\downarrow + e_j\downarrow \mid v\downarrow \in P, a(v\downarrow) \cdot a(e_j\downarrow) < 0,\ j \in 1..n\}, M$,
   if rules 1 and 2 are not applicable.

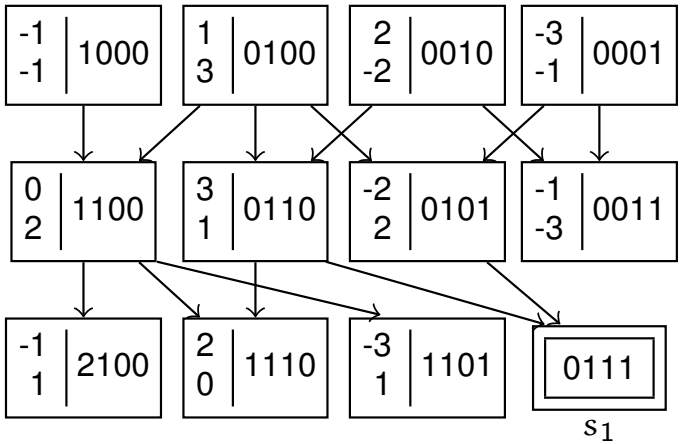
Contejean-Devie Algorithm on an Example

$$\begin{cases} -x_1 + x_2 + 2x_3 - 3x_4 = 0 \\ -x_1 + 3x_2 - 2x_3 - x_4 = 0 \end{cases}$$

$$e_1{\downarrow} = (1,0,0,0)^T \quad e_2{\downarrow} = (0,1,0,0)^T$$
$$e_3{\downarrow} = (0,0,1,0)^T \quad e_4{\downarrow} = (0,0,0,1)^T$$

Start: $\{e_1{\downarrow}, \ldots, e_4{\downarrow}\}, \emptyset$.

1. $\{v{\downarrow}\} \cup P', M \Longrightarrow P', M$,
   if $v{\downarrow} \gg u{\downarrow}$ for some $u{\downarrow} \in M$.
2. $\{v{\downarrow}\} \cup P', M \Longrightarrow P', \{v{\downarrow}\} \cup M$,
   if $a(v{\downarrow}) = 0{\downarrow}$ and rule 1 is not applicable.
3. $P, M \Longrightarrow \{v{\downarrow} + e_j{\downarrow} \mid v{\downarrow} \in P, a(v{\downarrow}) \cdot a(e_j{\downarrow}) < 0, j \in 1..n\}, M$,
   if rules 1 and 2 are not applicable.



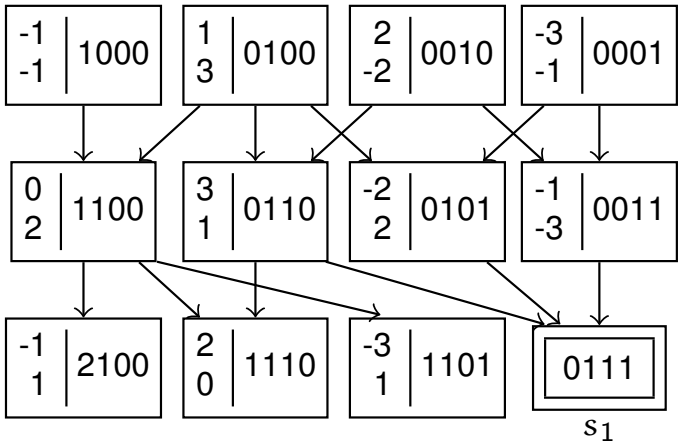
Contejean-Devie Algorithm on an Example

$$\begin{cases} -x_1 + x_2 + 2x_3 - 3x_4 = 0 \\ -x_1 + 3x_2 - 2x_3 - x_4 = 0 \end{cases}$$

$$e_1{\downarrow} = (1,0,0,0)^T \quad e_2{\downarrow} = (0,1,0,0)^T$$
$$e_3{\downarrow} = (0,0,1,0)^T \quad e_4{\downarrow} = (0,0,0,1)^T$$

Start: $\{e_1{\downarrow}, \ldots, e_4{\downarrow}\}, \emptyset$.

1. $\{v{\downarrow}\} \cup P', M \Longrightarrow P', M$,
   if $v{\downarrow} \gg u{\downarrow}$ for some $u{\downarrow} \in M$.
2. $\{v{\downarrow}\} \cup P', M \Longrightarrow P', \{v{\downarrow}\} \cup M$,
   if $a(v{\downarrow}) = 0{\downarrow}$ and rule 1 is not applicable.
3. $P, M \Longrightarrow \{v{\downarrow} + e_j{\downarrow} \mid v{\downarrow} \in P, a(v{\downarrow}) \cdot a(e_j{\downarrow}) < 0, j \in 1..n\}, M$,
   if rules 1 and 2 are not applicable.



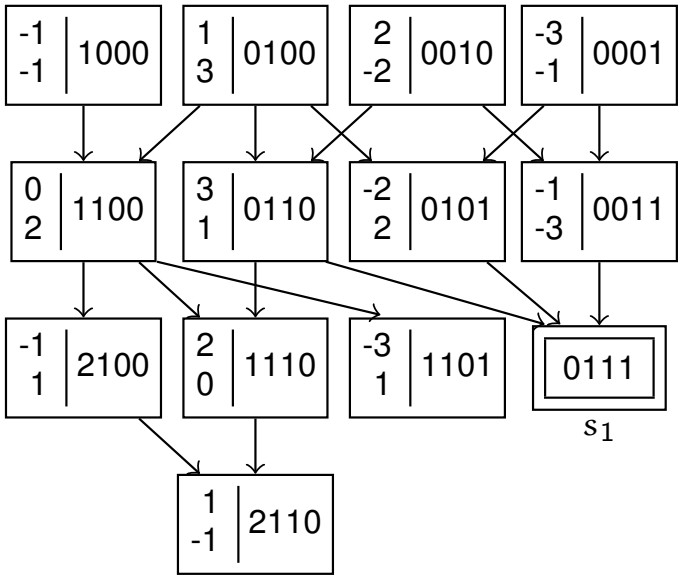
Contejean-Devie Algorithm on an Example

$$\begin{cases} -x_1 + x_2 + 2x_3 - 3x_4 = 0 \\ -x_1 + 3x_2 - 2x_3 - x_4 = 0 \end{cases}$$

$$e_1\downarrow = (1,0,0,0)^T \quad e_2\downarrow = (0,1,0,0)^T$$
$$e_3\downarrow = (0,0,1,0)^T \quad e_4\downarrow = (0,0,0,1)^T$$

Start: $\{e_1\downarrow, \ldots, e_4\downarrow\}, \emptyset$.

1. $\{v\downarrow\} \cup P', M \Longrightarrow P', M$,
   if $v\downarrow \gg u\downarrow$ for some $u\downarrow \in M$.
2. $\{v\downarrow\} \cup P', M \Longrightarrow P', \{v\downarrow\} \cup M$,
   if $a(v\downarrow) = 0\downarrow$ and rule 1 is not applicable.
3. $P, M \Longrightarrow \{v\downarrow + e_j\downarrow \mid v\downarrow \in P, a(v\downarrow) \cdot a(e_j\downarrow) < 0,\ j \in 1..n\}, M$,
   if rules 1 and 2 are not applicable.



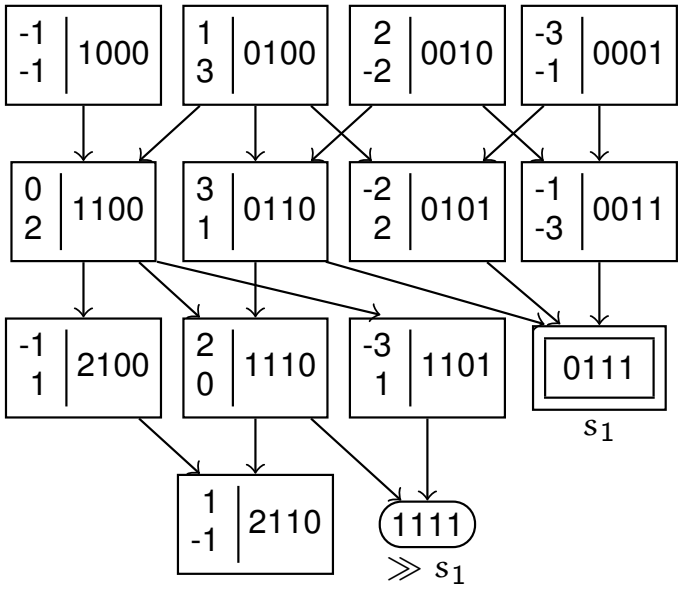
Contejean-Devie Algorithm on an Example

$$\begin{cases} -x_1 + x_2 + 2x_3 - 3x_4 = 0 \\ -x_1 + 3x_2 - 2x_3 - x_4 = 0 \end{cases}$$

$$e_1\downarrow = (1,0,0,0)^T \quad e_2\downarrow = (0,1,0,0)^T$$
$$e_3\downarrow = (0,0,1,0)^T \quad e_4\downarrow = (0,0,0,1)^T$$

Start: $\{e_1\downarrow, \ldots, e_4\downarrow\}, \emptyset$.

1. $\{v\downarrow\} \cup P', M \Longrightarrow P', M$,
   if $v\downarrow \gg u\downarrow$ for some $u\downarrow \in M$.
2. $\{v\downarrow\} \cup P', M \Longrightarrow P', \{v\downarrow\} \cup M$,
   if $a(v\downarrow) = 0\downarrow$ and rule 1 is not applicable.
3. $P, M \Longrightarrow \{v\downarrow + e_j\downarrow \mid v\downarrow \in P, a(v\downarrow) \cdot a(e_j\downarrow) < 0, j \in 1..n\}, M$,
   if rules 1 and 2 are not applicable.



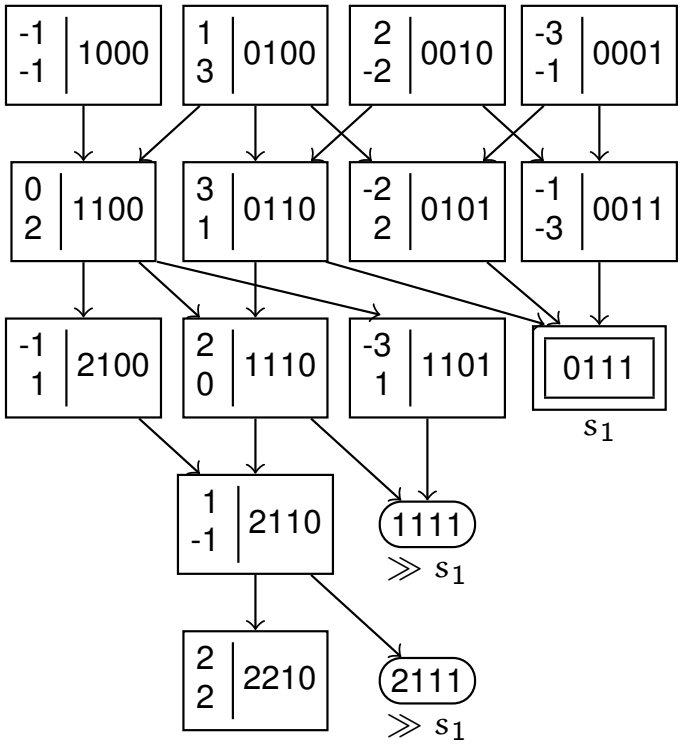
Contejean-Devie Algorithm on an Example

$$\begin{cases} -x_1 + x_2 + 2x_3 - 3x_4 = 0 \\ -x_1 + 3x_2 - 2x_3 - x_4 = 0 \end{cases}$$

$$e_1{\downarrow} = (1,0,0,0)^T \quad e_2{\downarrow} = (0,1,0,0)^T$$
$$e_3{\downarrow} = (0,0,1,0)^T \quad e_4{\downarrow} = (0,0,0,1)^T$$

Start: $\{e_1{\downarrow}, \ldots, e_4{\downarrow}\}, \emptyset$.

1. $\{v{\downarrow}\} \cup P', M \Longrightarrow P', M$,
   if $v{\downarrow} \gg u{\downarrow}$ for some $u{\downarrow} \in M$.
2. $\{v{\downarrow}\} \cup P', M \Longrightarrow P', \{v{\downarrow}\} \cup M$,
   if $a(v{\downarrow}) = 0{\downarrow}$ and rule 1 is not applicable.
3. $P, M \Longrightarrow \{v{\downarrow} + e_j{\downarrow} \mid v{\downarrow} \in P, a(v{\downarrow}) \cdot a(e_j{\downarrow}) < 0, j \in 1..n\}, M$,
   if rules 1 and 2 are not applicable.



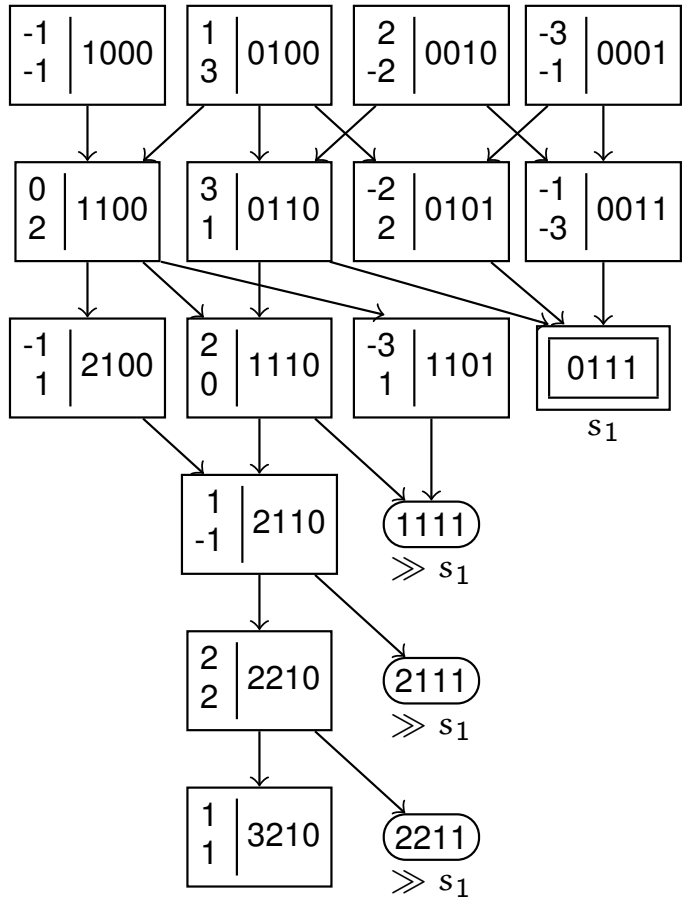
Contejean-Devie Algorithm on an Example

$$\begin{cases} -x_1 + x_2 + 2x_3 - 3x_4 = 0 \\ -x_1 + 3x_2 - 2x_3 - x_4 = 0 \end{cases}$$

$$e_1{\downarrow} = (1,0,0,0)^T \quad e_2{\downarrow} = (0,1,0,0)^T$$
$$e_3{\downarrow} = (0,0,1,0)^T \quad e_4{\downarrow} = (0,0,0,1)^T$$

Start: $\{e_1{\downarrow}, \ldots, e_4{\downarrow}\}, \emptyset$.

1. $\{v{\downarrow}\} \cup P', M \Longrightarrow P', M$,
   if $v{\downarrow} \gg u{\downarrow}$ for some $u{\downarrow} \in M$.
2. $\{v{\downarrow}\} \cup P', M \Longrightarrow P', \{v{\downarrow}\} \cup M$,
   if $a(v{\downarrow}) = 0{\downarrow}$ and rule 1 is not applicable.
3. $P, M \Longrightarrow \{v{\downarrow} + e_j{\downarrow} \mid v{\downarrow} \in P,$
   $a(v{\downarrow}) \cdot a(e_j{\downarrow}) < 0,\ j \in 1..n\}, M$,
   if rules 1 and 2 are not applicable.



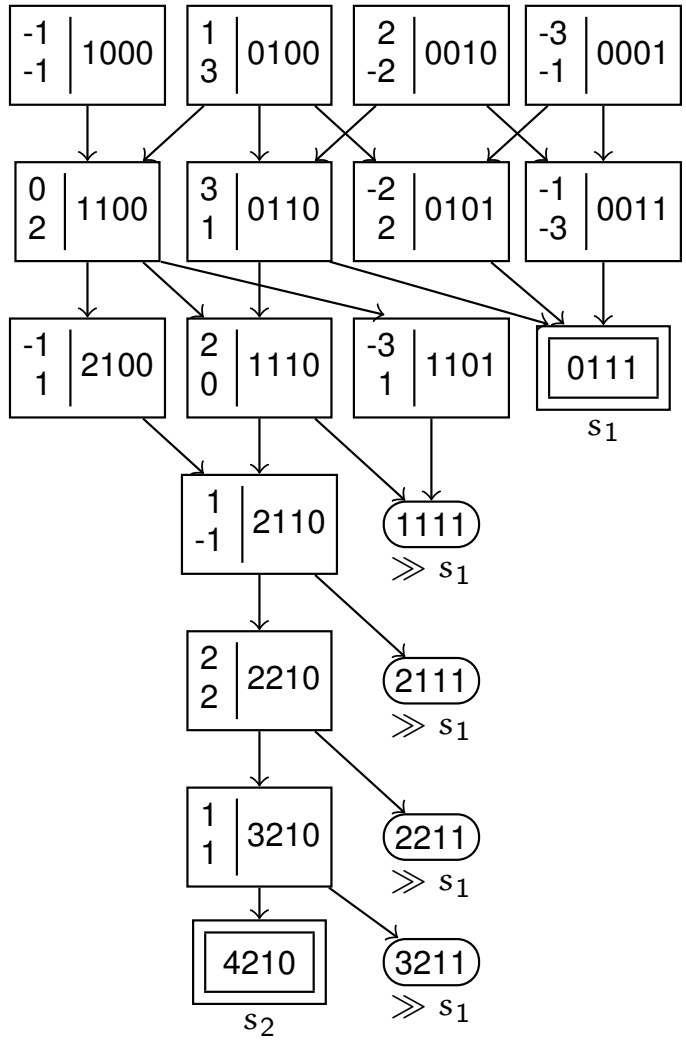
Contejean-Devie Algorithm on an Example

$$\begin{cases} -x_1 + x_2 + 2x_3 - 3x_4 = 0 \\ -x_1 + 3x_2 - 2x_3 - x_4 = 0 \end{cases}$$

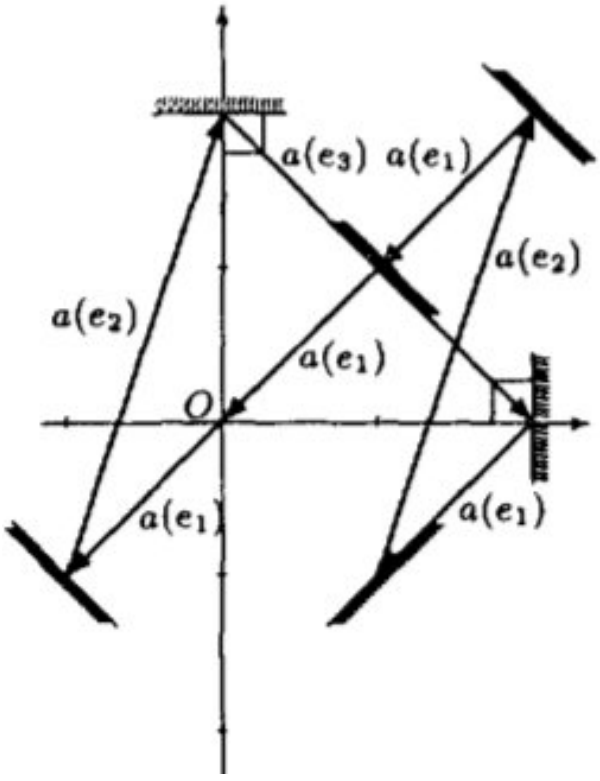
$$e_1\downarrow = (1,0,0,0)^T \quad e_2\downarrow = (0,1,0,0)^T$$
$$e_3\downarrow = (0,0,1,0)^T \quad e_4\downarrow = (0,0,0,1)^T$$

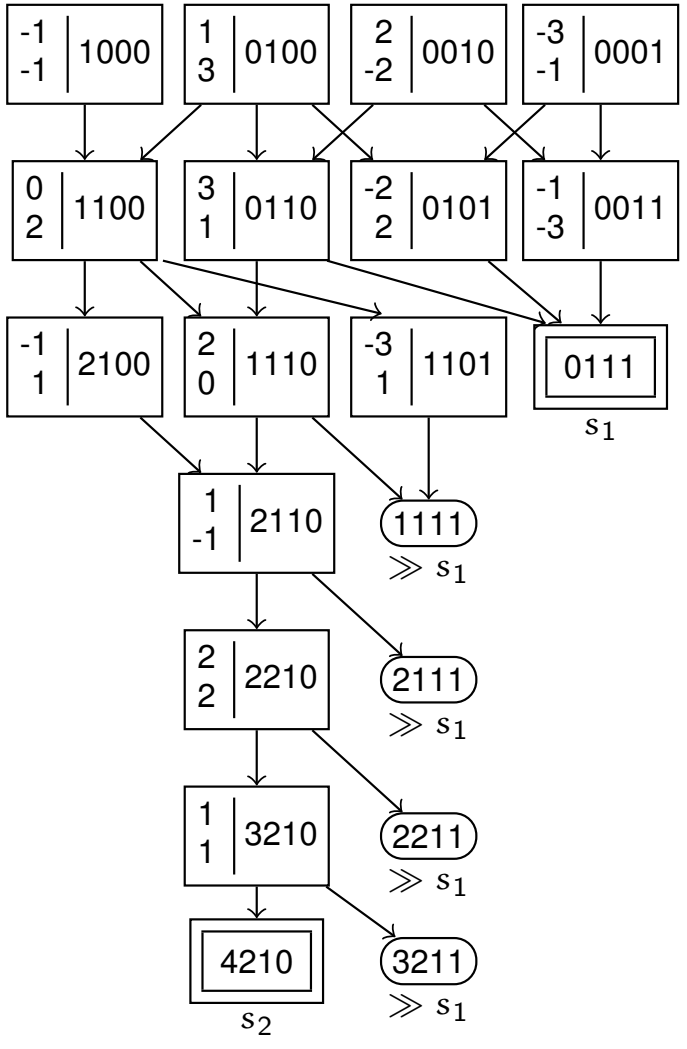
Start: $\{e_1\downarrow, \ldots, e_4\downarrow\}, \emptyset$.

1. $\{v\downarrow\} \cup P', M \Longrightarrow P', M$,
   if $v\downarrow \gg u\downarrow$ for some $u\downarrow \in M$.
2. $\{v\downarrow\} \cup P', M \Longrightarrow P', \{v\downarrow\} \cup M$,
   if $a(v\downarrow) = 0\downarrow$ and rule 1 is not applicable.
3. $P, M \Longrightarrow \{v\downarrow + e_j\downarrow \mid v\downarrow \in P,$
   $a(v\downarrow) \cdot a(e_j\downarrow) < 0,\ j \in 1..n\}, M$,
   if rules 1 and 2 are not applicable.



Contejean-Devie Algorithm on an Example





Properties of the Algorithm

$\mathfrak{a}(\mathfrak{x}\downarrow) = 0\downarrow$: An $\mathfrak{n}$-variate system of homogeneous LDEs.

$(e_1\downarrow, \ldots, e_n\downarrow)$: The canonical basis of $\mathbb{N}^n$.

$\mathcal{B}(\mathfrak{a}(\mathfrak{x}\downarrow) = 0\downarrow)$: Basis in the set of nontrivial natural solutions of $\mathfrak{a}(\mathfrak{x}\downarrow) = 0\downarrow$.

Theorem

- The Contejean-Devie algorithm terminates on any input.
- Let $(e_1\downarrow, \ldots, e_n\downarrow), \emptyset \Longrightarrow^* \emptyset, \mathcal{M}$ be the sequence of transformations performed by the Contejean-Devie algorithm for $\mathfrak{a}(\mathfrak{x}\downarrow) = 0\downarrow$. Then

$$\mathcal{B}(\mathfrak{a}(\mathfrak{x}\downarrow) = 0\downarrow) = \mathcal{M}.$$

Jump to the non-homogeneous case

Notation

- $\|x\downarrow\| = \sqrt{x_1^2 + \cdots + x_n^2}$.
- $|(s_1, \ldots, s_n)| = s_1 + \cdots + s_n$.

Completeness

Theorem

Let $P_0, M_0 \Longrightarrow^* \emptyset, M$ be the sequence of transformations performed by the Contejean-Devie algorithm for $a(x\downarrow) = 0\downarrow$ with $P_0 = (e_1\downarrow, \ldots, e_n\downarrow)$ and $M_0 = \emptyset$. Then $\mathcal{B}(a(x\downarrow) = 0\downarrow) \subseteq M$.

Proof.

Assume $s\downarrow \in \mathcal{B}(a(x\downarrow) = 0\downarrow)$ and show that there exists a sequence of vectors

$$v_1\downarrow = e_{j_0}\downarrow \ll \cdots \ll v_k\downarrow \ll v_{k+1}\downarrow = v_k\downarrow + e_{j_k}\downarrow \ll \cdots \ll v_{|s\downarrow|}\downarrow = s\downarrow$$

such that $v_i\downarrow \in P_{l_i}$, where P_{l_i} is from the given sequence of transformations and $l_i < l_j$ for $i < j$.

Completeness

Theorem

Let $P_0, M_0 \Longrightarrow^* \emptyset, M$ be the sequence of transformations performed by the Contejean-Devie algorithm for $a(x\downarrow) = 0\downarrow$ with $P_0 = (e_1\downarrow, \ldots, e_n\downarrow)$ and $M_0 = \emptyset$. Then $\mathcal{B}(a(x\downarrow) = 0\downarrow) \subseteq M$.

Proof (cont.)

For $e_{j0}\downarrow$, any basic vector $\ll s\downarrow$ can be chosen.

Such basic vectors do exist (since $s\downarrow \neq 0\downarrow$) and are in P_0.

Assume now we have $v_1\downarrow \ll \cdots \ll v_k\downarrow \ll s\downarrow$ with $v_k\downarrow \in P_{l_k}$.

Then there exists $s_k\downarrow$ with $s\downarrow = v_k\downarrow + s_k\downarrow$ and
$0 = \|a(s\downarrow)\|^2 = \|a(v_k\downarrow)\|^2 + \|a(s_k\downarrow)\|^2 + 2a(v_k\downarrow) \cdot a(s_k\downarrow)$.

It implies $a(v_k\downarrow) \cdot a(s_k\downarrow) < 0$.

Completeness

Theorem

Let $P_0, M_0 \Longrightarrow^* \emptyset, M$ be the sequence of transformations performed by the Contejean-Devie algorithm for $a(x\downarrow) = 0\downarrow$ with $P_0 = (e_1\downarrow, \ldots, e_n\downarrow)$ and $M_0 = \emptyset$. Then $\mathcal{B}(a(x\downarrow) = 0\downarrow) \subseteq M$.

Proof (cont.)

Hence, there exists $e_{j_k}\downarrow$ with $s_k\downarrow \gg e_{j_k}\downarrow$ such that $a(v_k\downarrow) \cdot a(e_{j_k}\downarrow) < 0$.

We take $v_{k+1}\downarrow = v_k\downarrow + e_{j_k}\downarrow$.

Then $s\downarrow \gg v_{k+1}\downarrow$ and by rule 3, $v_{k+1}\downarrow \in P_{l_{k+1}}$.

After $|s\downarrow|$ steps, we reach s. Hence, $s\downarrow \in P_{l_{|s|}}$.

Since $a(s\downarrow) = 0$, application of rule 2 moves $s\downarrow$ to M. □

Soundness

Theorem

Let $\mathrm{P}_0, \mathrm{M}_0 \Longrightarrow^* \emptyset, \mathrm{M}$ be the sequence of transformations performed by the Contejean-Devie algorithm for $\mathrm{a}(\mathrm{x}\downarrow) = 0\downarrow$ with $\mathrm{P}_0 = (\mathrm{e}_1\downarrow, \ldots, \mathrm{e}_\mathrm{n}\downarrow)$ and $\mathrm{M}_0 = \emptyset$. Then $\mathrm{M} \subseteq \mathcal{B}(\mathrm{a}(\mathrm{x}\downarrow) = 0\downarrow)$.

Proof.

Any $s\downarrow \in \mathrm{M}$ is a solution.

Show that it is minimal.

Assume it is not: $s\downarrow = s_1\downarrow + s_2\downarrow$, where $s_1\downarrow$ and $s_2\downarrow$ are non-null solutions smaller than s.

Soundness

Theorem

Let $\mathrm{P}_0, \mathrm{M}_0 \Longrightarrow^* \emptyset, \mathrm{M}$ be the sequence of transformations performed by the Contejean-Devie algorithm for $\mathrm{a}(\mathrm{x}\downarrow) = 0\downarrow$ with $\mathrm{P}_0 = (e_1\downarrow, \ldots, e_n\downarrow)$ and $\mathrm{M}_0 = \emptyset$. Then $\mathrm{M} \subseteq \mathcal{B}(\mathrm{a}(\mathrm{x}\downarrow) = 0\downarrow)$.

Proof (Cont.).

Assume $s\downarrow$ was obtained during the transformations as $s\downarrow = v_i\downarrow + e_{j_i}\downarrow$, where $v_i\downarrow \in \mathrm{P}_i$.

But then $v_i\downarrow \gg s_1\downarrow$ or $v_i\downarrow = s_1\downarrow$ or $v_i\downarrow \gg s_2\downarrow$ or $v_i\downarrow = s_1\downarrow$ and $v_i\downarrow$ is greater than an already computed minimal solution.

Therefore, it should have been removed from P_i.
A contradiction. □

Termination

Theorem

Let $v_1\downarrow, v_2\downarrow, \ldots$ be an infinite sequence satisfying the Contejean-Devie condition for $a(x\downarrow) = 0\downarrow$:

- $v_1\downarrow$ is a basic vector and for each $i \geqslant 1$ there exists $1 \leqslant j \leqslant n$ such that $a(v_i\downarrow) \cdot a(e_j\downarrow) < 0$ and $v_{i+1}\downarrow = v_i\downarrow + e_j\downarrow$.

Then there exist $v\downarrow$ and k such that

- $v\downarrow$ is a solution of $a(x\downarrow) = 0\downarrow$, and
- $v\downarrow \ll v_k\downarrow$.

Non-Homogeneous Case

Non-homogeneous linear Diophantine system with m equations and n variables:

$$\begin{cases} a_{11}x_1 & +\cdots+ & a_{1n}x_n & = & b_1 \\ \vdots & & \vdots & & \vdots \\ a_{m1}x_1 & +\cdots+ & a_{mn}x_n & = & b_m \end{cases}$$

- a's and b's are integers.
- Matrix form: $a(x\downarrow) = b\downarrow$.

Non-Homogeneous Case. Solving Idea

Turn the system into a homogeneous one, denoted S_0:

$$\begin{cases} -b_1 x_0 + a_{11} x_1 + \cdots + a_{1n} x_n = 0 \\ \vdots \qquad\quad \vdots \qquad\qquad \vdots \qquad\quad \vdots \\ -b_m x_0 + a_{m1} x_1 + \cdots + a_{mn} x_n = 0 \end{cases}$$

Solve S_0 and keep only the solutions with $x_0 \leqslant 1$.

$x_0 = 1$: a minimal solution for $a(x\downarrow) = b\downarrow$.

$x_0 = 0$: a minimal solution for $a(x\downarrow) = 0\downarrow$.

Non-Homogeneous Case. Solving Idea

Any solution of the non-homogeneous system $\mathfrak{a}(\mathfrak{x}\downarrow) = \mathfrak{b}\downarrow$ has the form $\mathfrak{x}\downarrow + \mathfrak{y}\downarrow$ where:

- $\mathfrak{x}\downarrow$ is a minimal solution of $\mathfrak{a}(\mathfrak{x}\downarrow) = \mathfrak{b}\downarrow$.
- $\mathfrak{y}\downarrow$ is a linear combination (with natural coefficients) of minimal solutions of $\mathfrak{a}(\mathfrak{x}\downarrow) = 0\downarrow$.

Back to ACU-Unification

Theorem

The decision problem for both ACU-Matching and ACU-unification with constants is NP-complete.

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 $\mathbb{E}$-unification procedure).

General Results

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

Rules:

- Trivial, Orient, Decomposition, Elimination, plus
- Lazy Paramodulation:

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

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

General Results

Universal E-unification procedure. Control.

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

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

General Results: Example

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

Unification problem: $\{f(x, x) \stackrel{\cdot}{=}^{?}_{E} x\}$.

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

$$\begin{aligned}
&\{f(x, x) \stackrel{\cdot}{=}^{?}_{E} x\}; \emptyset \\
&\quad \Longrightarrow_{LP} \{f(a, b) \stackrel{\cdot}{=}^{?}_{E} f(x, x), a \stackrel{\cdot}{=}^{?}_{E} x\}; \emptyset \\
&\quad \Longrightarrow_{D} \{a \stackrel{\cdot}{=}^{?}_{E} x, b \stackrel{\cdot}{=}^{?}_{E} x\}; \emptyset \\
&\quad \Longrightarrow_{O} \{x \stackrel{\cdot}{=}^{?}_{E} a, b \stackrel{\cdot}{=}^{?}_{E} x\}; \emptyset \\
&\quad \Longrightarrow_{S} \{b \stackrel{\cdot}{=}^{?}_{E} a\}; \{x \stackrel{\cdot}{=} a\} \\
&\quad \Longrightarrow_{LP} \{a \stackrel{\cdot}{=}^{?}_{E} a, b \stackrel{\cdot}{=}^{?}_{E} b\}; \{x \stackrel{\cdot}{=} a\} \\
&\quad \Longrightarrow^{+}_{T} \emptyset; \{x \stackrel{\cdot}{=} a\}
\end{aligned}$$

General Results

Pros and cons of the universal procedure:

- Pros: Is sound and complete. Can be used for any $\mathbb{E}$.
- Cons: Very inefficient. Usually does not yield a decision procedure or a (minimal) $\mathbb{E}$-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.

Summary

Syntactic unification and matching:

- Unification and matching algorithms.

Equational unification and matching:

- Classification with respect to unification type.
- Algorithms for commutative and ACU-unification, including solving systems of linear Diophantine equations.
- Universal $\mathbb{E}$-unification procedure.