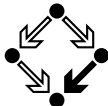


Rewriting

Part 4. Equational Problems. Syntactic Unification and Matching

Temur Kutsia

RISC, JKU Linz



Validity and satisfiability

Notation: $s \approx_E t$ iff $s \approx t$ belongs to the equational theory generated by E .

Validity and satisfiability

Notation: $s \approx_E t$ iff $s \approx t$ belongs to the equational theory generated by E .

Validity problem:

Given: A set of identities E and terms s and t .

Decide: $s \approx_E t$.

Validity and satisfiability

Notation: $s \approx_E t$ iff $s \approx t$ belongs to the equational theory generated by E .

Validity problem:

Given: A set of identities E and terms s and t .

Decide: $s \approx_E t$.

Satisfiability problem:

Given: A set of identities E and terms s and t .

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

Equational problems

The following methods solve special cases:

- ▶ **Term rewriting** decides \approx_E if \rightarrow_E is convergent.
(Discussed in the previous lecture)

Equational problems

The following methods solve special cases:

- ▶ **Term rewriting** decides \approx_E if \rightarrow_E is convergent.
(Discussed in the previous lecture)
- ▶ **Congruence closure** decides \approx_E when E is variable-free.
(Discussed in the previous lecture)

Equational problems

The following methods solve special cases:

- ▶ **Term rewriting** decides \approx_E if \rightarrow_E is convergent.
(Discussed in the previous lecture)
- ▶ **Congruence closure** decides \approx_E when E is variable-free.
(Discussed in the previous lecture)
- ▶ **Syntactic unification** computes σ such that $\sigma(s) = \sigma(t)$.
(Today)

Unification

Unification is the process of solving satisfiability problems:

Given: A set of identities E and two terms s and t .

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

Unification

Unification is the process of solving satisfiability problems:

Given: A set of identities E and two terms s and t .

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

- ▶ In syntactic unification, $E = \emptyset$.
- ▶ $r_1 \approx_{\emptyset} r_2$ iff $r_1 = r_2$.

Unification

Unification is the process of solving satisfiability problems:

Given: A set of identities E and two terms s and t .

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

► In syntactic unification, $E = \emptyset$.

► $r_1 \approx_{\emptyset} r_2$ iff $r_1 = r_2$.

Syntactic unification:

Given: Two terms s and t .

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

Unification

Syntactic unification:

Given: Two terms s and t .

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

- ▶ σ : a **unifier** of s and t .
- ▶ σ : a **solution** of the equation $s =^? t$.

Examples

$f(x) \stackrel{?}{=} f(a)$: exactly one unifier $\{x \mapsto a\}$

$x \stackrel{?}{=} f(y)$: infinitely many unifiers

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

$f(x) \stackrel{?}{=} g(y)$: no unifiers

$x \stackrel{?}{=} f(x)$: no unifiers

Examples

$x =^? f(y)$: infinitely many unifiers
 $\{x \mapsto f(y)\}, \{x \mapsto f(a), y \mapsto a\}, \dots$

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

Substitutions

Instantiation Quasi-Ordering

- ▶ A substitution σ is **more general** than ϑ , written $\sigma \lesssim \vartheta$, if there exists η such that $\eta\sigma = \vartheta$.
- ▶ ϑ is called an **instance** of σ .
- ▶ The relation \lesssim is quasi-ordering (reflexive and transitive binary relation), called **instantiation quasi-ordering**.
- ▶ \sim is the equivalence relation corresponding to \lesssim , i.e., the relation $\lesssim \cap \gtrsim$.

Example 4.2

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

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

Substitutions

Definition 4.2 (Variable Renaming)

A substitution $\sigma = \{x_1 \mapsto y_1, x_2 \mapsto y_2, \dots, x_n \mapsto y_n\}$ is called **variable renaming** iff $\{x_1, \dots, x_n\} = \{y_1, \dots, y_n\}$.
(Permuting the domain variables.)

Example 4.3

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

Substitutions

Definition 4.3 (Idempotent Substitution)

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

Example 4.4

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

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

Substitutions

Lemma 4.2

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

Proof.

Exercise. □

Substitutions

Lemma 4.2

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

Proof.

Exercise. □

Example 4.5

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

Substitutions

Theorem 4.4

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

Proof.

Exercise. □

Substitutions

Definition 4.4 (Unification Problem, Unifier, MGU)

- ▶ **Unification problem:** A finite set of equations

$$\Gamma = \{s_1 =? t_1, \dots, s_n =? t_n\}.$$

Substitutions

Definition 4.4 (Unification Problem, Unifier, MGU)

- ▶ **Unification problem:** A finite set of equations $\Gamma = \{s_1 =? t_1, \dots, s_n =? t_n\}$.
- ▶ **Unifier** or **solution** of Γ : A substitution σ such that $\sigma(s_i) = \sigma(t_i)$ for all $1 \leq i \leq n$.

Substitutions

Definition 4.4 (Unification Problem, Unifier, MGU)

- ▶ **Unification problem:** A finite set of equations $\Gamma = \{s_1 =? t_1, \dots, s_n =? t_n\}$.
- ▶ **Unifier** or **solution** of Γ : A substitution σ such that $\sigma(s_i) = \sigma(t_i)$ for all $1 \leq i \leq n$.
- ▶ $\mathcal{U}(\Gamma)$: The set of all unifiers of Γ . Γ is **unifiable** iff $\mathcal{U}(\Gamma) \neq \emptyset$.

Substitutions

Definition 4.4 (Unification Problem, Unifier, MGU)

- ▶ **Unification problem:** A finite set of equations $\Gamma = \{s_1 =? t_1, \dots, s_n =? t_n\}$.
- ▶ **Unifier** or **solution** of Γ : A substitution σ such that $\sigma(s_i) = \sigma(t_i)$ for all $1 \leq i \leq n$.
- ▶ $\mathcal{U}(\Gamma)$: The set of all unifiers of Γ . Γ is **unifiable** iff $\mathcal{U}(\Gamma) \neq \emptyset$.
- ▶ σ is a **most general unifier (mgu)** of Γ iff it is a least element of $\mathcal{U}(\Gamma)$:
 - ▶ $\sigma \in \mathcal{U}(\Gamma)$, and
 - ▶ $\sigma \lesssim \vartheta$ for every $\vartheta \in \mathcal{U}(\Gamma)$.

Unifiers

Example 4.6

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

For any other unifier ϑ of $x \stackrel{?}{=} y$, $\sigma \lesssim \vartheta$ because

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

Unifiers

Example 4.6

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

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

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

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

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

Unifiers

Example 4.6

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

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

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

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

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

$\sigma'' = \{x \mapsto y, z_1 \mapsto z_2, z_2 \mapsto z_1\}$ is an mgu of $x =? y$.

- ▶ $\sigma = \{z_1 \mapsto z_2, z_2 \mapsto z_1\}\sigma''$.
- ▶ σ'' is not idempotent.

Unification

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

Rule-based formulation of unification

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

The inference system \mathcal{U}

- ▶ A set of equations in **solved form**:

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

where each x_i occurs exactly once.

- ▶ For each idempotent substitution there exists exactly one set of equations in solved form.
- ▶ Notation:
 - ▶ $[\sigma]$ for the solved form set for an idempotent substitution σ .
 - ▶ σ_S for the idempotent substitution corresponding to a solved form set S .

The inference system \mathcal{U}

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

The inference system \mathcal{U}

Example 4.7

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

The inference system \mathcal{U}

Six transformation rules on systems:¹

Trivial:

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

Decomposition:

$$\begin{aligned} \{f(s_1, \dots, s_n) =^? f(t_1, \dots, t_n)\} \uplus P'; S \Leftrightarrow \\ \{s_1 =^? t_1, \dots, s_n =^? t_n\} \cup P'; S, \text{ where } n \geq 0. \end{aligned}$$

Symbol Clash:

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

¹ \uplus stands for disjoint union.

The inference system \mathcal{U}

Orient:

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

Occurs Check:

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

Variable Elimination:

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

if $x \notin \mathcal{Var}(t)$.

Unification with \mathcal{U}

In order to unify s and t :

1. Create an initial system $\{s \stackrel{?}{=} t\}; \emptyset$.
2. Apply successively rules from \mathcal{U} .

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

Examples

Example 4.8 (Failure)

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

$$\{p(f(a), g(x)) =^? p(y, y)\}; \emptyset \implies_{\text{Dec}}$$

$$\{f(a) =^? y, g(x) =^? y\}; \emptyset \implies_{\text{Or}}$$

$$\{y =^? f(a), g(x) =^? y\}; \emptyset \implies_{\text{VarEl}}$$

$$\{g(x) =^? f(a)\}; \{y \approx f(a)\} \implies_{\text{SymCl}}$$

\perp

Examples

Example 4.9 (Success)

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

$$\begin{aligned} &\{p(a, x, h(g(z))) =^? p(z, h(y), h(y))\}; \emptyset \implies_{\text{Dec}} \\ &\{a =^? z, x =^? h(y), h(g(z)) =^? h(y)\}; \emptyset \implies_{\text{Or}} \\ &\{z =^? a, x =^? h(y), h(g(z)) =^? h(y)\}; \emptyset \implies_{\text{VarEI}} \\ &\{x =^? h(y), h(g(a)) =^? h(y)\}; \{z \approx a\} \implies_{\text{VarEI}} \\ &\{h(g(a)) =^? h(y)\}; \{z \approx a, x \approx h(y)\} \implies_{\text{Dec}} \\ &\quad \{g(a) =^? y\}; \{z \approx a, x \approx h(y)\} \implies_{\text{Or}} \\ &\quad \{y =^? g(a)\}; \{z \approx a, x \approx h(y)\} \implies_{\text{VarEI}} \\ &\quad \emptyset; \{z \approx a, x \approx h(g(a)), y \approx g(a)\}. \end{aligned}$$

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

Examples

Example 4.10 (Failure)

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

$$\begin{aligned} \{p(x, x) =? p(y, f(y))\}; \emptyset &\Longrightarrow_{\text{Dec}} \\ \{x =? y, x =? f(y)\}; \emptyset &\Longrightarrow_{\text{VarEl}} \\ \{y =? f(y)\}; \{x \approx y\} &\Longrightarrow_{\text{OccCh}} \\ &\perp \end{aligned}$$

Properties of \mathcal{U} : termination

Lemma 4.3

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

$$P; \emptyset \Leftrightarrow P_1; S_1 \Leftrightarrow P_2; S_2 \Leftrightarrow \dots$$

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

Properties of \mathcal{L} : termination

Proof.

Complexity measure on the set P of equations: $\langle n_1, n_2, n_3 \rangle$, ordered lexicographically on triples of naturals, where

$n_1 =$ The number of distinct variables in P .

$n_2 =$ The number of symbols in P .

$n_3 =$ The number of equations in P of the form $t =^? x$ where t is not a variable.

Properties of \mathfrak{L} : termination

Proof [Cont.]

Each rule in \mathfrak{L} strictly reduces the complexity measure.

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

Properties of \mathcal{L} : termination

Proof [Cont.]

- ▶ A rule can always be applied to a system with non-empty P .
- ▶ The only systems to which no rule can be applied are \perp 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, \dots are in solved form.



Corollary 4.1

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

Properties of \mathcal{U} : correctness

Notation: Γ for systems.

Lemma 4.4

For any transformation $P; S \Leftrightarrow \Gamma$, a substitution ϑ unifies $P; S$ iff it unifies Γ .

Properties of \mathcal{U} : correctness

Proof.

Occurs Check: If $x \in \mathcal{V}ar(t)$ and $x \neq t$, then

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

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

Variable Elimination: From $\vartheta(x) = \vartheta(t)$, by structural induction on u :

$$\vartheta(u) = \vartheta\{x \mapsto t\}(u)$$

for any term, equation, or set of equations u . Then

$$\vartheta(P') = \vartheta\{x \mapsto t\}(P'), \quad \vartheta(S') = \vartheta\{x \mapsto t\}(S').$$



Properties of \mathcal{U} : correctness

Theorem 4.5 (Soundness)

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

Properties of \mathcal{U} : correctness

Theorem 4.5 (Soundness)

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

Proof.

By induction on the length of derivation, using the previous lemma and the fact that σ_S unifies S . □

Properties of \mathcal{L} : correctness

Theorem 4.6 (Completeness)

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

Properties of \mathcal{U} : correctness

Theorem 4.6 (Completeness)

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

Proof.

Such a sequence must end in $\emptyset; S$ where ϑ unifies S (why?). For every binding $x \mapsto t$ in σ_S , $\vartheta\sigma_S(x) = \vartheta(t) = \vartheta(x)$ and for every $x \notin \text{Dom}(\sigma_S)$, $\vartheta\sigma_S(x) = \vartheta(x)$. Hence, $\vartheta = \vartheta\sigma_S$. □

Properties of \mathcal{U} : correctness

Theorem 4.6 (Completeness)

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

Proof.

Such a sequence must end in $\emptyset; S$ where ϑ unifies S (why?). For every binding $x \mapsto t$ in σ_S , $\vartheta\sigma_S(x) = \vartheta(t) = \vartheta(x)$ and for every $x \notin \text{Dom}(\sigma_S)$, $\vartheta\sigma_S(x) = \vartheta(x)$. Hence, $\vartheta = \vartheta\sigma_S$. □

Corollary 4.2

If P has no unifiers, then any maximal sequence of transformations from $P; \emptyset$ must have the form $P; \emptyset \Leftrightarrow \dots \Leftrightarrow \perp$.

Observations

- ▶ \mathcal{U} computes an idempotent mgu.
- ▶ The choice of rules in computations via \mathcal{U} is “don’t care” nondeterminism (the word “any” in Completeness Theorem).
- ▶ Any control strategy will result to an mgu for unifiable terms, and failure for non-unifiable terms.
- ▶ Any practical algorithm that proceeds by performing transformations of \mathcal{U} in any order is
 - ▶ sound and complete,
 - ▶ generates mgus for unifiable terms.
- ▶ Not all transformation sequences have the same length.
- ▶ Not all transformation sequences end in exactly the same mgu.

Matching

Definition 4.5

Matcher, Matching Problem

- ▶ A substitution σ is a **matcher** of s to t if $\sigma(s) = t$.
- ▶ A matching equation between s and t is represented as $s \stackrel{?}{\sim} t$.
- ▶ A **matching problem** is a finite set of matching equations.

Matching vs unification

Example 4.11

$$f(x, y) \stackrel{?}{\sim} f(g(z), c)$$

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

$$f(x, y) \stackrel{?}{=} f(g(z), c)$$

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

Matching vs unification

Example 4.11

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

Matching vs unification

Example 4.11

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

Matching vs unification

Example 4.11

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

Matching vs unification

Example 4.11

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

How to solve matching problems

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

Matched form

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

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

The inference system \mathfrak{M}

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

The inference system \mathfrak{M}

Five transformation rules on matching systems:²

Decomposition:

$$\{f(s_1, \dots, s_n) \lesssim^? f(t_1, \dots, t_n)\} \uplus P'; S \Leftrightarrow \\ \{s_1 \lesssim^? t_1, \dots, s_n \lesssim^? t_n\} \cup P'; S, \text{ where } n \geq 0.$$

Symbol Clash:

$$\{f(s_1, \dots, s_n) \lesssim^? g(t_1, \dots, t_m)\} \uplus P'; S \Leftrightarrow \perp, \text{ if } f \neq g.$$

² \uplus stands for disjoint union.

The inference system \mathfrak{M}

Symbol-Variable Clash:

$$\{f(s_1, \dots, s_n) \lesssim^? x\} \uplus P'; S \Leftrightarrow \perp.$$

Merging Clash:

$$\{x \lesssim^? t_1\} \uplus P'; \{x \approx t_2\} \uplus S' \Leftrightarrow \perp, \text{ if } t_1 \neq t_2.$$

Elimination:

$$\{x \lesssim^? t\} \uplus P'; S \Leftrightarrow P'; \{x \approx t\} \cup S,$$

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

Matching with \mathfrak{M}

In order to match s to t

1. Create an initial system $\{s \stackrel{?}{\sim} t\}; \emptyset$.
2. Apply successively the rules from \mathfrak{M} .

Matching with \mathfrak{M}

Example 4.12

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

$$\{f(x, f(a, x)) \lesssim^? f(g(a), f(a, g(a)))\}; \emptyset \Leftrightarrow \text{Decomposition}$$

$$\{x \lesssim^? g(a), f(a, x) \lesssim^? f(a, g(a))\}; \emptyset \Leftrightarrow \text{Elimination}$$

$$\{f(a, x) \lesssim^? f(a, g(a))\}; \{x \approx g(a)\} \Leftrightarrow \text{Decomposition}$$

$$\{a \lesssim^? a, x \lesssim^? g(a)\}; \{x \approx g(a)\} \Leftrightarrow \text{Decomposition}$$

$$\{x \lesssim^? g(a)\}; \{x \approx g(a)\} \Leftrightarrow \text{Merge}$$

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

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

Matching with \mathfrak{M}

Example 4.13

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

$$\{f(x, x) \lesssim^? f(x, a)\}; \emptyset \Leftrightarrow \text{Decomposition}$$

$$\{x \lesssim^? x, x \lesssim^? a\}; \emptyset \Leftrightarrow \text{Elimination}$$

$$\{x \lesssim^? a\}; \{x \approx x\} \Leftrightarrow \text{Merging Clash}$$

\perp

No matcher.

Properties of \mathfrak{M} : termination

Theorem 4.7

For any finite set of matching problems P , every sequence of transformations in \mathfrak{M} of the form $P; \emptyset \Leftrightarrow P_1; S_1 \Leftrightarrow P_2; S_2 \Leftrightarrow \dots$ terminates either with \perp or with $\emptyset; S$, with S in matched form.

Properties of \mathfrak{M} : termination

Theorem 4.7

For any finite set of matching problems P , every sequence of transformations in \mathfrak{M} of the form $P; \emptyset \Leftrightarrow P_1; S_1 \Leftrightarrow P_2; S_2 \Leftrightarrow \dots$ terminates either with \perp or with $\emptyset; S$, with S in matched form.

Proof.

- ▶ Termination is obvious, since every rule strictly decreases the size of the first component of the matching system.
- ▶ A rule can always be applied to a system with non-empty P .
- ▶ The only systems to which no rule can be applied are \perp and $\emptyset; S$.
- ▶ Whenever $x \approx t$ is added to S , there is no other equation $x \approx t'$ in S . Hence, S_1, S_2, \dots are in matched form.



Properties of \mathfrak{M} : correctness

The following lemma is straightforward:

Lemma 4.5

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

Properties of \mathfrak{M} : correctness

Theorem 4.8 (Soundness)

If $P; \emptyset \Leftrightarrow^+ \emptyset; S$, then σ_S solves all matching equations in P .

Properties of \mathfrak{M} : correctness

Theorem 4.8 (Soundness)

If $P; \emptyset \Leftrightarrow^+ \emptyset; S$, then σ_S solves all matching equations in P .

Proof.

By induction on the length of derivations, using the previous lemma and the fact that σ_S solves the matching problems in S . □

Properties of \mathfrak{M} : correctness

Let $v(\{s_1 \approx t_1, \dots, s_n \approx t_n\})$ be $\mathcal{V}ar(\{s_1, \dots, s_n\})$.

Theorem 4.9 (Completeness)

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

Properties of \mathfrak{M} : correctness

Let $v(\{s_1 \approx t_1, \dots, s_n \approx t_n\})$ be $\text{Var}(\{s_1, \dots, s_n\})$.

Theorem 4.9 (Completeness)

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

Proof.

Such a sequence must end in $\emptyset; S$ where ϑ is a matcher of S .
 $v(S) = v(P)$. For every equation $x \approx t \in S$, either $t = x$ or $x \mapsto t \in \sigma_S$. Therefore, for any such x , $\sigma_S(x) = t = \vartheta(x)$. Hence, $\sigma_S = \vartheta|_{v(P)}$. □

Properties of \mathfrak{M} : correctness

Let $v(\{s_1 \approx t_1, \dots, s_n \approx t_n\})$ be $\mathcal{V}ar(\{s_1, \dots, s_n\})$.

Theorem 4.9 (Completeness)

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

Proof.

Such a sequence must end in $\emptyset; S$ where ϑ is a matcher of S . $v(S) = v(P)$. For every equation $x \approx t \in S$, either $t = x$ or $x \mapsto t \in \sigma_S$. Therefore, for any such x , $\sigma_S(x) = t = \vartheta(x)$. Hence, $\sigma_S = \vartheta|_{v(P)}$. □

Corollary 4.3

If P has no matchers, then any maximal sequence of transformations from $P; \emptyset$ must have the form $P; \emptyset \Leftrightarrow \dots \Leftrightarrow \perp$.