Rewriting

Part 3.2 Equational Problems. Syntactic Unification

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 $\to_E$ 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 decided $\approx_E$ 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 decided $\approx_E$ 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

$$
\begin{aligned}
f(x) =^? f(a) :&\quad \text{exactly one unifier } \{x \mapsto a\} \\
x =^? f(y) :&\quad \text{infinitely many unifiers} \\
&\qquad \{x \mapsto f(y)\}, \{x \mapsto f(a), y \mapsto a\}, \ldots \\
f(x) =^? g(y) :&\quad \text{no unifiers} \\
x =^? f(x) :&\quad \text{no unifiers}
\end{aligned}
$$

Examples

$x =^? f(y)$: infinitely many unifiers

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

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

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 3.2

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 3.2 (Variable Renaming)

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

Example 3.3

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

Substitutions

Definition 3.3 (Idempotent Substitution)

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

Example 3.4

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 3.2

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

Proof.

Exercise. □

Substitutions

Lemma 3.2

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

Proof.

Exercise. $\square$

Example 3.5

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

Substitutions

Theorem 3.4

σ is idempotent iff $\mathcal{D}om(\sigma) \cap \mathcal{VR}an(\sigma) = \emptyset$.

Proof.

Exercise. $\square$

Substitutions

Definition 3.4 (Unification Problem, Unifier, MGU)

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

Substitutions

Definition 3.4 (Unification Problem, Unifier, MGU)

- Unification problem: A finite set of equations $\Gamma = \{s_1 =^? t_1, \ldots, s_n =^? t_n\}$.
- 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 3.4 (Unification Problem, Unifier, MGU)

- Unification problem: A finite set of equations $\Gamma = \{s_1 =^? t_1, \ldots, s_n =^? t_n\}$.
- 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 3.4 (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 3.6

$\sigma := \{x \mapsto y\}$ 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.

Unifiers

Example 3.6

$\sigma := \{x \mapsto y\}$ 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 3.6

$\sigma := \{x \mapsto y\}$ 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 $\mathfrak{U}$

- A set of equations in solved form:

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

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

The Inference System $\mathfrak{U}$

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

The Inference System $\mathfrak{U}$

Example 3.7

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

The Inference System $\mathfrak{U}$

Six transformation rules on systems:[1]

Trivial:

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

Decomposition:

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

Symbol Clash:

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

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

The Inference System $\mathfrak{U}$

Orient:

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

Occurs Check:

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

Variable Elimination:

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

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

Unification with $\mathfrak{U}$

In order to unify s and t:

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

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

Properties of $\mathfrak{U}$: Termination

Lemma 3.3

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

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

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

Properties of $\mathfrak{U}$: 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 $\mathfrak{U}$: Termination

Proof [Cont.]

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

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

Properties of $\mathfrak{U}$: 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. $S_1, S_2, \ldots$ are in solved form.

□

Corollary 3.1

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

Properties of $\mathfrak{U}$: Correctness

Notation: Γ for systems.

Lemma 3.4

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

Properties of $\mathfrak{U}$: 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

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

□

Properties of $\mathfrak{U}$: Correctness

Theorem 3.5 (Soundness)

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

Properties of $\mathfrak{U}$: Correctness

Theorem 3.5 (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 $\mathfrak{U}$: Correctness

Theorem 3.6 (Completeness)

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

Properties of $\mathfrak{U}$: Correctness

Theorem 3.6 (Completeness)

If ϑ unifies every equation in P, then any maximal sequence of transformations $P; \emptyset \Leftrightarrow \cdots$ 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 $x \notin \mathcal{D}om(\sigma_S)$, $\vartheta\sigma_S(x) = \vartheta(x)$. Hence, $\vartheta = \vartheta\sigma_S$. □

Properties of $\mathfrak{U}$: Correctness

Theorem 3.6 (Completeness)

If ϑ unifies every equation in P, then any maximal sequence of transformations $P; \emptyset \Leftrightarrow \cdots$ 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 $x \notin \mathcal{D}om(\sigma_S)$, $\vartheta\sigma_S(x) = \vartheta(x)$. Hence, $\vartheta = \vartheta\sigma_S$. □

Corollary 3.2

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

Observations

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

Matching

Definition 3.5

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 $s \lesssim^? t$.
- A matching problem is a finite set of matching equations.

Matching vs Unification

Example 3.8

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

Matching vs Unification

Example 3.8

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

Matching vs Unification

Example 3.8

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

Matching vs Unification

Example 3.8

$f(x,y) \lesssim^? f(g(z),c)$	$f(x,y) =^? f(g(z),c)$
$\{x \mapsto g(z), y \mapsto c\}$	$\{x \mapsto g(z), y \mapsto c\}$
$f(x,y) \lesssim^? f(g(z),x)$	$f(x,y) =^? f(g(z),x)$
$\{x \mapsto g(z), y \mapsto x\}$	$\{x \mapsto g(z), y \mapsto g(z)\}$
$f(x,a) \lesssim^? f(b,y)$	$f(x,a) =^? f(b,y)$
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 3.8

$f(x,y) \lesssim^? f(g(z),c)$	$f(x,y) =^? f(g(z),c)$
$\{x \mapsto g(z), y \mapsto c\}$	$\{x \mapsto g(z), y \mapsto c\}$
$f(x,y) \lesssim^? f(g(z),x)$	$f(x,y) =^? f(g(z),x)$
$\{x \mapsto g(z), y \mapsto x\}$	$\{x \mapsto g(z), y \mapsto g(z)\}$
$f(x,a) \lesssim^? f(b,y)$	$f(x,a) =^? f(b,y)$
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 $\mathfrak{U}$ to work directly with the matching problem.

Matched Form

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

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

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.

Five transformation rules on matching systems:[2]

Decomposition:

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

Symbol Clash:

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

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

The Inference System $\mathfrak{M}$

Symbol-Variable Clash:

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

Merging Clash:

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

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 $\{s \lesssim^? t\}; \emptyset$.
2. Apply successively the rules from $\mathfrak{M}$.

Matching with $\mathfrak{M}$

Example 3.9

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 3.10

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

$$\{f(x, x) \lesssim^? f(x, a)\}; \emptyset \Leftrightarrow_{\text{Decomposition}}$$
$$\{x \lesssim^? x, x \lesssim^? a\}; \emptyset \Leftrightarrow_{\text{Elimination}}$$
$$\{x \lesssim^? a\}; \{x \approx x\} \Leftrightarrow_{\text{Merging Clash}}$$
$$\bot$$

No matcher.

Properties of $\mathfrak{M}$: Termination

Theorem 3.7

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

Properties of $\mathfrak{M}$: Termination

Theorem 3.7

For any finite set of matching problems P, every sequence of transformations in $\mathfrak{M}$ of the form $P; \emptyset \Leftrightarrow P_1; S_1 \Leftrightarrow P_2; S_2 \Leftrightarrow \cdots$ terminates either with $\bot$ 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 $\bot$ and $\emptyset; S$.
- Whenever $x \approx t$ is added to S, there is no other equation $x \approx t'$ in S. Hence, $S_1, S_2, \ldots$ are in matched form.

□

Properties of $\mathfrak{M}$: Correctness

The following lemma is straightforward:

Lemma 3.5
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 3.8 (Soundness)

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

Properties of $\mathfrak{M}$: Correctness

Theorem 3.8 (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 $v(\{s_1 \approx t_1, \ldots, s_n \approx t_n\})$ be $\mathcal{V}ar(\{s_1, \ldots, s_n\})$.

Theorem 3.9 (Completeness)

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

Properties of $\mathfrak{M}$: Correctness

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

Theorem 3.9 (Completeness)

If ϑ is a matcher of P, then any maximal sequence of transformations $P; \emptyset \Leftrightarrow \cdots$ 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 $v(\{s_1 \approx t_1, \ldots, s_n \approx t_n\})$ be $\mathcal{V}ar(\{s_1, \ldots, s_n\})$.

Theorem 3.9 (Completeness)

If ϑ is a matcher of P, then any maximal sequence of transformations $P; \emptyset \Leftrightarrow \cdots$ 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 3.3

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