

# Logic Programming

## Unification

Temur Kutsia

Research Institute for Symbolic Computation  
Johannes Kepler University, Linz, Austria  
`kutsia@risc.jku.at`

# Unification

Solving term equations:

Given: Two terms  $s$  and  $t$ .

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

# Substitutions

- ▶ A  $T(\mathcal{F}, \mathcal{V})$ -**substitution**: A function  $\sigma : \mathcal{V} \rightarrow T(\mathcal{F}, \mathcal{V})$ , whose **domain**

$$\text{Dom}(\sigma) := \{x \mid \sigma(x) \neq x\}$$

is finite.

- ▶ **Range** of a substitution  $\sigma$ :

$$\mathcal{Ran}(\sigma) := \{\sigma(x) \mid x \in \text{Dom}(\sigma)\}.$$

- ▶ **Variable range** of a substitution  $\sigma$ :

$$\mathcal{VRan}(\sigma) := \text{Var}(\mathcal{Ran}(\sigma)).$$

- ▶ Notation: lower case Greek letters  $\sigma, \vartheta, \varphi, \psi, \dots$   
Identity substitution:  $\varepsilon$ .

# Substitutions

- Notation: If  $\text{Dom}(\sigma) = \{x_1, \dots, x_n\}$ , then  $\sigma$  can be written as the set

$$\{x_1 \mapsto \sigma(x_1), \dots, x_n \mapsto \sigma(x_n)\}.$$

- Example:

$$\{x \mapsto i(y), y \mapsto e\}.$$

# Substitutions

- The substitution  $\sigma$  can be extended to a mapping

$$\sigma : T(\mathcal{F}, \mathcal{V}) \rightarrow T(\mathcal{F}, \mathcal{V})$$

by induction:

$$\sigma(f(t_1, \dots, t_n)) = f(\sigma(t_1), \dots, \sigma(t_n)).$$

- Example:

$$\sigma = \{x \mapsto i(y), y \mapsto e\}.$$

$$t = f(y, f(x, y))$$

$$\sigma(t) = f(e, f(i(y), e))$$

- $\mathcal{S}ub$  : The set of substitutions.

# More Notions about Substitutions

- **Composition** of  $\vartheta$  and  $\sigma$ :

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

- Composition of two substitutions is again a substitution.
- Composition is associative but not commutative.

# More Notions about Substitutions

Algorithm for obtaining a set representation of a composition of two substitutions in a set form.

► Given:

$$\theta = \{x_1 \mapsto t_1, \dots, x_n \mapsto t_n\}$$

$$\sigma = \{y_1 \mapsto s_1, \dots, y_m \mapsto s_m\},$$

the set representation of their composition  $\sigma\theta$  is obtained from the set

$$\{x_1 \mapsto \sigma(t_1), \dots, x_n \mapsto \sigma(t_n), y_1 \mapsto s_1, \dots, y_m \mapsto s_m\}$$

by deleting

- all  $y_i \mapsto s_i$ 's with  $y_i \in \{x_1, \dots, x_n\}$ ,
- all  $x_i \mapsto \sigma(t_i)$ 's with  $x_i = \sigma(t_i)$ .

# More Notions about Substitutions

## Example (Composition)

$$\theta = \{x \mapsto f(y), y \mapsto z\}.$$

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

$$\sigma\theta = \{x \mapsto f(b), z \mapsto y\}.$$



# More Notions about Substitutions

- ▶  $t$  is an **instance** of  $s$  iff there exists a  $\sigma$  such that

$$\sigma(s) = t.$$

- ▶ Notation:  $t \succeq s$  (or  $s \preceq t$ ).
- ▶ Reads:  $t$  is more specific than  $s$ , or  $s$  is more general than  $t$ .
- ▶  $\succeq$  is a quasi-order.
- ▶ Strict part:  $>$ .
- ▶ Example:  $f(e, f(i(y), e)) \succeq f(y, f(x, y))$ , because

$$\sigma(f(y, f(x, y))) = f(e, f(i(y), e))$$

for  $\sigma = \{x \mapsto i(y), y \mapsto e\}$

# Unification

Syntactic unification:

Given: Two terms  $s$  and  $t$ .

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

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

# Examples

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

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

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

$f(x) =^? g(y) :$  no unifiers

$x =^? f(x) :$  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  $\sigma$  is **more general** than  $\vartheta$ , written  $\sigma \lesssim \vartheta$ , if there exists  $\eta$  such that  $\eta\sigma = \vartheta$ .
- ▶  $\vartheta$  is called an **instance** of  $\sigma$ .
- ▶ 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

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 (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

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

# Substitutions

## Definition (Idempotent Substitution)

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

## Example

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

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

# Substitutions

## Lemma

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

## Proof.

Exercise. □

## Example

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



# Substitutions

## Theorem

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

Proof.

Exercise. □

# Substitutions

## Definition (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  $\Gamma$ : A substitution  $\sigma$  such that  $\sigma(s_i) = \sigma(t_i)$  for all  $1 \leq i \leq n$ .
- ▶  $\mathcal{U}(\Gamma)$ : The set of all unifiers of  $\Gamma$ .  $\Gamma$  is **unifiable** iff  $\mathcal{U}(\Gamma) \neq \emptyset$ .
- ▶  $\sigma$  is a **most general unifier (mgu)** of  $\Gamma$  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

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

For any other unifier  $\vartheta$  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  $x =^? y$ .

- ▶  $\sigma = \{z_1 \mapsto z_2, z_2 \mapsto z_1\}\sigma''$ .
- ▶  $\sigma''$  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  $\sigma$ .
  - ▶  $\sigma_S$  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

- ▶ 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 $\mathcal{U}$

Six transformation rules on systems:<sup>1</sup>

**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:**

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

---

<sup>1</sup> $\uplus$  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 $\mathfrak{U}$

In order to unify  $s$  and  $t$ :

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

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

# Examples

## Example (Failure)

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

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

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

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

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

$\perp$

# Examples

## Example (Success)

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

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

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

# Examples

## Example (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 $\mathfrak{U}$ : Termination

## Lemma

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

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

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

# Properties of $\mathcal{U}$ : 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 $\mathcal{U}$ : Termination

## Proof [Cont.]

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

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

# Properties of $\mathcal{U}$ : 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.  $S_1, S_2, \dots$  are in solved form.



## Corollary

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

## Properties of $\mathcal{U}$ : Correctness

Notation:  $\Gamma$  for systems.

### Lemma

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

# Properties of $\mathcal{U}$ : Correctness

## Proof.

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

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

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 (Soundness)

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

### Proof.

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

# Properties of $\mathcal{U}$ : Correctness

## Theorem (Completeness)

*If  $\vartheta$  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  $\vartheta$  unifies  $S$  (why?).  
For every binding  $x \mapsto t$  in  $\sigma_S$ ,  $\vartheta\sigma_S(x) = \vartheta(t) = \vartheta(x)$  and for every  $x \notin \text{Dom}(\sigma_S)$ ,  $\vartheta\sigma_S(x) = \vartheta(x)$ . Hence,  $\vartheta = \vartheta\sigma_S$ . □

### Corollary

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

# 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.

## Example 3.10 in Prolog

Recall: Unification algorithm fails for  $p(x, x) \stackrel{?}{=} p(y, f(y))$  because of the occurrence check.

But Prolog behaves differently:

### Example (Infinite Terms)

?-  $p(X, X) = p(Y, f(Y))$ .

$X = f(**), Y = f(**)$ .

In some versions of Prolog output looks like this:

$X = f(f(f(f(f(f(f(f(f(f(\dots))))))))))$

$Y = f(f(f(f(f(f(f(f(f(f(\dots))))))))))$



## Occurrence Check

Prolog unification algorithm skips Occurrence Check.

Reason: Occurrence Check can be expensive.

Justification: Most of the time this rule is not needed.

Drawback: Sometimes might lead to unexpected answers.

# Occurrence Check

## Example

```
less(X,s(X)).
```

```
foo:-less(s(Y),Y).
```

```
?- foo.
```

Yes