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** σ such that $\sigma(s) = \sigma(t)$.

Substitutions

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

  $$\mathcal{D}om(\sigma) := \{x \mid \sigma(x) \neq x\}$$

  is finite.
- Range of a substitution σ:

  $$\mathcal{R}an(\sigma) := \{\sigma(x) \mid x \in \mathcal{D}om(\sigma)\}.$$

- Variable range of a substitution σ:

  $$\mathcal{V}\mathcal{R}an(\sigma) := \mathcal{V}ar(\mathcal{R}an(\sigma)).$$

- Notation: lower case Greek letters $\sigma, \vartheta, \varphi, \psi, \ldots$.
  Identity substitution: ε.

Substitutions

- Notation: If $\mathcal{D}om(\sigma) = \{x_1, \ldots, x_n\}$, then σ can be written as the set

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

- Example:

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

Substitutions

- The substitution σ can be extended to a mapping

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

  by induction:

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

- Example:

$$\begin{aligned} \sigma &= \{x \mapsto i(y), y \mapsto e\}. \\ t &= f(y, f(x, y)) \\ \sigma(t) &= f(e, f(i(y), e)) \end{aligned}$$

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

More Notions about Substitutions

- Composition of ϑ and σ:

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

- 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 3.1 (Composition)

$$\begin{aligned} \theta &= \{x \mapsto f(y), y \mapsto z\}. \\ \sigma &= \{x \mapsto a, y \mapsto b, z \mapsto y\}. \\ \sigma\theta &= \{x \mapsto f(b), z \mapsto y\}. \end{aligned}$$

More Notions about Substitutions

- t is an instance of s iff there exists a σ such that

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

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

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

- 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. $\square$

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 $\Gamma = \{s_1 =^? t_1, \ldots, s_n =^? t_n\}$.
- 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.

$\sigma' := \{x \mapsto z, y \mapsto z\}$ 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 $\perp$ 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 $\mathfrak{U}$

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.

Examples

Example 3.8 (Failure)

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

$$\begin{aligned}
\{p(f(a), g(x)) =^? p(y, y)\};\ \emptyset &\Longrightarrow_{\mathsf{Dec}}\\
\{f(a) =^? y, g(x) =^? y\};\ \emptyset &\Longrightarrow_{\mathsf{Or}}\\
\{y =^? f(a), g(x) =^? y\};\ \emptyset &\Longrightarrow_{\mathsf{VarEl}}\\
\{g(x) =^? f(a)\};\ \{y \approx f(a)\} &\Longrightarrow_{\mathsf{SymCl}}\\
&\bot
\end{aligned}$$

Examples

Example 3.9 (Success)

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

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

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

Examples

Example 3.10 (Failure)

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

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

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 $t \doteq^? x$ 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 $\bot$ 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. $S_1, S_2, \dots$ are in solved form.

$\square$

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.

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 $\sigma_S \lesssim \vartheta$.

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, $\vartheta = \vartheta\sigma_S$. $\square$

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.

Example 3.10 in Prolog

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

But Prolog behaves differently:

Example 3.11 (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(...))))))))))

Y = f(f(f(f(f(f(f(f(f(f(...))))))))))
```

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 1

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

?- foo.

Yes
```