Logic Programming

Unification

Temur Kutsia

Research Institute for Symbolic Computation
Johannes Kepler University of Linz, Austria
kutsia@risc.uni-linz.ac.at

Contents

Unification

Unification algorithm: The heart of the computation model of logic programs.

Substitution

Definition (Substitution)

A *substitution* is a finite set of the form

$$\theta = \{v_1 \mapsto t_1, \ldots, v_n \mapsto t_n\}$$

- v_i's: distinct variables.
- t_i's: terms with $t_i \neq v_i$.
- Binding: $v_i \mapsto t_i$.

Substitution Application

Definition (Substitution application)

Substitution $\theta = \{v_1 \mapsto t_1, \ldots, v_n \mapsto t_n\}$ applied to an expression E,

$$E\theta$$

Simultaneously replacing each occurrence of v_i in E with t_i.

$E\theta$ is called the *instance* of E wrt θ.

E_1 is more general than E_2 if E_2 is an instance of E_1 (wrt some substitution).

Substitution Application

Example (Application)

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

Note that x was not replaced second time.

Composition

Definition (Substitution Composition)

Given two substitutions

$$\theta = \{v_1 \mapsto t_1, \ldots, v_n \mapsto t_n\}$$
$$\sigma = \{u_1 \mapsto s_1, \ldots, u_m \mapsto s_m\},$$

their *composition* $\theta\sigma$ is obtained from the set

$$\{v_1 \mapsto t_1\sigma, \ldots, v_n \mapsto t_n\sigma,$$
$$u_1 \mapsto s_1, \ldots, u_m \mapsto s_m\}$$

by deleting

- all $u_i \mapsto s_i$'s with $u_i \in \{v_1, \ldots, v_n\}$,
- all $v_i \mapsto t_i\sigma$'s with $v_i = t_i\sigma$.

Substitution Composition

Example (Composition)

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

Empty Substitution

Empty substitution, denoted ε:

- Empty set of bindings.
- $E\varepsilon = E$ for all expressions E.

Properties

Theorem

$$\theta\varepsilon = \varepsilon\theta = \theta.$$
$$(E\theta)\sigma = E(\theta\sigma).$$
$$(\theta\sigma)\lambda = \theta(\sigma\lambda).$$

Example (Properties)

Example

Given:

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

Then

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

Renaming Substitution

Definition (Renaming Substitution)

$\{x_1 \mapsto y_1, \ldots, x_n \mapsto y_n\}$ is a *renaming substitution* iff y_i's are distinct variables.

Renaming an Expression

Definition (Renaming Substitution for an Expression)

Let V be the set of variables of an expression E.

A substitution

$$\theta = \{x_1 \mapsto y_1, \ldots, x_n \mapsto y_n\}$$

is a *renaming substitution for E* iff

- θ is a renaming substitution, and
- $\{x_1, \ldots, x_n\} \subseteq V$, and
- $(V \setminus \{x_1, \ldots, x_n\}) \cap \{y_1, \ldots, y_n\} = \emptyset$.

Renaming an Expression

Example

- $E = f(x, a, y, z)$
- $\sigma_1 = \{x \mapsto u_1, y \mapsto u_2, z \mapsto u_3\}$ is a renaming subst. for E.
- $\sigma_2 = \{x \mapsto u_1, y \mapsto u_2\}$ is a renaming subst. for E.
- $\sigma_3 = \{x \mapsto y, y \mapsto x, z \mapsto u\}$ is a renaming subst. for E.
- $\sigma_4 = \{x \mapsto y, z \mapsto u\}$ is not a renaming subst. for E.
- $\sigma_5 = \{x \mapsto u, y \mapsto u, z \mapsto u\}$ is not a renaming subst.

Variants

Definition (Variant)

Expression E and expression F are *variants* iff there exist substitutions θ and σ such that

- $E\theta = F$ and
- $F\sigma = E$.

Variants and Renaming

Theorem

Expression E and expression F are variants iff there exist *renaming* *substitutions* θ *and* σ *such that*

- $E\theta = F$ *and*
- $F\sigma = E$.

Instantiation Quasi-Ordering

Definition (More General Substitution)

A substitution θ is *more general* than a substitution σ, written $\theta \leq \sigma$, iff there exists a substitution λ such that

$$\theta\lambda = \sigma.$$

The relation $\leq$ on substitutions is called the *instantiation quasi-ordering*.

Instantiation Quasi-Ordering

Example (More General)

Let θ and σ be the substitutions:

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

Then $\theta \leq \sigma$ because $\theta\lambda = \sigma$ where

$$\lambda = \{y \mapsto z\}.$$

Unifier

Definition (Unifier of Expressions)

A substitution θ is a *unifier* of expressions E and F iff

$$E\theta = F\theta.$$

Unifier

Example (Unifier of Expressions)

Let E and F be two expressions:

$$E = f(x, b, g(z)),$$
$$F = f(f(y), y, g(u)).$$

Then $\theta = \{x \mapsto f(b), y \mapsto b, z \mapsto u\}$ is a unifier of E and F:

$$E\theta = f(f(b), b, g(u)),$$
$$F\theta = f(f(b), b, g(u)).$$

Unification Problem, Unifier

Definition (Unification Problem)

Unification problem is a finite set of equations (expression pairs).

Definition (Unifier)

σ is a *unifier of a unification problem*

$$\{E_1 \stackrel{?}{=} F_1, \dots, E_n \stackrel{?}{=} F_n\}$$

iff σ is a unifier of E_i and F_i for each $1 \leq i \leq n$, i.e., iff

$$E_1\sigma = F_1\sigma,$$
$$\cdots,$$
$$E_n\sigma = F_n\sigma$$

Most General Unifier

Definition (MGU)

A unifier θ of E and F is *most general* iff θ is more general than any other unifier of E and F.

Unifiers and MGU

Example (Unifiers)

Let E and F be two expressions:

$$E = f(x, b, g(z)),$$
$$F = f(f(y), y, g(u)).$$

Unifiers of E and F (infinitely many):

$$\theta_1 = \{x \mapsto f(b), y \mapsto b, z \mapsto u\},$$
$$\theta_2 = \{x \mapsto f(b), y \mapsto b, u \mapsto z\},$$
$$\theta_3 = \{x \mapsto f(b), y \mapsto b, z \mapsto a, u \mapsto a\},$$
$$\theta_4 = \{x \mapsto f(b), y \mapsto b, z \mapsto u, w \mapsto d\},$$
$$\ldots$$

Unifiers and MGU

Example (MGU)

Let E and F be expressions from the previous example:

$$E = f(x, b, g(z)),\ F = f(f(y), y, g(u)).$$

MGU's of E and F:

$$\theta_1 = \{x \mapsto f(b), y \mapsto b, z \mapsto u\},$$
$$\theta_2 = \{x \mapsto f(b), y \mapsto b, u \mapsto z\}.$$

$\theta_1 \leq \theta_2$: $\theta_2 = \theta_1\lambda_1$ with $\lambda_1 = \{u \mapsto z\}$.
$\theta_2 \leq \theta_1$: $\theta_1 = \theta_2\lambda_2$ with $\lambda_2 = \{z \mapsto u\}$.

Note: λ_1 and λ_2 are renaming substitutions.

Equivalence of mgu-s

Theorem

Most general unifier of two expressions is unique up to variable renaming

Unification Algorithm

Rule-based approach.

- General form of rules:

$$P;\ \sigma \Longrightarrow Q;\ \theta \text{ or}$$
$$P;\ \sigma \Longrightarrow \bot.$$

- $\bot$ denotes failure.
- σ and θ are substitutions.
- P and Q are unification problems: $\{E_1 \stackrel{?}{=} F_1, \ldots, E_n \stackrel{?}{=} F_n\}$.

Unification Rules

Trivial:

$$\{s \stackrel{?}{=} s\} \cup P';\ \sigma \Longrightarrow P';\ \sigma.$$

Decomposition:

$$\{f(s_1, \ldots, s_n) \stackrel{?}{=} f(t_1, \ldots, t_n)\} \cup P';\ \sigma \Longrightarrow \{s_1 \stackrel{?}{=} t_1, \ldots, s_n \stackrel{?}{=} t_n\} \cup P';\ \sigma.$$

if $f(s_1, \ldots, s_n) \neq f(t_1, \ldots, t_n)$.

Symbol Clash:

$$\{f(s_1, \ldots, s_n) \stackrel{?}{=} g(t_1, \ldots, t_m)\} \cup P';\ \sigma \Longrightarrow \bot.$$

if $f \neq g$.

Unification Rules (Contd.)

Orient:

$$\{t \stackrel{?}{=} x\} \cup P';\ \sigma \Longrightarrow \{x \stackrel{?}{=} t\} \cup P';\ \sigma,$$

if t is not a variable.

Occurs Check:

$$\{x \stackrel{?}{=} t\} \cup P';\ \sigma \Longrightarrow \bot,$$

if x occurs in t and $x \neq t$.

Variable Elimination:

$$\{x \stackrel{?}{=} t\} \cup P';\ \sigma \Longrightarrow P'\theta;\ \sigma\theta,$$

if x does not occur in t, and $\theta = \{x \mapsto t\}$.

Unification Algorithm

In order to unify expressions E_1 and E_2:

1. Create initial system $\{E_1 \stackrel{?}{=} E_2\}; \varepsilon$.
2. Apply successively unification rules.

Termination

Theorem (Termination)

The unification algorithm terminates either with $\bot$ or with $\emptyset; \sigma$.

Soundness

Theorem (Soundness)

If $P;\ \varepsilon \Longrightarrow^{+} \emptyset;\ \sigma$ *then* σ *is a unifier of* P.

Completeness

Theorem (Completeness)

For any unifier θ of P the unification algorithm finds a unifier σ of P such that $\sigma \leq \theta$.

Major Result

Theorem (Main Theorem)

If two expressions are unifiable then the unification algorithm computes their MGU.

Examples

Example (Failure)

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

$$\begin{aligned}
\{p(f(a), g(x)) \stackrel{?}{=} p(y, y)\};\ \varepsilon &\Longrightarrow_{\text{Dec}} \\
\{f(a) \stackrel{?}{=} y, g(x) \stackrel{?}{=} y\};\ \varepsilon &\Longrightarrow_{\text{Or}} \\
\{y \stackrel{?}{=} f(a), g(x) \stackrel{?}{=} y\};\ \varepsilon &\Longrightarrow_{\text{VarEl}} \\
\{g(x) \stackrel{?}{=} f(a)\};\ \{y \mapsto f(a)\} &\Longrightarrow_{\text{SymCl}} \\
\bot &
\end{aligned}$$

Examples

Example (Success)

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

$$\{p(a, x, h(g(z))) \stackrel{?}{=} p(z, h(y), h(y))\};\ \varepsilon \Longrightarrow_{\text{Dec}}$$
$$\{a \stackrel{?}{=} z, x \stackrel{?}{=} h(y), h(g(z)) \stackrel{?}{=} h(y)\};\ \varepsilon \Longrightarrow_{\text{Or}}$$
$$\{z \stackrel{?}{=} a, x \stackrel{?}{=} h(y), h(g(z)) \stackrel{?}{=} h(y)\};\ \varepsilon \Longrightarrow_{\text{VarEl}}$$
$$\{x \stackrel{?}{=} h(y), h(g(a)) \stackrel{?}{=} h(y)\};\ \{z \mapsto a\} \Longrightarrow_{\text{VarEl}}$$
$$\{h(g(a)) \stackrel{?}{=} h(y)\};\ \{z \mapsto a, x \mapsto h(y)\} \Longrightarrow_{\text{Dec}}$$
$$\{g(a) \stackrel{?}{=} y\};\ \{z \mapsto a, x \mapsto h(y)\} \Longrightarrow_{\text{Or}}$$
$$\{y \stackrel{?}{=} g(a)\};\ \{z \mapsto a, x \mapsto h(y)\} \Longrightarrow_{\text{VarEl}}$$
$$\emptyset;\ \{z \mapsto a, x \mapsto h(g(a)), y \mapsto g(a)\}.$$

Examples

Example (Failure)

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

$$
\begin{aligned}
\{p(x, x) \stackrel{?}{=} p(y, f(y))\};\ \varepsilon &\Longrightarrow_{\text{Dec}} \\
\{x \stackrel{?}{=} y, x \stackrel{?}{=} f(y)\};\ \varepsilon &\Longrightarrow_{\text{VarEl}} \\
\{y \stackrel{?}{=} f(y)\};\ \{x \mapsto y\} &\Longrightarrow_{\text{OccCh}} \\
&\bot
\end{aligned}
$$

Previous Example on PROLOG

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(...))))))))))

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

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

?- foo.

Yes
```