Logic Programming
Unification

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

Substitutions

Unifiers

Unification Algorithm
Unification

Unification algorithm: The heart of the computation model of logic programs.
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: \( v_i \mapsto t_i \).
Definition (Substitution application)

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

$$E\theta$$

(the instance of $E$ wrt $\theta$): Simultaneously replacing each occurrence of $v_i$ in $E$ with $t_i$. 
Substitution Application

Example (Application)

\[ E = p(x, y, f(a)). \]
\[ \theta = \{ x \mapsto b, y \mapsto x \}. \]
\[ E\theta = p(b, x, f(a)). \]

Note that \( x \) was not replaced second time.
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)

\[ \theta = \{ x \mapsto f(y), y \mapsto z \}. \]
\[ \sigma = \{ x \mapsto a, y \mapsto b, z \mapsto y \}. \]
\[ \theta \sigma = \{ x \mapsto f(b), z \mapsto y \}. \]
Empty Substitution

Empty substitution, denoted $\varepsilon$:
- 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
\begin{itemize}
  \item $\theta$ is a renaming substitution, and
  \item $\{x_1, \ldots, x_n\} \subseteq V$, and
  \item $(V \setminus \{x_1, \ldots, x_n\}) \cap \{y_1, \ldots, y_n\} = \emptyset$.
\end{itemize}
Definition (Variant)

Expression $E$ and expression $F$ are *variants* iff there exist substitutions $\theta$ and $\sigma$ such that

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

Expression $E$ and expression $F$ are variants iff there exist renaming substitutions $\theta$ and $\sigma$ such that

- $E\theta = F$ and
- $F\sigma = E$. 
Definition (More General Substitution)
A substitution $\theta$ is *more general* than a substitution $\sigma$, written $\theta \leq \sigma$, iff there exists a substitution $\lambda$ such that

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

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

Example (More General)
Let \( \theta \) and \( \sigma \) 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

\[
\lambda = \{ y \mapsto z \}.
\]
Definition (Unifier of Expressions)

A substitution $\theta$ is a *unifier* of expressions $E$ and $F$ iff

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

$$E\theta = f(f(b), b, g(u)),$$
$$F\theta = f(f(b), b, g(u)).$$
Definition (Unifier of a Set of Expression Pairs)

\( \sigma \) is a **unifier of a set of expression pairs**

\[
\{ \langle E_1, F_1 \rangle, \ldots, \langle E_n, F_n \rangle \}
\]

iff \( \sigma \) is a unifier of \( E_i \) and \( F_i \) for each \( 1 \leq i \leq n \), i.e., iff

\[
E_1\sigma = F_1\sigma, \\
\ldots, \\
E_n\sigma = F_n\sigma
\]
Most General Unifier

Definition (MGU)

A unifier $\theta$ of $E$ and $F$ is *most general* iff $\theta$ 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:

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

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: $\lambda_1$ and $\lambda_2$ 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:

  $P; \sigma \rightarrow Q; \theta$ or
  $P; \sigma \rightarrow \bot$.

- $\bot$ denotes failure.

- $\sigma$ and $\theta$ are substitutions.

- $P$ and $Q$ are sets of expression pairs:
  $\{\langle E_1, F_1 \rangle, \ldots, \langle E_n, F_n \rangle\}$. 
Unification Rules

**Trivial:**

\[ \{ \langle s, s \rangle \} \cup P'; \sigma \Rightarrow P'; \sigma. \]

**Decomposition:**

\[ \{ \langle f(s_1, \ldots, s_n), f(t_1, \ldots, t_n) \rangle \} \cup P'; \sigma \Rightarrow \{ \langle s_1, t_1 \rangle, \ldots, \langle s_n, t_n \rangle \} \cup P'; \sigma. \]

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

**Symbol Clash:**

\[ \{ \langle f(s_1, \ldots, s_n), g(t_1, \ldots, t_m) \rangle \} \cup P'; \sigma \Rightarrow \bot. \]

if \( f \neq g \).
Unification Rules (Contd.)

Orient:
$$\{\langle t, x \rangle \} \cup P'; \sigma \implies \{\langle x, t \rangle \} \cup P'; \sigma,$$

if $t$ is not a variable.

Occurs Check:
$$\{\langle x, t \rangle \} \cup P'; \sigma \implies \bot,$$

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

Variable Elimination:
$$\{\langle x, t \rangle \} \cup P'; \sigma \implies P'; \sigma' \theta,$$

if $x$ does not occur in $t$, and $\theta = \{x \mapsto t\}$. 
In order to unify expressions $E_1$ and $E_2$:

1. Create initial system $\{\langle E_1, E_2 \rangle \}; \varepsilon$.
2. Apply successively unification rules.
Termination

Theorem (Termination)

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

Theorem (Soundness)

If $P; \varepsilon \xrightarrow{+} \emptyset; \sigma$ then $\sigma$ is a unifier of $P$. 
Theorem (Completeness)

For any unifier $\theta$ of $P$ the unification algorithm finds a unifier $\sigma$ of $P$ such that $\sigma \leq \theta$. 
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 $p(y, y)$.

$$\{\langle p(f(a), g(x)), p(y, y) \rangle \}; \quad \varepsilon \mapsto_{\text{Dec}}$$

$$\{\langle f(a), y \rangle, \langle g(x), y \rangle \}; \quad \varepsilon \mapsto_{\text{Or}}$$

$$\{\langle y, f(a) \rangle, \langle g(x), y \rangle \}; \quad \varepsilon \mapsto_{\text{VarEl}}$$

$$\{\langle g(x), f(a) \rangle \}; \quad \{ y \mapsto f(a) \} \mapsto_{\text{SymCl}}$$
Examples

Example (Success)

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

\[
\begin{align*}
\{\langle p(a, x, h(g(z))), p(z, h(y), h(y)) \rangle \}; & \quad \varepsilon \quad \Rightarrow \text{Dec} \\
\{\langle a, z \rangle, \langle x, h(y) \rangle, \langle h(g(z)), h(y) \rangle \}; & \quad \varepsilon \quad \Rightarrow \text{Or} \\
\{\langle z, a \rangle, \langle x, h(y) \rangle, \langle h(g(z)), h(y) \rangle \}; & \quad \varepsilon \quad \Rightarrow \text{VarEl} \\
\{\langle x, h(y) \rangle, \langle h(g(a)), h(y) \rangle \}; & \quad \{ z \mapsto a \} \quad \Rightarrow \text{VarEl} \\
\{\langle h(g(a)), h(y) \rangle \}; & \quad \{ z \mapsto a, x \mapsto h(y) \} \quad \Rightarrow \text{Dec} \\
\{\langle g(a), y \rangle \}; & \quad \{ z \mapsto a, x \mapsto h(y) \} \quad \Rightarrow \text{Or} \\
\{\langle y, g(a) \rangle \}; & \quad \{ z \mapsto a, x \mapsto h(y) \} \quad \Rightarrow \text{VarEl} \\
\emptyset; & \quad \{ z \mapsto a, x \mapsto h(g(a)), y \mapsto g(a) \}.
\end{align*}
\]
Examples

Example (Failure)
Unify \( p(x, x) \) and \( p(y, f(y)) \).

\[
\{ \langle p(x, x), p(y, f(y)) \rangle \}; \quad \varepsilon \xrightarrow{} \text{Dec}
\]

\[
\{ \langle x, y \rangle, \langle x, f(y) \rangle \}; \quad \varepsilon \xrightarrow{} \text{VarEl}
\]

\[
\{ \langle y, f(y) \rangle \}; \quad \{ x \mapsto y \} \xrightarrow{} \text{OccCh}
\]

\[\bot\]
Example (Infinite Terms)

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

X = f(f(f(f(f(f(f(f(f(f(...))))))))))

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

Yes
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

\texttt{less(X,s(X)).}
\texttt{foo:-less(s(Y),Y).}

?- foo.

Yes