

# Unification

We say that two expressions  $E$  and  $E'$  (e.g., terms or atoms) are *unifiable* if there exists a substitution  $\sigma$  such that  $E\sigma = E'\sigma$ .

If  $E$  and  $E'$  are unifiable, the substitution  $\sigma$  is also called a *unifier* of  $E$  and  $E'$ .

We use the notation

$$E \stackrel{?}{=} E'$$

to indicate that we are interested in the question whether  $E$  and  $E'$  are unifiable.

*Examples.* Let  $f$  and  $g$  be (unary) function symbols,  $a$  be a constant, and  $x$  and  $y$  be variables. Consider the unification problems:

$$\begin{array}{l} f(x) \stackrel{?}{=} f(a) \\ x \stackrel{?}{=} f(y) \\ f(x) \stackrel{?}{=} g(y) \\ x \stackrel{?}{=} f(x) \end{array}$$

For the first problem, there is one unifier,  $[x \mapsto a]$ ; for the second, there are many unifiers,  $[x \mapsto f(y)]$ ,  $[x \mapsto f(a), y \mapsto a]$ , etc.

For the other two problems, there exist no unifiers.

# Most General Unifiers

A *unification problem* is a finite set of pairs of expressions, usually written

$$E_1 =? E'_1, \dots, E_n =? E'_n.$$

A *unifier* or *solution* of such a problem is a substitution  $\sigma$  such that  $E_i\sigma = E'_i\sigma$ , for  $i = 1, \dots, n$ .

Some unifiers are preferable to others.

A substitution  $\sigma$  is said to be *more general* than a substitution  $\tau$  if there exists another substitution  $\tau'$  such that  $\tau$  is equal to the composition  $\sigma\tau'$  of  $\sigma$  and  $\tau'$ , where the composite substitution  $\sigma\tau'$  maps each variable  $x$  to  $\tau'(\sigma(x))$ .

For example,  $\sigma = [x \mapsto f(y)]$  is more general than  $\tau = [x \mapsto f(a), y \mapsto a]$ , because  $\tau = \sigma\tau'$ , where  $\tau' = [y \mapsto a]$ .

A *most general unifier* of a unification problem  $U$  is a unifier of  $U$  that is more general than any other unifier of  $U$ . The following theorem states a key result about unification.

## Theorem

If a unification problem has a solution then it has a most general unifier.

# Solved Forms

A unification problem  $U$  is said to be in *solved form* if it can be written as  $x_1 =^? t_1, \dots, x_n =^? t_n$ , where the variables  $x_i$  are pairwise distinct and do not occur in any of the terms  $t_j$ .

A unification problem  $U$  in solved form evidently has a unifier, namely the substitution

$$\vec{U} = [x_1 \mapsto t_1, \dots, x_n \mapsto t_n].$$

## Lemma.

If  $U$  is in solved form, then  $\sigma = \vec{U}\sigma$ , for all unifiers  $\sigma$  of  $U$ .

## Corollary.

If  $U$  is in solved form, then  $\vec{U}$  is a most general unifier of  $U$ . Moreover the unifier  $\vec{U}$  is idempotent, i.e.,  $\vec{U}\vec{U} = \vec{U}$ .

We will outline a method for solving unification problems that is based on transforming the problem to a solved form (if a unifier exists at all).

# Unification by Transformation

The following rules can be used to transform unification problems to solved form.

## Delete

$$t =? t, U \Rightarrow U$$

## Decompose

$$\begin{aligned} f(t_1, \dots, t_n) =? f(u_1, \dots, u_n), U \\ \Rightarrow t_1 =? u_1, \dots, t_n =? u_n, U \end{aligned}$$

## Orient

$$t =? x, U \Rightarrow x =? t, U$$

if  $t$  is not a variable

## Eliminate

$$x =? t, U \Rightarrow x =? t, U[x \mapsto t]$$

if the variable  $x$  occurs in  $U$ , but not in  $t$ .

Note. By  $s =? t, U$  we denote a unification problem where  $U$  does not contain  $s =? t$ .

# Soundness of Transformations

The above transformation rules are sound in the following sense.

## Lemma.

If  $U \Rightarrow U'$ , then  $U$  and  $U'$  have the same unifiers.

The lemma can be proved by inspection of the various rules.

## Corollary. [Soundness]

If  $U \Rightarrow^* U'$  and  $U'$  is in solved form, then  $\vec{U}'$  is a most general unifier of  $U$ .

Here  $U \Rightarrow^* U'$  indicates that  $U'$  can be obtained from  $U$  by zero or more transformation steps.

# Completeness of Transformations

## Theorem.

If  $U \Rightarrow^* U'$  and no further rule can be applied to  $U'$ , but  $U'$  is *not* in solved form, then  $U$  has no solution.

## *Sketch of Proof.*

If no transformation rule can be applied to  $U'$ , yet  $U'$  is *not* in solved form, then  $U'$  either contains an element

$$f(t_1, \dots, t_n) \stackrel{?}{=} g(u_1, \dots, u_k)$$

where  $f \neq g$ , or an element

$$x \stackrel{?}{=} t$$

where  $x$  is a variable occurring in  $t$ .

In either case, the unification problem has no solution.

# Additional Unification Rules

If a unification problem is unsolvable, it is often not necessary to apply transformation rules exhaustively.

As an optimization, we introduce a special unification problem “*fail*” that is considered to be unsolvable, and add the following two rules:

## Clash

$$\{f(t_1, \dots, t_n) =^? g(u_1, \dots, u_n)\} \cup U \Rightarrow \textit{fail}$$

if  $f \neq g$ .

## Occurs-Check

$$\{x =^? t\} \cup U \Rightarrow \textit{fail}$$

if the variable  $x$  occurs in  $t$ , but  $x \neq t$ .

# Unification of Atoms

The transformation rules can be applied to atomic formulas or terms.

Since atoms are expressions of the form

$$P(t_1, \dots, t_n),$$

where  $P$  is a predicate symbol, only two rules are possibly applicable to unification problems  $A =? B$ , where  $A$  and  $B$  are atoms:

If the two atoms have the same predicate symbol, decomposition is applicable; otherwise, the clash rule can be applied.

# Termination

We next use a lexicographic ordering (on triples) to prove that all sequences of transformation steps are finite, i.e., terminate.

## Lemma.

All possible sequences of transformations from any given unification problem are finite.

*Proof.* We assign a triple of natural numbers to each unification problem  $U$  and then show that each application of a transformation rule decreases these triples with respect to a well-founded lexicographic ordering.

We say that a variable  $x$  is *solved* with respect to a unification problem  $U$  if  $x$  occurs exactly once in  $U$ , namely on the left-hand side of some equation  $x =? t$ .

To each unification problem  $U$  we assign a triple of natural numbers:

1.  $n_1$  is the number of variables in  $U$  that are *not* solved,
2.  $n_2$  is the size of  $U$ , and
3.  $n_3$  is number of equations of the form  $t =? x$  in  $U$ .

We observe the following about application of transformation rules:

(a) Deletion and decomposition either decrease  $n_1$  (or leave it unchanged) and decrease  $n_2$ .

(b) Orientation either decreases  $n_1$  (or leaves it unchanged), does not change  $n_2$ , and decreases  $n_3$ .

(c) Elimination decreases  $n_1$ .

Since the three-fold lexicographic combination of the greater-than relation on the natural numbers provides a well-founded ordering on triples  $(n_1, n_2, n_3)$ , we may conclude that there can be no infinite sequence of transformations.

# Remarks on Efficiency

The transformation rules for unification we have described yield a unification method that is exponential in the worst case.

Consider the unification problem

$$f(g(x_1, x_1), \dots, g(x_{n-1}, x_{n-1})) \stackrel{?}{=} f(x_2, x_3, \dots, x_n),$$

which is solvable. The following substitution is a most general unifier:

$$x_2 \mapsto g(x_1, x_1)$$

$$x_3 \mapsto g(g(x_1, x_1), g(x_1, x_1))$$

...

$$x_n \mapsto g(g(\dots, g(x_1, x_1) \dots), g(\dots, g(x_1, x_1) \dots))$$

The variable  $x_n$  is mapped to a term containing  $2^n - 1$  occurrences of the function symbol  $g$ .

Unification methods that represent terms as trees take exponential time for the construction of this most general unifier.

There are efficient (linear-time) unification algorithms that use directed acyclic graphs to represent terms, so that each term or subterm is represented only *once* and different occurrences of the same term share its representation.