Prof. Dr. Cas Cremers, Dr. Aleksi Peltonen
Alexander Dax, Niklas Medinger

`https://cms.cispa.saarland/farwsp24/`

# Cheat Sheet - Unification

## 1 Unification - Intuition

In mathematics and computer science, unification is the process of making two expressions look exactly the same by finding appropriate substitutions for their variables. Imagine you have two terms (expressions with variables), and you want to see if there is a way to replace the variables so that both terms are identical. If you can find such a substitution that makes the two terms look the same, we say the terms are *unifiable*.

Consider two terms, $s$ and $t$, that we want to unify. These terms may include constants (fixed values) and variables (placeholders that can stand for various things). To unify $s$ and $t$, we need to find a way to assign specific values or expressions to the variables in both terms so that, once substituted, the two terms become identical. If we can achieve this with as many variables left as possible, we call that substitution the *most general unifier.*

### Substitutions

A *substitution* is a mapping $\sigma : \mathcal{V} \to \mathcal{T}$ from variables to terms.
We write $t\sigma$ to denote applying the substitution $\sigma$ to the term $t$.
This replaces each variable $x$ in $t$ by the term $\sigma\{x\}$.

**Example**

For $t = h(enc(m, x))$ and $\sigma = \{x \mapsto \mathit{kdf}(k, a, b)\}$, we can apply the substitution $\sigma\{x\} = \mathit{kdf}(k, a, b)$ to get $t\sigma = h(enc(m, \mathit{kdf}(k, a, b)))$.

**Note**

We use $\sigma = \{x \mapsto a, y \mapsto enc(a, b)\}$ and $\sigma = \{x/a, y/enc(a, b)\}$ interchangably.

### Unification

*Unification* determines if two terms with variables can be made equal. Two terms $u$ and $v$ are said *unifiable* if there exists a substitution $\sigma$, called a *unifier*, such that $u\sigma = v\sigma$.

**Example**

The terms $t_1 = x$ and $t_2 = h(y)$ are unifiable with e.g., $\sigma = \{x \mapsto h(a), y \mapsto a\}$

### Most general unifier vs. unifier

A *unifier* for two terms $u$ and $v$ is any substitution $\sigma$ that makes the terms identical, i.e., $u\sigma = v\sigma$. Unifiers are not necessarily unique; there can be many possible substitutions that make two terms equal. However, among these, some are more general than others.

**Definition of Most General Unifier (MGU):** A *most general unifier* (MGU) is a unifier that is as general as possible, meaning that it uses as little constants as possible. In other words, an MGU is a unifier $\sigma$ such that any other unifier $\tau$ of $u$ and $v$ can be expressed as $\tau = \sigma\delta$ for some substitution $\delta$. This means any other unifier can be obtained by further applying a substitution to the MGU.

An important property of an MGU is that the substitution it provides should be in a form where no variable on the right-hand side of a substitution occurs on the left-hand side of another substitution in the unifier. This avoids circular dependencies and keeps the substitution in its most general form, ensuring there are no unintended restrictions.

Example

Consider terms $t_1 = f(x, z)$ and $t_2 = f(y, a)$:

- A unifier for $t_1$ and $t_2$ could be $\{x \mapsto a, y \mapsto a, z \mapsto a\}$, but this substitution is restrictive as it forces all variables to be $a$.
- An MGU for $t_1$ and $t_2$ is $\sigma = \{x \mapsto y, z \mapsto a\}$.
  This is more general because any other unifier (like the one above) can be derived by further substituting specific values into $\sigma$.
- If we had a substitution like $\sigma' = \{x \mapsto a, y \mapsto x, z \mapsto x\}$, it would not satisfy this condition, as $x$ appears on the right-hand side for $y$ and $z$, which makes the unification less general (and potentially leads to circular references.)

## 2 An Example Algorithm to compute the MGU

While it is feasible to compute a most general unifier manually for smaller sets of equations, there are algorithms specifically designed to automate this process. Table 1 presents a straightforward approach for calculating the most general unifier efficiently.

| | |
|---|---|
| **Delete:** | If $E$ contains an equation $t = t$, remove it. |
| | $E \cup \{t = t\} \Rightarrow E$ |
| **Decompose:** | For $E \cup \{f(s_0, \ldots, s_k) = f(t_0, \ldots, t_k)\}$, add $s_i = t_i$ for each $i$. |
| | $E \cup \{f(s_0, \ldots, s_k) = f(t_0, \ldots, t_k)\} \Rightarrow E \cup \{s_0 = t_0, \ldots, s_k = t_k\}$ |
| **Conflict:** | If $E$ contains $f(s_0, \ldots, s_k) = g(t_0, \ldots, t_m)$ with $f \neq g$ or $k \neq m$, terminate with $\perp$. |
| **Swap:** | If $E$ has $f(s_0, \ldots, s_k) = x$, replace it with $x = f(s_0, \ldots, s_k)$. |
| **Eliminate:** | If $E$ has $x = t$ with $x$ not in $t$, replace all occurrences of $x$ in $E$ with $t$. |
| | $E \cup \{x = t\} \Rightarrow E\{x \mapsto t\} \cup \{x = t\}$ |
| **Occurs Check:** | If $x = f(s_0, \ldots, s_k)$ appears in $E$ and $x$ is within $f(s_0, \ldots, s_k)$, terminate with $\perp$. |

Table 1: Martelli & Montanari `https://dl.acm.org/doi/10.1145/357162.357169`

To execute the algorithm, apply any rules until you either end up at $\perp$, meaning no unifier exists, or until no rules can be applied anymore. In the latter case, the result will be the most general unifier if you replace the "=" with "$\mapsto$" (or "/").

**Note: Occurs Check** The *occurs check* prevents an infinite loop by ensuring that a variable does not appear within its substitution term. For example, an equation $x = f(x)$ has no finite solution and should be rejected.

## 3 Using the Algorithm

### Applying the Unification Algorithm to Exercise 1 (a) of Sheet 2

**Solution:**

(1) $\{enc(x, h(x, y)) = enc(enc(y, z), h(y, z'))\}$

$$\begin{aligned}
&\{enc(x, h(x, y)) = enc(enc(y, z), h(y, z'))\}\\
\Rightarrow_{decompose} &E \cup \{x = enc(y, z), h(x, y) = h(y, z')\}\\
\Rightarrow_{decompose} &E \cup \{x = enc(y, z), x = y, y = z'\}\\
\Rightarrow_{eliminate} &E \cup \{y = enc(y, z), x = y, y = z'\}\\
\Rightarrow_{occurs-check} &\perp
\end{aligned}$$

(2) $\{h(enc(x, x)) = z, h(h(enc(a, y))) = h(z)\}$

$$\begin{aligned}
&\{h(enc(x, x)) = z, h(h(enc(a, y))) = h(z)\}\\
\Rightarrow_{decompose} &E \cup \{h(enc(x, x)) = z, h(enc(a, y)) = z\}\\
\Rightarrow_{swap} &E \cup \{h(enc(x, x)) = z, z = h(enc(a, y)\}\\
\Rightarrow_{eliminate} &E \cup \{h(enc(x, x)) = h(enc(a, y)), z = h(enc(a, y))\}\\
\Rightarrow_{decompose\ x\ 2} &E \cup \{x = a, x = y, z = h(enc(a, y))\}\\
\Rightarrow_{swap} &E \cup \{x = a, y = x, z = h(enc(a, y))\}\\
\Rightarrow_{eliminate\ x\ 2} &E \cup \{x = a, y = a, z = h(enc(a, a))\}
\end{aligned}$$

(3) $\{h(enc(x, x)) = z, h(h(enc(a, y))) = h(z), mac(x, y) = v\}$

$$\begin{aligned}
&\{h(enc(x, x)) = z, h(h(enc(a, y))) = h(z), mac(x, y) = v\}\\
\Rightarrow_{decompose} &E \cup \{h(enc(x, x)) = z, h(enc(a, y)) = z, mac(x, y) = v\}\\
\Rightarrow_{swap} &E \cup \{h(enc(x, x)) = z, z = h(enc(a, y)), mac(x, y) = v\}\\
\Rightarrow_{eliminate} &E \cup \{h(enc(x, x)) = h(enc(a, y)), z = h(enc(a, y)), mac(x, y) = v\}\\
\Rightarrow_{decompose\ x\ 2} &E \cup \{x = a, x = y, z = h(enc(a, y)), mac(x, y) = v\}\\
\Rightarrow_{swap\ x\ 2} &E \cup \{x = a, y = x, z = h(enc(a, y)), v = mac(x, y)\}\\
\Rightarrow_{eliminate\ x\ 2} &E \cup \{x = a, y = a, z = h(enc(a, a)), v = mac(a, a)\}
\end{aligned}$$

(4) $\{\mathrm{prf}(x, y) = \mathrm{prf}(x', y'), y' = enc(x, a), enc(h(a), z) = y\}$

$$\begin{aligned}
&\{\mathrm{prf}(x, y) = \mathrm{prf}(x', y'), y' = enc(x, a), enc(h(a), z) = y\}\\
\Rightarrow_{decompose} &\{x = x', y = y', y' = enc(x, a), enc(h(a), z) = y\}\\
\Rightarrow_{eliminate} &\{x = x', y = y', y' = enc(x', a), enc(h(a), z) = y\}\\
\Rightarrow_{eliminate} &\{x = x', y = y', y' = enc(x', a), enc(h(a), z) = y'\}\\
\Rightarrow_{eliminate} &\{x = x', y = enc(x', a), y' = enc(x', a), enc(h(a), z) = enc(x', a)\}\\
\Rightarrow_{decompose} &\{x = x', y = enc(x', a), y' = enc(x', a), h(a) = x', z = a\}\\
\Rightarrow_{swap} &\{x = x', y = enc(x', a), y' = enc(x', a), x' = h(a), z = a\}\\
\Rightarrow_{eliminate} &\{x = h(a), y = enc(h(a), a), y' = enc(h(a), a), x' = h(a), z = a\}
\end{aligned}$$