Unification

Definition 1 (Unification) *A substitution σ (of possibly several variables by terms) is called a unifier of a set of formulas $\Delta = \{\varphi_1, \ldots, \varphi_n\}$ iff $\sigma(\varphi_1) = \cdots = \sigma(\varphi_n)$ holds. We also write $|\sigma(\Delta)| = 1$ and call Δ unifiable.*

Definition 2 (MGU) *A unifier μ of a set of formulas Δ is called a most general unifier (mgu) of Δ iff for every unifier σ of Δ there exists a substitution σ' with $\sigma = \mu \circ \sigma'$.*

(The composition $\mu \circ \sigma'$ is the substitution we get by first applying μ to a formula and then σ'.)

Remark. We also speak of unifiers (and mgus) for sets of *terms*.

Unification Algorithm: Preparation

We shall formulate a unification algorithm for literals only, but it can easily be adapted to work with general formulas (or terms).

Subexpressions. Let φ be a literal. We refer to formulas and terms appearing within φ as the *subexpressions* of φ. If there is a subexpression in φ starting at position i we call it $\varphi^{(i)}$ (otherwise $\varphi^{(i)}$ is undefined; for example, if there is a comma at the ith position).

Disagreement pairs. Let φ and ψ be literals with $\varphi \neq \psi$ and let i be the smallest number such that $\varphi^{(i)}$ and $\psi^{(i)}$ are defined and $\varphi^{(i)} \neq \psi^{(i)}$. Then $(\varphi^{(i)}, \psi^{(i)})$ is called the *disagreement pair* of φ and ψ. Example:

$$\begin{aligned} \varphi &= P(g_1(c), f_1(a, g_1(x), g_2(a, g_1(b)))) \\ \psi &= P(g_1(c), f_1(a, g_1(x), g_2(\underset{\uparrow}{f_2(x,y)}, z)) \end{aligned}$$

Disagreement pair: $(a, f_2(x, y))$

Robinson's Unification Algorithm

set $\mu := [\,]$ (empty substitution)
while $|\mu(\Delta)| > 1$ **do** {
    pick a disagreement pair p in $\mu(\Delta)$;
    if *no variable in* p **then** {
        stop and **return** 'not unifiable';
    } **else** {
        let $p = (x, t)$ with x being a variable;
        if x *occurs in* t **then*** {
            stop and **return** 'not unifiable';
        } **else** {
            set $\mu := \mu \circ [x \leftarrow t]$;
        }
    }
}
return μ;

Input: Δ (set of literals)
Output: μ (mgu of Δ)
or 'not unifiable'

* so-called *occur-check*

Unification $\neq$ Matching (Prolog)

Most Prolog systems do not implement a sound unification algorithm, as this would be computationally too expensive (they usually omit the *occur-check*).

Example: SWI-Prolog matching the variable `X` and the term `f(X)`:

```
?- X = f(X).
X = f(f(f(f(f(f(f(f(f(f(...))))))))))
Yes
```

If you require sound unification you have to implement it yourself.