Unification and Skolemization

Benedikt Seifert
07.06.2018
Informal motivation

Unification Replace variables to make two formulas identical if it is possible to achieve

Skolemization Perform steps to derive a formula without $\exists$-quantifier which behaves like the original formula
Motivation for Unification

Consider the two terms

\[ s = f(x, g(a, b)) \quad t = f(g(y, b), x) \]
Motivation for Unification

Consider the two terms

\[ s = f(x, g(a, b)) \quad t = f(g(y, b), x) \]

Replace \( x \) with \( g(a, b) \)

\[ s = f(g(a, b), g(a, b)) \quad t = f(g(y, b), g(a, b)) \]
Motivation for Unification

Consider the two terms

\[ s = f(x, g(a, b)) \quad t = f(g(y, b), x) \]

Replace \( x \) with \( g(a, b) \)

\[ s = f(g(a, b), g(a, b)) \quad t = f(g(y, b), g(a, b)) \]

Replace \( y \) with \( a \)

\[ s = f(g(a, b), g(a, b)) \quad t = f(g(a, b), g(a, b)) \]

The terms are now unified
Substitution

The replacement steps are called substitution

A substitution is a function $\sigma$ with a finite set of variables called domain and the set of replacements for the items in the domain called codomain.

The substitution $\sigma$ that transforms $t = f(g(y, b), x)$ to $t\sigma = f(g(a, b), g(a, b))$ we write $t\{y/a, x/g(a, b)\}$

With domain \{x, y\} and codomain \{a, g(a, b)\}
Consider the substitution \( \sigma = \{x/f(x)\} \)

\[
s = f(x) \quad s\sigma = f(f(x))
\]

From \( s \) you can always generate \( s\sigma \) but you cannot substitute from \( s\sigma \) to \( s \). Thus \( s\sigma \) is called instance of \( s \) or you say \( s \) subsumes \( s\sigma \)
From these considerations we can define an order of subsumptions

\[ \sigma \leq \sigma\{x/f(x)\} \leq \sigma\{x/f(f(x))\} \leq \ldots \]

Basically we are interested in the first element of that series unifying two terms which is called the most general unifier.
Unification is used extensively in e.g. Prolog. Consider this knowledge base

\[
\text{vertical}(\text{line}(\text{point}(X,Y), \text{point}(X,Z))).
\]

And these queries

\[
?\leftarrow \text{vertical}(\text{line}(\text{point}(1,1), \text{point}(1,3))). \quad \text{yes}
\]

\[
?\leftarrow \text{vertical}(\text{line}(\text{point}(1,1), \text{point}(2,3))). \quad \text{no}
\]

Prolog verifies and falsifies the queries based on the knowledge base with unification.
Other usages of unification would be

- type checking
- type inference
- database queries
- natural language processing
- ..
Consider the formula

\[ \forall y \forall z (m(y, z) \Rightarrow \exists x p(y, x)) \]

Aim: get rid of \( \exists x \) in \( Qx\varphi = \exists x p(y, x) \)
Consider the formula

$$\forall y \forall z (m(y, z) \implies \exists x p(y, x))$$

Aim: get rid of $\exists x$ in $Qx \varphi = \exists x p(y, x)$

Solution: replace $x$ with n-ary function $f$ with $n$ the amount of free variables in $\varphi$

$$\forall y \forall z (m(y, z) \implies p(y, f(y)))$$
Skolemization downside

The formula before skolemization \( \varphi \) and after skolemization \( \varphi_{sko} \) are in general not logically equivalent. \( \varphi \nrightarrow \varphi_{sko} \)

Example:

\[
\varphi = \forall x \exists y \ lessThan(x, y) \\
\varphi_{sko} = \forall x \ lessThan(x, f(x))
\]
Applications of Skolemization

Skolemization is one of the main steps of preparing formulas for Automated Theorem Proving mostly with enhancements to guarantee better runtime behavior.

It is as well a preparation step for the resolution method of first order logic.
Questions or remarks?