Syntactic unification

Julio Mariño

Theory of Programming Languages
MASTER IN FORMAL METHODS FOR SOFTWARE ENGINEERING
Universidad (Politécnica | Complutense | Autónoma) de Madrid

December 14, 2021

prelude: first order syntax

- We will be considering first order terms just like the ones considered in the lectures on rewriting. These terms are made of variables, constants and (k-ary) functors.
- example: $f(x, g(z))$, $x^2 - x$, 42, $employee(pepe, janitor)$, $a \to (a \to b)$, etc.
- Generally speaking, a unification problem is finding some variable substitution that makes two terms equal.
- example: If we consider arithmetic expressions under the standard arithmetic interpretation for constants and functions, then the terms $x^2 - x$ and 42 are unified by the substitution $\{x \mapsto 7\}$.
- In a syntactic unification problem terms are just interpreted literally, i.e. as its syntactic appearance – the *free* algebra.
- example: $employee(pepe, y)$ and $employee(x, janitor)$ can be unified by $\{x \mapsto pepe, y \mapsto janitor\}$. $a \to (a \to b)$ and $c \to c$ cannot be unified syntactically.

substitutions and unifiers

- Substitutions on first order terms are defined in the usual recursive way. As there are no *scoping* constructs such as quantifiers there are no such issues as variable capture, etc.
- Substitutions can be composed. Composition is associative, as usual, but generally not commutative. The composition of substitutions σ and σ' is another substitution $\sigma \circ \sigma'$ such that for every term t, $(\sigma \circ \sigma')(t) = \sigma(\sigma'(t))$.
- A term s is *more general* than t when there is some substitution σ such that $t = \sigma(s)$.
- A substitution σ is *more general* than τ when for every term t, $\sigma(t)$ is *more general* than $\tau(t)$. Corollary: A substitution σ is *more general* than τ iff there is some substitution σ' such that $\tau = \sigma \circ \sigma'$
- A substitution σ is a unifier of two terms s and t when $\sigma(s) = \sigma(t)$. A *most general unifier* (mgu for short) of two terms is a unifier that is more general than any other unifier for those terms.
- The question now is: given two first order terms, is there a way of finding an syntactic mgu for them, if it exists?

an algorithm for most general unifiers

Martelli & Montanari

- Main idea: transform a system of (syntactic) equations into an equivalent one which is either contradictory or trivially satisfiable.
- The rules:

$$
\begin{array}{rcl}
S \cup \{t = t\} & \rightsquigarrow & S \\
 & & \text{(DELETE)} \\
S \cup \{f(s_1, \ldots, s_k) = f(t_1, \ldots, t_k)\} & \rightsquigarrow & S \cup \{s_1 = t_1, \ldots, s_k = t_k\} \\
 & & \text{(DECOMPOSE)} \\
S \cup \{f(s_1, \ldots, s_k) = g(t_1, \ldots, t_m)\} & \rightsquigarrow & \perp \\
 & & f \neq g \vee k \neq m \quad \text{(CONFLICT)} \\
S \cup \{f(s_1, \ldots, s_k) = x)\} & \rightsquigarrow & S \cup \{x = f(s_1, \ldots, s_k))\} \\
 & & \text{(SWAP)} \\
S \cup \{x = f(t1, \ldots, t_k)\} & \rightsquigarrow & \perp \\
 & & x \in \mathrm{vars}(f(t1, \ldots, t_k)) \quad \text{(OCCURS CHECK)} \\
S \cup \{x = t\} & \rightsquigarrow & S\{x \mapsto t\} \cup \{x = t\} \\
 & & x \notin \mathrm{vars}(t) \quad \text{(ELIMINATE)}
\end{array}
$$

examples

$$f(x, h(x)) \quad = \quad f(g(y), z)$$

examples

$$f(x, h(x)) = f(g(y), z)$$

DECOMPOSE:

$$\begin{aligned} x &= g(y) \\ h(x) &= z \end{aligned}$$

examples

$$f(x, h(x)) \quad = \quad f(g(y), z)$$

DECOMPOSE:

$$\begin{array}{rcl} x & = & g(y) \\ h(x) & = & z \end{array}$$

SWAP:

$$\begin{array}{rcl} x & = & g(y) \\ z & = & h(x) \end{array}$$

examples

$$f(x, h(x)) \quad = \quad f(g(y), z)$$

DECOMPOSE:

$$\begin{array}{rcl} x & = & g(y) \\ h(x) & = & z \end{array}$$

SWAP:

$$\begin{array}{rcl} x & = & g(y) \\ z & = & h(x) \end{array}$$

ELIMINATE:

$$\begin{array}{rcl} x & = & g(y) \\ z & = & h(g(x)) \end{array}$$

examples

$$f(x, h(x)) \quad = \quad f(g(y), z)$$

DECOMPOSE:

$$\begin{array}{rcl} x & = & g(y) \\ h(x) & = & z \end{array}$$

SWAP:

$$\begin{array}{rcl} x & = & g(y) \\ z & = & h(x) \end{array}$$

ELIMINATE:

$$\begin{array}{rcl} x & = & g(y) \\ z & = & h(g(x)) \end{array}$$

mgu is $\{x \mapsto g(y), z \mapsto h(g(y))\}$

examples (II)

$$f(x, h(x)) \quad = \quad f(g(z), z)$$

examples (II)

$$f(x, h(x)) = f(g(z), z)$$

DECOMPOSE:

$$\begin{aligned} x &= g(z) \\ h(x) &= z \end{aligned}$$

examples (II)

$$f(x, h(x)) \quad = \quad f(g(z), z)$$

$$\begin{array}{c} \text{DECOMPOSE:} \\ \begin{array}{rcl} x & = & g(z) \\ h(x) & = & z \end{array} \end{array}$$

$$\begin{array}{c} \text{SWAP:} \\ \begin{array}{rcl} x & = & g(z) \\ z & = & h(x) \end{array} \end{array}$$

examples (II)

$$f(x, h(x)) \quad = \quad f(g(z), z)$$

DECOMPOSE:

$$\begin{array}{rcl} x & = & g(z) \\ h(x) & = & z \end{array}$$

SWAP:

$$\begin{array}{rcl} x & = & g(z) \\ z & = & h(x) \end{array}$$

ELIMINATE:

$$\begin{array}{rcl} x & = & g(y) \\ z & = & h(g(z)) \end{array}$$

examples (II)

$$f(x, h(x)) \quad = \quad f(g(z), z)$$

DECOMPOSE:

$$\begin{array}{rcl} x & = & g(z) \\ h(x) & = & z \end{array}$$

SWAP:

$$\begin{array}{rcl} x & = & g(z) \\ z & = & h(x) \end{array}$$

ELIMINATE:

$$\begin{array}{rcl} x & = & g(y) \\ z & = & h(g(z)) \end{array}$$

OCCURS CHECK:

$$\perp$$

examples (II)

$$f(x, h(x)) \quad = \quad f(g(z), z)$$

DECOMPOSE:

$$\begin{array}{rcl} x & = & g(z) \\ h(x) & = & z \end{array}$$

SWAP:

$$\begin{array}{rcl} x & = & g(z) \\ z & = & h(x) \end{array}$$

ELIMINATE:

$$\begin{array}{rcl} x & = & g(y) \\ z & = & h(g(z)) \end{array}$$

OCCURS CHECK:

$$\perp$$

There is no mgu.