Technical University of Munich
Chair for Logic and Verification

Prof. Tobias Nipkow
Kevin Kappelmann

SS 2021 | Exercise Sheet 8 | 04.06.2021

Exercise 8.1. [Simultaneous substitution]
Recall that $[t_1/x_1, \ldots, t_n/x_n]$ is the *simultaneous* substitution of $x_1, \ldots, x_n$ by $t_1, \ldots, t_n$.

1. Can we always express $[t_1/x_1, \ldots, t_n/x_n]$ as a series of one-variable substitutions?
2. Can we always summarise a series of one-variable substitutions to a single simultaneous substitution?

Solution:

1. No. Counterexample: $[y/x, x/y]$ (exchanges x and y).
   Note: It is possible for a concrete F though because we could make up fresh variable names, e.g. $F(x,y)[y/x, x/y] = F(x,y)[z/x][x/y][y/z]$.
2. Yes. We can give a rule to "consolidate" a simultaneous substitution and a one-variable substitution:
$$[t_1/x_1, \ldots, t_n/x_n][u/y] = \begin{cases} \big[t_1[u/y]/x_1, \ldots, t_n[u/y]/x_n, u/y\big], & \text{if } y \notin \{x_1, \ldots, x_n\} \\ \big[t_1[u/y]/x_1, \ldots, t_n[u/y]/x_n\big], & \text{otherwise} \end{cases}$$
   Repeatedly apply this rule to obtain a single simultaneous substitution.

Exercise 8.2. [Occurs check]
What happens if one omits the occurs check in the unification algorithm? Find an example where the unification algorithm without occurs check diverges or returns the wrong result.

Solution:
Consider $x \stackrel{?}{=} f(x)$. Without the occurs check, we first produce $\sigma = \{x \mapsto f(x)\}$. The algorithm keeps going and produces $\sigma' = \{x \mapsto f(f(x))\}$, then $\sigma'' = \{x \mapsto f(f(f(x)))\}$ and so on.

Exercise 8.3. [Unifiable terms]
Specify the most general unifiers for the following sets of terms, if one exists:

$$L_1 = \{f(x,y), f(h(a),x)\}$$
$$L_2 = \{f(x,y), f(h(x),x)\}$$
$$L_3 = \{f(x,b), f(h(y),z)\}$$
$$L_4 = \{f(x,x), f(h(y),y)\}$$

Solution:

L_1 :	$[h(a)/x, h(a)/y]$
L_2 :	No unifier, occurs check fails on $x \sim h(x)$
L_3 :	$[h(y)/x, b/z]$
L_4 :	No unifier, occurs check fails on $h(y) \sim y$

Exercise 8.4. [**Formulas without negation**]
Prove that every predicate logic formula that only contains $\wedge, \vee, \forall, \exists, \longrightarrow$ and atomic formulas is satisfiable. Is such a formula also valid?

Solution:
Let $\mathcal{A}$ be a structure with a singleton universe (e.g. $U_{\mathcal{A}} = \{0\}$) that interprets all predicates to be true everywhere. Then, by straightforward induction on the formula, we get that $\mathcal{A}$ is a model where in each step we introduce an interpretation for all free and bound variables to the only element of the universe.

However, the formula needs not to be valid, e.g. $\forall x.P(x)$.

Homework 8.1. [Most general unifier] (+)
Consider the unification problem $x \stackrel{?}{=} f(y)$. Without running the unification algorithm, prove that

1. $\sigma_1 = [f(y)/x]$ is a most general unifier.
2. $\sigma_2 = [f(c)/x, c/y]$ is unifier, but not a most general unifier.

Solution:

1. σ is obvioulsy a unifier. Let σ be a unifier of x and $f(y)$. Then $x\sigma = f(t)$ and $y\sigma = t$ for some term t. Thus $\sigma = \sigma_1[t/y]$. Hence σ is an mgu.
2. σ_2 is obviously a unifier but it is not a most general one since $\sigma_2 = \sigma_1[c/y]$ but there is no δ such that $\sigma_1 = \sigma_2\delta$.

Homework 8.2. [Unification] (+)
Use the algorithm presented in the lecture to compute a most general unifier for the following set of formulas: $\{P(g(x), f(a)), P(y, x), P(g(f(z)), f(z))\}$

Solution:
Algorithmic.

Homework 8.3. [Untangling simultaneous substitution] (++)
Recall Exercise 8.1. Demonstrate how to "untangle" a simultaneous substitution that has been obtained by consolidating one-variable substitutions back into one-variable substitutions.

Solution:
Take some substitution $[t_1/x_1, \ldots, t_n/x_n]$. Let t'_i denote the term obtained by replacing all subterms t_n in t_i by x_n. We have $t_i = t'_i[t_n/x_n]$ and thus

$$[t_1/x_1, \ldots, t_n/x_n] = [t'_1/x_1, \ldots, t'_{n-1}/x_{n-1}][t_n/x_n].$$

Apply this process repeatedly to obtain the wanted series of one-variable substitutions.

Homework 8.4. [Anti-Unification] (+++)

A term t is a *generalisation* of a list of terms S if for each $s \in S$ there is a substitution σ_s such that $t\sigma_s = s$. A term t is a *most specific generalisation* (msg) of S if for any generalisation t' of S, there is a substitution $\sigma_{t'}$ such that $t'\sigma_{t'} = t$.

Give a recursive procedure that computes the msg of a finite list S. Apply your algorithm to the list $S \coloneqq \big[f(g(x), x, d, x), f(x, g(x), d, g(x)), f(h(c), h(c), d, h(c))\big]$ and prove that the returned msg is indeed an msg of S.

Hint: design an algorithm that operates recursively on the structure of terms.

Bonus: Prove that your algorithm always returns the msg.

Solution:

Call our algorithm $\mathsf{msg}(\cdot)$. Input: non-empty list S

1. If all terms in S are equal, then return $\mathsf{head}(S)$.
2. If $S = \big[f(t_1^1, \ldots, t_n^1), \ldots, f(t_1^k, \ldots, t_n^k)\big]$ then compute $t_i \coloneqq \mathsf{msg}(S_i)$ with $S_i \coloneqq [t_i^1, \ldots, t_i^k]$ for $1 \leq i \leq n$ and return $f(t_1, \ldots, t_n)$.
3. Otherwise return x_S.

As for the example:

1. We hit case 2 and must recurse
2. $S_1 = [g(x), x, h(c)]$: We hit case 3 and return $x_{[g(x),x,h(c)]}$.
3. $S_2 = [x, g(x), h(c)]$: We hit case 3 and return $x_{[x,g(x),h(c)]}$.
4. $S_3 = [d, d, d]$: We hit case 1 and return d.
5. $S_4 = [x, g(x), h(c)]$: We hit case 3 and return $x_{[x,g(x),h(c)]}$.
6. We return $f(x_{[g(x),x,h(c)]}, x_{[x,g(x),h(c)]}, d, x_{[x,g(x),h(c)]})$. The returned term is equivalent to $f(x, y, d, y) =: t$.

It is easy to check that t is a generalisation of S. Let t_i be the i-th term in S. Let t' be another generalisation. Then there are $\sigma_1, \sigma_2, \sigma_3$ such that $t'\sigma_i = t_i$. If t' is a variable, then t is obviously more specific. Hence, t' must be of the form $f(t'_1, \ldots, t'_4)$. Since $t'\sigma_2 = f(x, g(x), d, g(x))$, t'_1 must be a variable x_1 and t'_3 a variable x_3 or the constant d. Since $t'\sigma_1 = f(g(x), x, d, x)$, t'_2 must be a variable x_2 and t'_4 a variable x_4. So either $t' = f(x_1, x_2, x_3, x_4)$ or $t' = f(x_1, x_2, d, x_4)$. In the former case, t is more specific by already setting x_3 to d and in the latter case t is more specific by already unifying x_2 and x_4.

> Nature will always maintain her rights and prevail in the end over any abstract reasoning whatsoever.
>
> — David Hume