Technical University of Munich
Chair for Logic and Verification

Prof. Tobias Nipkow
Kevin Kappelmann

SS 2022 | Exercise Sheet 8 | 17.06.2022

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

1. Can we always express $[t_1/x_1, \dots, t_n/x_n]$ 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:

$$[t_1/x_1, \dots, t_n/x_n][u/y] = \begin{cases} [t_1[u/y]/x_1, \dots, t_n[u/y]/x_n, u/y], & \text{if } y \notin \{x_1, \dots, x_n\} \\ [t_1[u/y]/x_1, \dots, t_n[u/y]/x_n], & \text{otherwise} \end{cases}$$

   Repeatedly apply this rule to obtain a single simultaneous substitution.

Exercise 8.2. **[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_1 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. Let $y\delta := t$ and $v\delta := v\sigma$ for every variable $v \notin \{x, y\}$. Then $\sigma = \sigma_1\delta$. Hence σ_1 is an mgu. (Note: we could also have chosen $\delta = \sigma$ in this specific case).
2. σ_2 is obviously a unifier but it is not a most general one since there is no δ such that $[f(y)/x] = \sigma_1 = \sigma_2\delta = [f(c)/x, c/y]\delta$ because $c\delta = c \neq y$.

Exercise 8.3. **[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 8.4. [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 $h(y) \sim y$

Homework 8.1. **[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 8.2. **[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 8.3. [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 $S \coloneqq \big[f(g(x), x, d, x), f(x, g(x), d, g(x)), f(h(c), h(c), d, h(c))\big]$ (where c, d are constants) and prove that the returned msg is indeed an msg of S.

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

Optional: 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 $1 \le i \le n$ 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 $f(x, y, d, y) \eqqcolon t$.

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.

Homework 8.4. [We're Far From The Shallow Now] (+++)
In this exercise, we consider FOL without constants.

A term is called *shallow* if it contains no nested function. For example, x and $f(x)$ are shallow while $f(f(x))$ is not.

An atom is called *simple* if it only contains shallow terms. For example, $R(x)$ and $R(f(x))$ are simple while $R(f(f(x)))$ is not.

An atom is *covering* if every functional subterm of it contains all variables of the atom. For example, $R(x_1, x_2)$ and $R(f(x_1, x_2), x_2)$ are covering while $R(f(x_1), x_2)$ is not.

Let $A \coloneqq R(t_1, \ldots, t_n)$ and $B \coloneqq R(t'_1, \ldots, t'_n)$ be atoms that are simple and covering, $\mathsf{vars}(A) \cap \mathsf{vars}(B) = \emptyset$, and assume θ is an mgu of A, B. Show that $C \coloneqq A\theta = B\theta$ is simple.

Solution:
In the following, we abbreviate a list of terms $t_1, \ldots, t_n$ by $\vec{t}$. Let $\vec{x} \coloneqq \mathsf{vars}(A), \vec{y} \coloneqq \mathsf{vars}(B)$, and fix a fresh variable $z \notin \vec{x} \cup \vec{y}$. We consider the following cases:

1. A and B are function-free.
2. A contains a function and B is function-free (or vice versa).
3. A and B contain functions.

<u>Case 1:</u> Define a substitution θ' by $v\theta' \coloneqq z$ for any $v \in \vec{x} \cup \vec{y}$. Then $A\theta' = B\theta'$ are function-free. Since θ is an mgu, there is some δ such that $\theta' = \theta\delta$. Since δ may only introduce functions and not eliminate them, also $A\theta = B\theta$ is function-free and hence simple.

<u>Case 2:</u> WLOG assume A contains a function and B is function-free (the other case is symmetric). We define a substitution θ' by

$$v\theta' \coloneqq \begin{cases} z, & \text{if } v \in \vec{x} \\ t_i\theta', & \text{if } v = t'_i \in \vec{y} \end{cases}.$$

We show that θ' is well-defined; that is, if $y_k = t'_i = t'_j$, then $t_i\theta' = t_j\theta'$. Again, we consider three cases:

- If $t_i, t_j \in \vec{x}$, then $t_i\theta' = z = t_j\theta'$.
- Assume $t_i = f_1(\vec{x})$ and $t_j = f_2(\vec{x})$. As θ is a unifier, we have $t_i\theta = t'_i\theta = t'_j\theta = t_j\theta$, and hence $f_1 = f_2$. Consequently, $t_i\theta' = f_1(\vec{z}) = f_2(\vec{z}) = t_j\theta'$.
- Lastly, consider the cases $t_i \in \vec{x}$ and $t_j = f(\vec{x})$, or $t_j \in \vec{x}$ and $t_i = f(\vec{x})$. WLOG, assume the former (the other case is symmetric). Then $t_i\theta = t'_i\theta = t'_j\theta = t_j\theta = f(\vec{x})\theta$. But $t_i\theta \neq f(\vec{x})\theta$ since $t_i \in \vec{x}$ (occurs check), a contradiction. Consequently, the case $t_i \in \vec{x}$ and $t_j = f(\vec{x})$ cannot emerge.

Hence, θ' is well-defined. Moreover, $t'_i\theta' = t_i\theta'$ by definition; that is, $A\theta' = B\theta'$. Moreover, θ' does not introduce functions on $\vec{x}$ and only shallow terms on $\vec{y}$, and hence $A\theta' = B\theta'$ is simple. Hence, as in Case 1, $A\theta = B\theta$ is simple.

<u>Case 3:</u> Finally assume A and B contain functions. Again define a substitution θ' by $v\theta' \coloneqq z$ for any $v \in \vec{x} \cup \vec{y}$. We show that $A\theta' = B\theta'$; that is, $t_i\theta' = t'_i\theta'$ for $1 \leq i \leq n$.

- If $t_i \in \vec{x}$ and $t'_i \in \vec{y}$, then $t_i\theta' = z = t'_i\theta$.
- If $t_i = f_1(\vec{x})$ and $t'_i = f_2(\vec{y})$, then again $f_1 = f_2$ using that θ is a unifier, and consequently, $t_i\theta' = f_1(\vec{z}) = f_2(\vec{z}) = t'_i\theta'$.
- Lastly, consider the cases $t_i \in \vec{x}$ and $t'_i = f(\vec{y})$, or $t_i = f(\vec{x})$ and $t'_i \in \vec{y}$. WLOG, assume the former. Then $t_i\theta = t'_i\theta = f(\vec{y})\theta$. As A is functional, there is some $j \in \{1, \ldots, n\}$ with $t_j = f'(\vec{x})$, and thus $f'(\vec{x})\theta = t_j\theta = t'_j\theta$. We have two cases:
  - If $t'_j = f'(\vec{y})$, then $f'(\vec{x})\theta = f'(\vec{y})\theta$. As $t_i \in \vec{x}$, this implies $t_i\theta = y_k\theta$ for some $y_k \in \vec{y}$. Hence, $y_k\theta = t_i\theta = t'_i\theta = f(\vec{y})\theta$, a contradiction (occurs check).
  - If $t'_j = y_k$, then $y_k\theta = f'(\vec{x})\theta$. As $t_i \in \vec{x}$, $t_i\theta$ is a subterm of $y_k\theta$. But we also have $t_i\theta = f(\vec{y})\theta$, a contradiction.

  Consequently, the case $t_i \in \vec{x}$ and $t'_i = f(\vec{y})$ cannot emerge.

Hence, θ' unifies A and B. Moreover, θ' does not introduce functions, and hence $A\theta' = B\theta'$ is simple. Thus, $A\theta = B\theta$ is simple.

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