Exercise 24 (Equivalence Classes)

Let \( \Sigma = \{ f, g \} \) and

\[
E = \{ f(f(x)) \approx f(x), \ g(f(x)) \approx f(x), \ f(g(x)) \approx g(x), \ g(g(x)) \approx g(x) \}
\]

a) Describe the equivalence classes of \( \approx_E \).

b) For each equivalence class \([t]_{\approx_E}\), determine a shortest term \( \hat{t} \) in \([t]_{\approx_E}\).

c) Give a model for \( E \) that has more than one element.

Solution

a) We argue by using Birkhoff’s theorem.

As \( \Sigma \) only contains unary function symbols, all terms in \( T(\Sigma, V) \) are repeated applications of the function symbols \( f \) and \( g \) to some variable \( x \in V \).

An application of an equation in \( E \) (reduction relation \( \leftrightarrow_E \)) leaves the innermost function symbol unchanged. Thus, we get three equivalence classes for each variable \( x \in V \):

i) The set of all terms with \( f(x) \) as innermost subterm.

ii) The set of all terms with \( g(x) \) as innermost subterm.

iii) The singleton set \( \{x\} \).

b) The shortest subterms in the equivalence classes for variable \( x \) are \( f(x), g(x) \) and \( x \), respectively.

c) We choose a \( \Sigma \)-algebra \( A \) with carrier set \( A = \mathbb{N} \) and the following interpretations of the function symbols: \( f^A = g^A = id \).

Exercise 25 (Congruence Closure)

Let \( \Sigma = \{ f, a, b \} \) and \( G = \{ f(a, b) \approx a \} \). Using congruence closure, decide whether the terms \( s \) and \( t \) are equivalent with respect to the equation set \( G \). Use the abstract algorithm on equation sets.

a) \( s = f(f(a, b), b), \quad t = a \)

b) \( s = f(f(a, b), b), \quad t = b \)
Solution

\[ H_0 = G = \{ f(a, b) = a \} \]
\[ H_1 = \{ \begin{array}{l}
R_a = a, b = b, f(a, b) = f(a, b), f(f(a, b), b) = f(f(a, b), b), \\
S_a = f(a, b), f(f(a, b), b) = f(a, b) 
\end{array} \} \]
\[ H_2 = \{ \begin{array}{l}
S_a = f(a, b), f(f(a, b), b), \\
T f(f(a, b), b) = a 
\end{array} \} \]

At this point, we have shown \( f(f(a, b), b) = a \).
\[ H_3 = \{ S_a = f(f(a, b), b) \} \]

Here, the algorithm terminates and we get \( f(f(a, b), b) \neq b \).

Exercise 26 (Unification)

Let \( u, x, y, \) and \( z \) be variables. Use the unification algorithm recalled in the lecture to solve the following two unification problems:

a) \( S_1 = \{ f(h(x), g(x, u)) \overset{\gamma}{=} f(z, g(f(y, y), z)) \} \)
b) \( S_2 = \{ h(x, g(x, y), y) \overset{\gamma}{=} h(x, g(a, y), y), z \overset{\gamma}{=} h(x, g(x, b), b) \} \)

Solution

We proceed by applying one of the four rules Delete, Decompose, Orient and Eliminate in each step.

a)

\[
\begin{align*}
\{ f(h(x), g(x, u)) & \overset{\gamma}{=} f(z, g(f(y, y), z)) \} & \quad \text{Decompose} \\
\{ h(x) & \overset{\gamma}{=} z, g(x, u) \overset{\gamma}{=} g(f(y, y), z) \} & \quad \text{Decompose} \\
\{ h(x) & \overset{\gamma}{=} z, x \overset{\gamma}{=} f(y, y), u \overset{\gamma}{=} z \} & \quad \text{Orient} \\
\{ z & \overset{\gamma}{=} h(x), x \overset{\gamma}{=} f(y, y), u \overset{\gamma}{=} h(x) \} & \quad \text{Eliminate} \\
\{ z & \overset{\gamma}{=} h(f(y, y)), x \overset{\gamma}{=} f(y, y), u \overset{\gamma}{=} h(f(y, y)) \} & \quad \text{Eliminate}
\end{align*}
\]
b)

\[
\{ h(x, g(x, y), y) \Rightarrow h(x, g(a, y), y), z \Rightarrow h(x, g(x, b), b) \} \quad \Rightarrow \text{Decompose}
\]

\[
\{ x \Rightarrow x, g(x, y) \Rightarrow g(a, y), y \Rightarrow y, z \Rightarrow h(x, g(x, b), b) \} \quad \Rightarrow \text{Delete}
\]

\[
\{ g(x, y) \Rightarrow g(a, y), z \Rightarrow h(x, g(x, b), b) \} \quad \Rightarrow \text{Decompose}
\]

\[
\{ x \Rightarrow a, y \Rightarrow y, z \Rightarrow h(x, g(x, b), b) \} \quad \Rightarrow \text{Delete}
\]

\[
\{ x \Rightarrow a, z \Rightarrow h(x, g(x, b), b) \} \quad \Rightarrow \text{Eliminate}
\]

Homework 27 (Programming Assignment)

You are assigned to implement congruence closure in Haskell. Additional instructions can be found on the lecture web page.

Homework 28 (More General Substitutions)

Let \( \sigma \) and \( \sigma' \) be substitutions such that each one is an instance of the other: Formally, we have both \( \sigma \preceq \sigma' \) and \( \sigma \succeq \sigma' \). Show that in this case, there must exist a renaming \( \rho \) (i.e., an injective substitution where \( \text{Ran}(\rho) \subseteq V \)) such that \( \sigma = \rho \sigma' \).