RHEINISCH-
WESTFÄLISCHE
TECHNISCHE
HOCHSCHULE
AACHEN

LEHR- UND FORSCHUNGSGEBIET INFORMATIK II
RWTH Aachen · D-52056 Aachen · GERMANY
http://www-i2.informatik.rwth-aachen.de/lufgi2.html

Prof. Dr. Jürgen Giesl
Darius Dlugosz

Exercises *Functional Programming* – Sheet 8

Solutions will be collected until Thursday, July 12, 2001.

Exercise 1 (1 points)

Identify the free variables of each of the following lambda terms and also apply each of the substitutions $[x/(f\,x)]$, $[y/\lambda y.y\,x]$, $[z/\lambda x.x\,z]$ to each of these terms:

(a) $(\lambda x.x\,y)\,(\lambda y.y)$

(b) $\lambda x\,y.z\,(\lambda z.z\,(\lambda x.y))$

(c) $(\lambda x\,y.x\,z\,(y\,z))\,(\lambda x.y\,(\lambda y.y))$

Exercise 2 (2 points)

Give all possible reduction sequences for $\rightarrow_\beta$ starting with:

(a) $(\lambda x.\mathsf{plus}\,x\,x)\,((\lambda y.\mathsf{times}\,y\,y)\,(\mathsf{plus}\,5\,5))$

(b) $(\lambda g.g\,\mathsf{plus}\,(g\,\mathsf{times}\,(g\,\mathsf{plus}\,5)))\,(\lambda f\,x.f\,x\,x)$

(c) $(\lambda v\,f.f\,(f\,v))\,((\lambda z.z\,z)\,(\lambda z.z\,z))\,((\lambda x\,y.x)\,5)$

Exercise 3 (2 points)

The notion of "δ-reduction" is defined as follows:
A set δ of rules of the form $c\,t_1 \ldots t_n \rightarrow r$ with $c \in \mathcal{C}$, $t_1, \ldots, t_n, r \in \Lambda$ is called a delta-rule set if

(1) $t_1, \ldots, t_n, r$ are closed lambda terms

(2) all t_i are in $\rightarrow_\beta$-normal form and they don't contain any left-hand side of a rule from δ

(3) in δ there exist no two rules $c\,t_1 \ldots t_n \rightarrow r$ and $c\,t_1 \ldots t_m \rightarrow r'$ with $m \geq n$

For such a set δ we define the relation $\rightarrow_\delta$ as the least relation with

- $l \rightarrow_\delta r$ for all $l \rightarrow r \in \delta$

- if $t_1 \to_\delta t_2$, then also $(t_1\, r) \to_\delta (t_2\, r)$, $(r\, t_1) \to_\delta (r\, t_2)$ and $\lambda y.t_1 \to_\delta \lambda y.t_2$ for all $r \in \Lambda$, $y \in \mathcal{V}$.

We denote the combination of β- and δ-reduction with $\to_{\beta\delta}$, i.e., $\to_{\beta\delta} = \to_\beta \cup \to_\delta$.

Find counterexamples to show that all three conditions (1), (2), (3) are necessary for the uniqueness of normal forms of the $\to_{\beta\delta}$-relation. This means if one condition is not valid and the other two hold, uniqueness of normal forms is not guaranteed.

Exercise 4 (2 points)

For each of the following terms write down the weak head normal form and show the reduction steps with the $\to_{\beta\delta}$-relation, where δ is the set of rules from definition 3.3.5:

(a) $\lambda x\, y.(\lambda z.z)\, x\, (\mathsf{plus}\, y\, 1)$

(b) $(\lambda x\, y.x\, (\lambda z.y\, z))\, (((\lambda x\, y.y)\, 8)\, (\lambda x.(\lambda y.y)\, x))$

(c) $(\lambda h.(\lambda x.h\, (x\, x))\, (\lambda x.h\, (x\, x)))\, ((\lambda x.x)\, (\mathsf{plus}\, 1\, 5))$

Exercise 5 (4 points)

Translate the following Haskell-expressions into lambda terms and reduce them to weak head normal form with the $\to_{\beta\delta}$-relation, where δ is the set of rules from definition 3.3.5:

(a) `let not' = \x -> if x == True then False else True in not' True`

(b) `let half = \x -> if x <= 1 then 0 else half (minus x 1) in half 3`

(c)
```
let double = \x -> if x == 0 then 0 else (plus 2 (double (x - 1)))
      in double 2
```

(d) `let fact = \x -> if x <= 0 then 1 else (times (fact (x - 1)) x) in fact 2`

Hint: You can write `(minus x 1)` instead of `(x - 1)`, `(equal x 0)` instead of `x == 0`, etc.

Remark:
Definition 3.3.5 (δ-rules for HASKELL-programs) Let Con be the set of constructors of a HASKELL-program, where Con_n denotes the set of all constructors of arity n. Let δ_0 be the rules of the predefined operators in HASKELL, i.e., δ_0 contains rules such as `plus 1 2` $\to$ `3`, `not True` $\to$ `False`, etc. All these rules have the form $f\, t_1 \ldots t_n \to r$, where $f \in \mathcal{C}_0$ and $t_1, \ldots, t_n, r \in \mathsf{Con}_0$. Because there are not two rules with the same left-hand side, δ_0 is a delta-rule set. The entire set δ is defined as follows:

$$
\begin{aligned}
\delta \;=\;& \delta_0 \cup \\
& \{\mathtt{bot} \to \mathtt{bot}, \\
& \mathtt{isa}_{()}\,\mathtt{tuple}_0 \to \mathtt{True}, \\
& \mathtt{if}\,\mathtt{True} \to \lambda xy.x, \\
& \mathtt{if}\,\mathtt{False} \to \lambda xy.y\} \cup \\
& \{\mathtt{fix}\,t \to t\,(\mathtt{fix}\,t) \mid t \in \Lambda,\ t \text{ closed and in } \to_{\beta\delta}\text{-WHNF}\} \cup \\
& \{\mathtt{isa}_{\underline{\mathsf{constr}}}\,(\underline{\mathsf{constr}}\,t_1\ \ldots\ t_n) \to \mathtt{True} \mid \underline{\mathsf{constr}} \in \mathsf{Con}_n,\ n \geq 0, \\
& \qquad\qquad t_1, \ldots, t_n \in \Lambda,\ t_j \text{ closed}\} \cup \\
& \{\mathtt{isa}_{\underline{\mathsf{constr}}}\,(\underline{\mathsf{constr}'}\,t_1\ \ldots\ t_m) \to \mathtt{False} \mid \underline{\mathsf{constr}'} \in \mathsf{Con}_m,\ \underline{\mathsf{constr}} \neq \underline{\mathsf{constr}'},\ m \geq 0, \\
& \qquad\qquad t_1, \ldots, t_m \in \Lambda,\ t_j \text{ closed}\} \cup \\
& \{\mathtt{argof}_{\underline{\mathsf{constr}}}\,(\underline{\mathsf{constr}}\,t_1\ \ldots\ t_n) \to \mathtt{tuple}_n\,t_1\ \ldots\ t_n \mid \underline{\mathsf{constr}} \in \mathsf{Con}_n,\ n \geq 0, \\
& \qquad\qquad t_1, \ldots, t_n \in \Lambda,\ t_j \text{ closed}\} \cup \\
& \{\mathtt{sel}_{n,i}\,(\mathtt{tuple}_n\,t_1\ \ldots\ t_n) \to t_i \mid n \geq 2,\ i \leq n,\ t_1, \ldots, t_n \in \Lambda,\ t_j \text{ closed}\}
\end{aligned}
$$