

Prof. Dr. Jürgen Giesl
Peter Schneider-Kamp

Exercises *Functional Programming* – Sheet 11

Solutions will be collected until Wednesday, July 13, 2005 in the exercise course.

Exercises can be solved both in English and in German.

Exercise 1 (1 + 1 + 2 points)

For each of the following terms please show the reduction steps of the WHNO-reduction with the $\rightarrow_{\beta\delta}$ -relation up to weak head normal form, where δ is the set of rules from Definition 3.3.5:

- (a) $(\lambda x. \text{if } (\text{eq } x \ 2) \ 3 \ 4) \ 5$
- (b) $(\lambda x y. (\lambda z. \text{plus } x \ z) \ 0) \ 1$
- (c) $(\lambda x y. y(x \ x \ y))(\lambda x y. y(x \ x \ y))((\lambda x. x) (\text{plus } 6 \ 7))$

Exercise 2 (2 + 4 + 2 points)

Please translate the following Haskell-expressions into lambda terms using \mathcal{Lam} and reduce them to weak head normal form by WHNO-reduction with the $\rightarrow_{\beta\delta}$ -relation, where δ is the set of rules from Definition 3.3.5:

- (a) `let not' = \x -> if x == False then True else False in not' False`
- (b) `let square = \x -> x * x in square (square 2)`
- (c) `let inf = Succ inf in inf`

Hint: You can write `(eq x False)` instead of `x == False`, `(times x x)` instead of `(x * x)`, 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 $\text{plus } 1\ 5 \rightarrow 6$, $\text{not True} \rightarrow \text{False}$, etc. All these rules have the form $f\ t_1 \dots t_n \rightarrow r$, where $f \in \mathcal{C}_0$ and $t_1, \dots, t_n, r \in \text{Con}_0$. 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:

$$\begin{aligned} \delta = & \delta_0 \cup \\ & \{\text{bot} \rightarrow \text{bot}, \\ & \text{isa}(_) \text{tuple}_0 \rightarrow \text{True}, \\ & \text{if True} \rightarrow \lambda xy.x, \\ & \text{if False} \rightarrow \lambda xy.y, \\ & \text{fix} \rightarrow \lambda f.f (\text{fix } f)\} \cup \\ & \{\text{isa}_{n\text{-tuple}}(\text{tuple}_n\ t_1 \dots t_n) \rightarrow \text{True} \mid n \in \{0, 2, 3, \dots\}\} \cup \\ & \{\text{isa}_{\text{constr}}(\text{constr}\ t_1 \dots t_n) \rightarrow \text{True} \mid \text{constr} \in \text{Con}_n, n \geq 0, \\ & \quad t_1, \dots, t_n \in \Lambda, t_j \text{ closed}\} \cup \\ & \{\text{isa}_{\text{constr}}(\text{constr}'\ t_1 \dots t_m) \rightarrow \text{False} \mid \text{constr}' \in \text{Con}_m, \text{constr} \neq \text{constr}', m \geq 0, \\ & \quad t_1, \dots, t_m \in \Lambda, t_j \text{ closed}\} \cup \\ & \{\text{argof}_{\text{constr}}(\text{constr}\ t_1 \dots t_n) \rightarrow \text{tuple}_n\ t_1 \dots t_n \mid \text{constr} \in \text{Con}_n, n \in \{0, 2, 3, \dots\}, \\ & \quad t_1, \dots, t_n \in \Lambda, t_j \text{ closed}\} \cup \\ & \{\text{argof}_{\text{constr}}(\text{constr}\ t) \rightarrow t \mid \text{constr} \in \text{Con}_1, t \in \Lambda, t \text{ closed}\} \cup \\ & \{\text{sel}_{n,i}(\text{tuple}_n\ t_1 \dots t_n) \rightarrow t_i \mid n \geq 2, i \leq n, t_1, \dots, t_n \in \Lambda, t_j \text{ closed}\} \end{aligned}$$

Exercise 3 (2 + 2 points)

In the lecture, fix was translated into the following pure lambda-term:

$$\overline{\text{fix}} = (\lambda x y.y (x x y))(\lambda x y.y (x x y))$$

For this translation, $\overline{\text{fix}}\ z \rightarrow_{\beta}^* z(\overline{\text{fix}}\ z)$ holds.

There exists another fix-point-combinator called Y with $Y = \lambda f.(\lambda x.f (x x))(\lambda x.f (x x))$. Please show:

- (a) $Y\ z \not\rightarrow_{\beta}^* z(Y\ z)$
- (b) $Y\ z \leftrightarrow_{\beta}^* z(Y\ z)$