

Prof. Dr. Jürgen Giesl  
Peter Schneider-Kamp  
Stephan Swiderski

## Exercises *Functional Programming* – Sheet 10

Solutions will be collected until Tuesday, July 3, 2007 in the exercise course.

Exercises can be solved both in English and in German.

### Exercise 1 (6 + 5 points)

The data structure `List` is defined by

```
data List a = Nil | Cons a (List a)
```

- (a) Transform the following Haskell-expression to a simple Haskell-expression using the transformation rules (1)-(9) from the lecture. Please give all intermediate expressions and indicate in each step which transformation rule was used.

```
let sumUp s Nil = Cons s Nil
    sumUp s (Cons x xs) = Cons s (sumUp (s+x) xs)
    in sumUp 0 (Cons 1 (Cons 2 Nil))
```

- (b) Consider the data structure `Tree`.

```
data Tree a = Node a (List (Tree a))
```

Transform the following Haskell-expression to a Haskell expression of the form `let ts = ...` using the transformation rules (1), (11), and (12) from the lecture such that `ts` is a pair of functions equivalent to `(treeTree, forestDepth)`.

```
let treeDepth (Node x ts) = 1 + (forestDepth ts)
    forestDepth Nil = 0
    forestDepth (Cons t ts) = max (treeDepth t) (forestDepth ts)
    in treeDepth (Node 1 (Cons (Node 2 Nil) Nil))
```

*Hint:*

You can simplify your results during the transformation as in the example in the lecture:

- (i) `if (isan-tuple exp) then exp1 else exp2` can be replaced by `exp1`.
- (ii) `(\var->exp) exp'` can be replaced by `exp`, where all free occurrences of `var` are replaced by `exp'`.

## Exercise 2 (2 + 3 + 3 points)

Mark the sequences that correspond to leftmost-innermost reductions and leftmost-outermost reductions, respectively.

- (a) Give all possible reduction sequences with  $\rightarrow_\beta$  starting with the following term:

$$(\lambda x y. \text{div } y x) 5 ((\lambda z. \text{times } 3 z) 4)$$

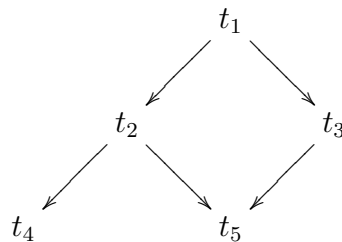
- (b) Give the leftmost-outermost and the leftmost-innermost reduction sequences with  $\rightarrow_\beta$  starting with the following term:

$$(\lambda g. g \text{ times } (g \text{ plus } (g \text{ div } 3))) (\lambda f x. f x 2)$$

- (c) Give all reduction sequences with  $\rightarrow_\beta$  with at most 4 steps starting with the following term:

$$(\lambda x y. y(x x y))(\lambda x y. y(x x y))(\lambda f x. f(f x))$$

*Hint:* You can save space by representing the reduction sequences as (directed) graphs. E.g., one can represent  $t_1 \rightarrow_\beta t_2 \rightarrow_\beta t_4$ ,  $t_1 \rightarrow_\beta t_2 \rightarrow_\beta t_5$ , and  $t_1 \rightarrow_\beta t_3 \rightarrow_\beta t_5$  by



## Exercise 3 (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  $\delta$  contains the following rules:

$$\begin{aligned} \text{isa}_{\text{Nil}} (\text{Cons } 5 \text{ Nil}) &\rightarrow \text{False} \\ \text{if False} &\rightarrow \lambda x y. y \\ \text{div } 9 \ 3 &\rightarrow 3 \\ \text{plus } 6 \ 7 &\rightarrow 13 \end{aligned}$$

- (a)  $(\lambda x. \text{if } (\text{isa}_{\text{Nil}} x) 3 4) (\text{Cons } 5 \text{ Nil})$
- (b)  $(\lambda x y. (\lambda z. \text{div } x z) 3) 9$
- (c)  $(\lambda x y. y(x x y))(\lambda x y. y(x x y))((\lambda x. x) (\lambda x. \text{plus } 6 7))$

## Exercise 4 (2 + 4 + 2 points)

Please translate the following Haskell-expressions into lambda terms using *Λ*am:

- (a) `let and' = \x -> \y -> if x == False then False else y  
in and' True False`
- (b) `let double = \x -> 2 * x  
squareF = \f -> \x -> f (f x)  
in squareF double 2`
- (c) `let inf = Succ inf in inf`

*Hint:* You can write `(eq x False)` instead of `x == False`, `(times x y)` instead of `(x * y)`, etc.