

Prof. Dr. Jürgen Giesl
René Thiemann

Exercises *Functional Programming* – Sheet 11

Solutions will be collected until Thursday, Jan 16, 2003 in the exercise course.

Exercise 1 (5 + 3 + 2 points)

Transform the following Haskell-expressions to simple Haskell-expressions using the transformation rules from the lecture. Additionally, determine in part (b) the semantics of the transformed expression by computing the least fixpoint. You do not need to show every step explicitly!

- (a)

```
let mod 0 _ = 0
    mod _ 0 = 0
    mod x y = if_mod (x <= y) x y
    if_mod True  x y = mod (x-y) y
    if_mod False x _ = x
in \x y -> mod x (y+10)
```
- (b)

```
let (v,w,x,y,z) = (y,z,4,x,w)
in v
```

You can simplify your results during the transformation like it was done in the example in the lecture, so

- $(\backslash \text{var} \rightarrow \text{exp}) \text{ exp}'$ can be replaced by exp' , where all occurrences of var are replaced by exp'
- $\text{sel}_{n,i} (x_1, \dots, x_n)$ can be replaced by x_i
- $\text{if} (\text{isa}_() \text{ exp}) \text{ then } \text{exp1} \text{ else } \text{exp2}$ can be replaced by exp1
- ...

The last simplification rule given above is only valid for type-correct and terminating expressions!

Exercise 2 (1+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 3 (1 + 1.5 + 1.5 points)

Give all possible reduction sequences for \rightarrow_β starting with (a) and (c) and give only the leftmost-outermost and the leftmost-innermost reduction sequence for (b):

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

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

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

Exercise 4 (6 points)

Implement \rightarrow_β and \rightarrow_α in Haskell. Please use the following framework. (Some additional functions and the framework can be downloaded from the website. There, you can for example see the exact definition of `Set`)

```
type Variable = Int
type Constructor = String
data Term = V Variable | C Constructor | Abs Variable Term | Appl Term Term
type Substitution = (Variable,Term)

free_vars :: Term -> Set Variable
apply_subst :: Substitution -> Term -> Term
alpha_reduction :: Set Variable -> Term -> Term
beta_reduction :: Term -> Maybe Term
```

`alpha_reduction set t` should return a term t' , where $t \rightarrow_\alpha^* t'$ such that no bound variable of t' is in `set`. Implement `beta_reduction` in such a way, that `beta_reduction t` delivers `Nothing`, if t is in \rightarrow_β -normal form, and `Just t'` otherwise, where t' is a term with $t \rightarrow_\beta t'$.

Check your implementation against the results of the previous exercises.

Remark: $t_0 \rightarrow_\alpha^* t_n$ iff $t_0 \rightarrow_\alpha \dots \rightarrow_\alpha t_n$ with $n \geq 0$.

Important:

Those of you, who are interested in getting the exercise certificate, should send me an e-mail until Thursday, 16. Januar, 1:30pm. The e-mail is not a binding registration, but will only be used to see how many of you want to participate. Further details will be discussed in the global exercise that Thursday.

Thanks, René.