University of Innsbruck **Institute of Computer Science**
Test 2B **January 21, 2011**

Functional Programming **WS 2010/2011** **LVA 703018**

Solutions

This test consists of three exercises. *Explain your answers.* The available points for each item are written in the margin.

(8) **1.** Reduce the λ-term

$$(\lambda zv.\, v)\ (\lambda z.\, z)\ (\lambda abz.\, a\ (b\ z))\ (\lambda z.\, z)\ (\lambda zv.\, z)\ w$$

to normal form using the leftmost innermost reduction strategy.

Solution.

$$\begin{aligned}
&(\lambda zv.\, v)\ (\lambda z.\, z)\ (\lambda abz.\, a\ (b\ z))\ (\lambda z.\, z)\ (\lambda zv.\, z)\ w\\
&= \underline{(\lambda zv.\, v)\ (\lambda z.\, z)}\ (\lambda abz.\, a\ (b\ z))\ (\lambda z.\, z)\ (\lambda zv.\, z)\ w\\
&\to_\beta \underline{(\lambda v.\, v)\ (\lambda abz.\, a\ (b\ z))}\ (\lambda z.\, z)\ (\lambda zv.\, z)\ w\\
&\to_\beta \underline{(\lambda abz.\, a\ (b\ z))\ (\lambda z.\, z)}\ (\lambda zv.\, z)\ w\\
&\to_\beta (\lambda bz.\, \underline{(\lambda z.\, z)\ (b\ z)})\ (\lambda zv.\, z)\ w\\
&\to_\beta \underline{(\lambda bz.\, b\ z)\ (\lambda zv.\, z)}\ w\\
&\to_\beta (\lambda z.\, \underline{(\lambda zv.\, z)\ z})\ w\\
&\to_\beta \underline{(\lambda zv.\, z)\ w}\\
&\to_\beta \lambda v.\, w
\end{aligned}$$

(8) **2.** Consider the Haskell functions

```
length []      =  0
length (x:xs)  =  length xs + 1

map f []      =  []
map f (x:xs)  =  f x : map f xs
```

Prove by induction that

$$\texttt{length}\ (\texttt{map}\ f\ xs) = \texttt{length}\ xs$$

for all lists xs.

Solution. We use induction over xs to show the property

$$P(xs) = (\texttt{length}\ (\texttt{map}\ f\ xs) = \texttt{length}\ xs)$$

- **Base Case** ($xs = \texttt{[]}$). $P(\texttt{[]})$ is shown by the derivation:

$$\texttt{length}\ (\texttt{map}\ f\ \texttt{[]}) = \texttt{length}\ (\texttt{[]}) = 0$$

- **Step Case** ($xs = z : zs$). The IH is $P(zs) = (\texttt{length}\ (\texttt{map}\ f\ zs) = \texttt{length}\ zs)$. $P(z : zs)$ is shown by the derivation:

$$\begin{aligned}
\texttt{length}\ (\texttt{map}\ f\ (z : zs)) &= \texttt{length}\ (f\ z : \texttt{map}\ f\ zs)\\
&= \texttt{length}\ (\texttt{map}\ f\ zs) + 1\\
&\overset{\text{IH}}{=} \texttt{length}\ zs + 1\\
&= \texttt{length}\ (z : zs)
\end{aligned}$$

(9) **3.** Use type checking to prove that **let** $add = (\lambda xy.\, x + y)$ **in** **if** b **then** $add\ z\ 1$ **else** $add\ z\ 0$ has the type Int with respect to the environment $E = P \cup \{b :: \mathsf{Bool}, z :: \mathsf{Int}\}$.

Solution.

1	$x :: \mathsf{Int}$	assumption
2	$y :: \mathsf{Int}$	assumption
3	$(+) :: \mathsf{Int} \to \mathsf{Int} \to \mathsf{Int}$	ins E
4	$(+)\ x :: \mathsf{Int} \to \mathsf{Int}$	app 3, 1
5	$x + y :: \mathsf{Int}$	app 4, 2
6	$\lambda y.\, x + y :: \mathsf{Int} \to \mathsf{Int}$	abs 2–5
7	$\lambda xy.\, x + y :: \mathsf{Int} \to \mathsf{Int} \to \mathsf{Int}$	abs 1–6
8	$add :: \mathsf{Int} \to \mathsf{Int} \to \mathsf{Int}$	assumption
9	$b :: \mathsf{Bool}$	ins E
10	$z :: \mathsf{Int}$	ins E
11	$add\ z :: \mathsf{Int} \to \mathsf{Int}$	app 8, 10
12	$1 :: \mathsf{Int}$	ins E
13	$add\ z\ 1 :: \mathsf{Int}$	app 11, 12
14	$0 :: \mathsf{Int}$	ins E
15	$add\ z\ 0 :: \mathsf{Int}$	app 11, 14
16	**if** b **then** $add\ z\ 1$ **else** $add\ z\ 0 :: \mathsf{Int}$	ite 9, 13, 15
17	**let** $add = (\lambda xy.\, x + y)$ **in** **if** b **then** $add\ z\ 1$ **else** $add\ z\ 0 :: \mathsf{Int}$	let 7, 8–16