University of Innsbruck | **Institute of Computer Science**
1st Exam | **January 23, 2009**

Functional Programming | **WS 2008/2009** | **LVA 703017**

Solutions

1. Consider the lambda-term $t = (\lambda xy.x\ (\lambda xy.y)\ y\ y)\ (\lambda xyz.z\ x\ y)\ (\lambda x.x)$.

[10] (a) Reduce t stepwise to normal form, using the leftmost innermost strategy.

Solution.

$$\begin{aligned}(\lambda xy.x\ (\lambda xy.y)\ y\ y)\ (\lambda xyz.z\ x\ y)\ (\lambda x.x) &\to_\beta (\lambda y.(\lambda xyz.z\ x\ y)\ (\lambda xy.y)\ y\ y)\ (\lambda x.x)\\ &\to_\beta (\lambda y.(\lambda yz.z\ (\lambda xy.y)\ y)\ y\ y)\ (\lambda x.x)\\ &\to_\beta (\lambda y.(\lambda z.z\ (\lambda xy.y)\ y)\ y)\ (\lambda x.x)\\ &\to_\beta (\lambda y.y\ (\lambda xy.y)\ y)\ (\lambda x.x)\\ &\to_\beta (\lambda x.x)\ (\lambda xy.y)\ (\lambda x.x)\\ &\to_\beta (\lambda xy.y)\ (\lambda x.x)\\ &\to_\beta \lambda y.y\end{aligned}$$

[10] (b) Reduce t stepwise to normal form, using the leftmost outermost strategy.

Solution.

$$\begin{aligned}(\lambda xy.x\ (\lambda xy.y)\ y\ y)\ (\lambda xyz.z\ x\ y)\ (\lambda x.x) &\to_\beta (\lambda y.(\lambda xyz.z\ x\ y)\ (\lambda xy.y)\ y\ y)\ (\lambda x.x)\\ &\to_\beta (\lambda xyz.z\ x\ y)\ (\lambda xy.y)\ (\lambda x.x)\ (\lambda x.x)\\ &\to_\beta (\lambda yz.z\ (\lambda xy.y)\ y)\ (\lambda x.x)\ (\lambda x.x)\\ &\to_\beta (\lambda z.z\ (\lambda xy.y)\ (\lambda x.x))\ (\lambda x.x)\\ &\to_\beta (\lambda x.x)\ (\lambda xy.y)\ (\lambda x.x)\\ &\to_\beta (\lambda xy.y)\ (\lambda x.x)\\ &\to_\beta \lambda y.y\end{aligned}$$

[20] **2.** Consider the OCaml functions

```
let rec (@) xs ys = match xs with []    -> ys
                                | x::xs -> x::(xs @ ys)

let rec rev = function []    -> []
                     | x::xs -> (rev xs) @ [x]
```

Prove by induction that `rev`$(xs$ `@` $ys) =$ (`rev` ys) `@` (`rev` xs) for all lists xs and ys. You may use associativity of '`@`' and the fact that `[]` is a right identity of '`@`', i.e.,

$$(xs \texttt{ @ } ys) \texttt{ @ } zs = xs \texttt{ @ } (ys \texttt{ @ } zs) \qquad (\star)$$
$$xs \texttt{ @ } [] = xs \qquad (\star\star)$$

for all lists xs, ys, and zs.

Solution.

Base Case ($xs = \texttt{[]}$). The base case concludes by the derivation

$$\begin{aligned}
\texttt{rev}(xs \mathbin{\texttt{@}} ys) &= \texttt{rev}(\texttt{[]} \mathbin{\texttt{@}} ys) && \text{(since } xs = \texttt{[]}\text{)}\\
&= \texttt{rev}\ ys && \text{(by definition of @)}\\
&= (\texttt{rev}\ ys) \mathbin{\texttt{@}} \texttt{[]} && \text{(by } (\star\star)\text{)}\\
&= (\texttt{rev}\ ys) \mathbin{\texttt{@}} (\texttt{rev}\ \texttt{[]}) && \text{(by definition of rev)}\\
&= (\texttt{rev}\ ys) \mathbin{\texttt{@}} (\texttt{rev}\ xs)
\end{aligned}$$

Step Case ($xs = x :: zs$). By IH it holds that $\texttt{rev}\ (zs \mathbin{\texttt{@}} ys) = (\texttt{rev}\ ys) \mathbin{\texttt{@}} (\texttt{rev}\ zs)$. The step case concludes by the derivation

$$\begin{aligned}
\texttt{rev}(xs \mathbin{\texttt{@}} ys) &= \texttt{rev}((z :: zs) \mathbin{\texttt{@}} ys) && (xs = z :: zs)\\
&= \texttt{rev}(z :: (zs \mathbin{\texttt{@}} ys)) && \text{(def. of @)}\\
&= (\texttt{rev}(zs \mathbin{\texttt{@}} ys)) \mathbin{\texttt{@}} \texttt{[}z\texttt{]} && \text{(def. of rev)}\\
&= ((\texttt{rev}\ ys) \mathbin{\texttt{@}} (\texttt{rev}\ zs)) \mathbin{\texttt{@}} \texttt{[}z\texttt{]} && \text{(by IH)}\\
&= (\texttt{rev}\ ys) \mathbin{\texttt{@}} ((\texttt{rev}\ zs) \mathbin{\texttt{@}} \texttt{[}z\texttt{]}) && \text{(by } (\star)\text{)}\\
&= (\texttt{rev}\ ys) \mathbin{\texttt{@}} (\texttt{rev}(z :: zs)) && \text{(def. of rev)}\\
&= (\texttt{rev}\ ys) \mathbin{\texttt{@}} (\texttt{rev}\ xs)
\end{aligned}$$

3. Consider the OCaml functions `mem` and `unique`, defined by:

```
let rec mem y = function []    -> false
                       | x::xs -> x = y || mem y xs

let rec unique = function []    -> []
                        | x::xs -> if mem x xs then unique xs
                                               else x :: unique xs
```

[10] (a) Implement a tail-recursive variant of `unique`.

Solution.

```
let unique xs =
 let rec rev acc = function []    -> acc
                          | x::xs -> rev (x::acc) xs
 in
 let rec unique acc = function
  | []    -> rev [] acc
  | x::xs -> if mem x xs then unique acc xs
                         else unique (x::acc) xs
 in
 unique [] xs
```

[10] (b) Use tupling to implement a function `percentage : 'a -> 'a list -> float` that determines for a given element x in a list xs the percentage it constitutes to the full list, e.g.,

`percentage 'a' ['a';'b';'c';'a']` $= 0.5$

Solution.

```
let percentage x ys =
 let rec p x = function
  | []    -> (0,0)
  | y::ys -> let (i,j) = p x ys in if x = y then (i+1,j+1)
                                             else (i,j+1)
 in
 if ys = [] then 0.0
            else let (i,j) = p x ys in float_of_int i /. float_of_int j
```

4. Consider the λ-term $t = (\lambda x.y\ x)\ (\lambda yz.z\ y)\ w$.

[5] (a) Reduce t to normal form.

Solution. $t \to_\beta y\ (\lambda yz.z\ y)\ w$

[5] (b) Give the set $\mathcal{FV}\mathsf{ar}(t)$ of free variables of t.

Solution. $\mathcal{FV}\mathsf{ar}(t) = \{w, y\}$

[5] (c) Give the set $\mathcal{BV}\mathsf{ar}(t)$ of bound variables of t.

Solution. $\mathcal{BV}\mathsf{ar}(t) = \{x, y, z\}$

[5] (d) Give the set $\mathcal{S}\mathsf{ub}(t)$ of all subterms of t.

Solution. $\mathcal{S}\mathsf{ub}(t) = \{y, x, y\ x, \lambda x.y\ x, z, z\ y, \lambda z.z\ y, \lambda yz.z\ y, (\lambda x.y\ x)\ (\lambda yz.z\ y), w, t\}$

5. Consider the typing environment

$$E = \{1 : \mathsf{int},\ +\ : \mathsf{int} \to \mathsf{int} \to \mathsf{int},\ \mathsf{p} : \mathsf{int} \to \mathsf{int} \to \mathsf{pair}(\mathsf{int}, \mathsf{int})\}.$$

[10] (a) Prove the typing judgment $E \vdash \mathbf{let}\ x = 1\ \mathbf{in}\ \mathsf{p}\ x\ (x + x) : \mathsf{pair}(\mathsf{int}, \mathsf{int})$.

Solution.

$$\dfrac{E \vdash 1 : \mathsf{int} \qquad \dfrac{\dfrac{E, x : \mathsf{int} \vdash \mathsf{p} : \mathsf{int} \to \mathsf{int} \to \mathsf{pair}(\mathsf{int}, \mathsf{int}) \quad E, x : \mathsf{int} \vdash x : \mathsf{int}}{E, x : \mathsf{int} \vdash \mathsf{p}\ x : \mathsf{int} \to \mathsf{pair}(\mathsf{int}, \mathsf{int})}\ (\mathrm{app}) \quad \star}{E, x : \mathsf{int} \vdash \mathsf{p}\ x\ (x + x) : \mathsf{pair}(\mathsf{int}, \mathsf{int})}\ (\mathrm{app})}{E \vdash \mathbf{let}\ x = 1\ \mathbf{in}\ \mathsf{p}\ x\ (x + x) : \mathsf{pair}(\mathsf{int}, \mathsf{int})}\ (\mathrm{let})$$

and $\star$ is

$$\dfrac{\dfrac{E, x : \mathsf{int} \vdash (+) : \mathsf{int} \to \mathsf{int} \to \mathsf{int} \quad E, x : \mathsf{int} \vdash x : \mathsf{int}}{E, x : \mathsf{int} \vdash (+)\ x : \mathsf{int} \to \mathsf{int}}\ (\mathrm{app}) \qquad E, x : \mathsf{int} \vdash x : \mathsf{int}}{E, x : \mathsf{int} \vdash (x + x) : \mathsf{int}}\ (\mathrm{app})$$

Solutions

[10] (b) Transform the type inference problem $E \rhd \textbf{let } x = 1 \textbf{ in } \mathsf{p}\ x\ (x + x) : \alpha_0$ into a unification problem.

Solution.

$$E \rhd \textbf{let } x = 1 \textbf{ in } \mathsf{p}\ x\ (x + x) : \alpha_0$$
$$\overset{\mathsf{let}}{\Rightarrow}$$
$$E \rhd 1 : \alpha_1; E, x : \alpha_1 \rhd \mathsf{p}\ x\ (x + x) : \alpha_0$$
$$\overset{\mathsf{con}}{\Rightarrow}$$
$$\mathsf{int} \approx \alpha_1; E, x : \alpha_1 \rhd \mathsf{p}\ x\ (x + x) : \alpha_0$$
$$\overset{\mathsf{app}}{\Rightarrow}$$
$$\mathsf{int} \approx \alpha_1; E, x : \alpha_1 \rhd \mathsf{p}\ x : \alpha_2 \to \alpha_0; E, x : \alpha_1 \rhd (x + x) : \alpha_2$$
$$\overset{\mathsf{app}}{\Rightarrow}$$
$$\mathsf{int} \approx \alpha_1; E, x : \alpha_1 \rhd \mathsf{p} : \alpha_3 \to \alpha_2 \to \alpha_0; E, x : \alpha_1 \rhd x : \alpha_3; E, x : \alpha_1 \rhd (x + x) : \alpha_2$$
$$\overset{\mathsf{con}}{\Rightarrow}$$
$$\mathsf{int} \approx \alpha_1; \mathsf{int} \to \mathsf{int} \to \mathsf{pair}(\mathsf{int}, \mathsf{int}) \approx \alpha_3 \to \alpha_2 \to \alpha_0;$$
$$E, x : \alpha_1 \rhd x : \alpha_3; E, x : \alpha_1 \rhd (x + x) : \alpha_2$$
$$\overset{\mathsf{con}}{\Rightarrow}$$
$$\mathsf{int} \approx \alpha_1; \mathsf{int} \to \mathsf{int} \to \mathsf{pair}(\mathsf{int}, \mathsf{int}) \approx \alpha_3 \to \alpha_2 \to \alpha_0;$$
$$\alpha_1 \approx \alpha_3; E, x : \alpha_1 \rhd (x + x) : \alpha_2$$
$$\overset{\mathsf{app}}{\Rightarrow}$$
$$\mathsf{int} \approx \alpha_1; \mathsf{int} \to \mathsf{int} \to \mathsf{pair}(\mathsf{int}, \mathsf{int}) \approx \alpha_3 \to \alpha_2 \to \alpha_0;$$
$$\alpha_1 \approx \alpha_3; E, x : \alpha_1 \rhd (+)\ x : \alpha_4 \to \alpha_2; E, x : \alpha_1 \rhd x : \alpha_4$$
$$\overset{\mathsf{app}}{\Rightarrow}$$
$$\mathsf{int} \approx \alpha_1; \mathsf{int} \to \mathsf{int} \to \mathsf{pair}(\mathsf{int}, \mathsf{int}) \approx \alpha_3 \to \alpha_2 \to \alpha_0;$$
$$\alpha_1 \approx \alpha_3; E, x : \alpha_1 \rhd (+) : \alpha_5 \to \alpha_4 \to \alpha_2; E, x : \alpha_1 \rhd x : \alpha_5; E, x : \alpha_1 \rhd x : \alpha_4$$
$$\overset{\mathsf{con}}{\Rightarrow}$$
$$\mathsf{int} \approx \alpha_1; \mathsf{int} \to \mathsf{int} \to \mathsf{pair}(\mathsf{int}, \mathsf{int}) \approx \alpha_3 \to \alpha_2 \to \alpha_0;$$
$$\alpha_1 \approx \alpha_3; \mathsf{int} \to \mathsf{int} \to \mathsf{int} \approx \alpha_5 \to \alpha_4 \to \alpha_2; E, x : \alpha_1 \rhd x : \alpha_5; E, x : \alpha_1 \rhd x : \alpha_4$$
$$\overset{\mathsf{con}}{\Rightarrow}$$
$$\mathsf{int} \approx \alpha_1; \mathsf{int} \to \mathsf{int} \to \mathsf{pair}(\mathsf{int}, \mathsf{int}) \approx \alpha_3 \to \alpha_2 \to \alpha_0;$$
$$\alpha_1 \approx \alpha_3; \mathsf{int} \to \mathsf{int} \to \mathsf{int} \approx \alpha_5 \to \alpha_4 \to \alpha_2; \alpha_1 \approx \alpha_5; E, x : \alpha_1 \rhd x : \alpha_4$$
$$\overset{\mathsf{con}}{\Rightarrow}$$
$$\mathsf{int} \approx \alpha_1; \mathsf{int} \to \mathsf{int} \to \mathsf{pair}(\mathsf{int}, \mathsf{int}) \approx \alpha_3 \to \alpha_2 \to \alpha_0;$$
$$\alpha_1 \approx \alpha_3; \mathsf{int} \to \mathsf{int} \to \mathsf{int} \approx \alpha_5 \to \alpha_4 \to \alpha_2; \alpha_1 \approx \alpha_5; \alpha_1 \approx \alpha_4$$