University of Innsbruck | **Institute of Computer Science**
1st Exam | **January 31, 2014**

Functional Programming | **WS 2013/2014** | **LVA 703024**

Solutions

The exam consists of 4 exercises. Explain your answers. You need 50 points to pass.

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

(9) (a) Reduce t to normal form using the leftmost outermost reduction strategy. Give all β-steps!

Solution.

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

(5) (b) Compute the free and the bound variables of t.

Solution. The free variables are $\{x, z\}$. The bound variables are $\{x, y, z\}$.

(5) (c) Give five subterms of t.

Solution. The subterms are t, $(\lambda xyz.x\ z)\ (\lambda xyz.x\ z)\ x$, $(\lambda xyz.x\ z)\ (\lambda xyz.x\ z)$, $\lambda xyz.x\ z$, $\lambda yz.x\ z$, $\lambda z.x\ z$, $x\ z$, x, z.

(6) (d) Give two non-variable subterms of t that are in weak head normal form (WHNF).

Solution. The non-variable subterms of t which are in WHNF are: $\lambda xyz.x\ z$, $\lambda yz.x\ z$, $\lambda z.x\ z$

2. Consider the type for binary trees

```
type 'a t = Empty | Node of ('a t * 'a * 'a t)
```

together with a function to mirror trees

```
let rec mirror = function
  | Empty -> Empty
  | Node (l,x,r) -> Node(mirror r,x,mirror l)
```

Prove by structural induction that for all binary trees t the following property holds:

$$\texttt{mirror}\ (\texttt{mirror}\ t) = t$$

(5) (a) Base case: Show the base case.

Solution. In the base case we have $t = \texttt{Empty}$. Hence

$$\begin{aligned}\texttt{mirror}\ (\texttt{mirror Empty}) &= \texttt{mirror Empty} && \text{(definition of mirror)}\\ &= \texttt{Empty} && \text{(definition of mirror)}\end{aligned}$$

(20) (b) Step case: Identify the property to prove (5 points), the induction hypothesis (5 points), and prove the step case (10 points).

Solution. In the step case we have $t = \texttt{Node}(l, x, r)$. The property to prove is

$$\texttt{mirror}\ (\texttt{mirror}\ (\texttt{Node}(l, x, r))) = \texttt{Node}(l, x, r)\,.$$

The induction hypothesis is

$$\texttt{mirror}\ (\texttt{mirror}\ l) = l$$

and

$$\texttt{mirror}\ (\texttt{mirror}\ r) = r\,.$$

We transform the left-hand side into the right-hand side

$$\begin{aligned}
&\texttt{mirror}\ (\texttt{mirror}\ (\texttt{Node}(l, x, r))) \\
&= \texttt{mirror}\ (\texttt{Node}(\texttt{mirror}\ r, x, \texttt{mirror}\ l)) && \text{(definition of } \texttt{mirror}\text{)} \\
&= \texttt{Node}(\texttt{mirror}\ (\texttt{mirror}\ l), x, \texttt{mirror}\ (\texttt{mirror}\ r)) && \text{(definition of } \texttt{mirror}\text{)} \\
&\overset{\text{IH}}{=} \texttt{Node}(l, x, r)
\end{aligned}$$

3. Consider the function `mirror` from Exercise 2 and the function `merge` defined below.

```
let rec merge xs ys = match (xs,ys) with
 | ([],ys) -> ys
 | (xs,[]) -> xs
 | (x::xs,y::ys) -> if x < y then x::(merge xs (y::ys))
                            else y::(merge (x::xs) ys)
```

(10) (a) Determine for each of the functions `mirror` and `merge` if they are tail recursive or not.

Solution. The function `mirror` is not tail recursive since after the recursive call(s) a tree is built. The function `merge` is not tail recursive since an element is added to the list obtained from the recursive call.

(10) (b) Determine for each of the functions `mirror` and `merge` their runtime by means of O-notation (for an input of size n).

Hint: Perform a worst-case analysis.

Solution. Both functions have runtime $O(n)$. For `mirror` this can be seen since for every subtree the function is recursively called once. For `merge` in each recursive call either of the two lists gets reduced by one element, resulting in (at most) linear runtime as well.

(5) (c) Can the Master Theorem be used to determine the runtime of `mirror`?

Solution. No, since $b \not> 1$ if the tree is not balanced. For balanced trees however we have $b = 2$ and the master theorem gives runtime $\Theta(n)$.

4. Consider the CoreML expression $e = \lambda f x y.x\ y$ and the (empty) typing environment $E = \varnothing$.

(15) (a) Transform the type inference problem $E \rhd e : \alpha_0$ into a unification problem.

Solution.

$$E \rhd \lambda f xy.x\ y : \alpha_0$$
$$\overset{\mathsf{abs}}{\Rightarrow}$$
$$E, f : \alpha_1 \rhd \lambda xy.x\ y : \alpha_2; \alpha_0 \approx \alpha_1 \to \alpha_2$$
$$\overset{\mathsf{abs}}{\Rightarrow}$$
$$E, f : \alpha_1, x : \alpha_3 \rhd \lambda y.x\ y : \alpha_4; \alpha_2 \approx \alpha_3 \to \alpha_4; \alpha_0 \approx \alpha_1 \to \alpha_2$$
$$\overset{\mathsf{abs}}{\Rightarrow}$$
$$E, f : \alpha_1, x : \alpha_3, y : \alpha_5 \rhd x\ y : \alpha_6; \alpha_4 \approx \alpha_5 \to \alpha_6; \alpha_2 \approx \alpha_3 \to \alpha_4; \alpha_0 \approx \alpha_1 \to \alpha_2$$
$$\overset{\mathsf{abs}}{\Rightarrow}$$
$$E, f : \alpha_1, x : \alpha_3, y : \alpha_5 \rhd x : \alpha_7 \to \alpha_6; E, f : \alpha_1, x : \alpha_3, y : \alpha_5 \rhd y : \alpha_7;$$
$$\alpha_4 \approx \alpha_5 \to \alpha_6; \alpha_2 \approx \alpha_3 \to \alpha_4; \alpha_0 \approx \alpha_1 \to \alpha_2$$
$$\overset{\mathsf{con}}{\Rightarrow}$$
$$x \approx \alpha_7 \to \alpha_6; E, f : \alpha_1, x : \alpha_3, y : \alpha_5 \rhd y : \alpha_7; \alpha_4 \approx \alpha_5 \to \alpha_6; \alpha_2 \approx \alpha_3 \to \alpha_4; \alpha_0 \approx \alpha_1 \to \alpha_2$$
$$\overset{\mathsf{con}}{\Rightarrow}$$
$$\alpha_3 \approx \alpha_7 \to \alpha_6; \alpha_5 \approx \alpha_7; \alpha_4 \approx \alpha_5 \to \alpha_6; \alpha_2 \approx \alpha_3 \to \alpha_4; \alpha_0 \approx \alpha_1 \to \alpha_2$$

(10) (b) Write the CoreML expression e in OCaml notation (5 points) and determine its most general type (5 points).

Hint: You can check/compute the type by solving the unification problem from (a).

Solution. The CoreML expression e corresponds to the anonymous function
`fun f x y -> x y : 'a -> ('b -> 'c) -> 'b -> 'c`.