Solutions

1. Consider the lambda-term $t = (\lambda p.p\ (\lambda xy.y))\ ((\lambda xyf.f\ x\ y)\ (\lambda x.x)\ (\lambda x.x))$.

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

Solution.

$$\begin{aligned}
(\lambda p.p\ (\lambda xy.y))\ ((\lambda xyf.f\ x\ y)\ (\lambda x.x)\ (\lambda x.x)) &\to (\lambda p.p\ (\lambda xy.y))\ ((\lambda yf.f\ (\lambda x.x)\ y)\ (\lambda x.x)) \\
&\to (\lambda p.p\ (\lambda xy.y))\ (\lambda f.f\ (\lambda x.x)\ (\lambda x.x)) \\
&\to (\lambda f.f\ (\lambda x.x)\ (\lambda x.x))\ (\lambda xy.y) \\
&\to (\lambda xy.y)\ (\lambda x.x)\ (\lambda x.x) \\
&\to (\lambda y.y)\ (\lambda x.x) \\
&\to \lambda x.x
\end{aligned}$$

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

Solution.

$$\begin{aligned}
(\lambda p.p\ (\lambda xy.y))\ ((\lambda xyf.f\ x\ y)\ (\lambda x.x)\ (\lambda x.x)) &\to (\lambda xyf.f\ x\ y)\ (\lambda x.x)\ (\lambda x.x)\ (\lambda xy.y) \\
&\to (\lambda yf.f\ (\lambda x.x)\ y)\ (\lambda x.x)\ (\lambda xy.y) \\
&\to (\lambda f.f\ (\lambda x.x)\ (\lambda x.x))\ (\lambda xy.y) \\
&\to (\lambda xy.y)\ (\lambda x.x)\ (\lambda x.x) \\
&\to (\lambda y.y)\ (\lambda x.x) \\
&\to \lambda x.x
\end{aligned}$$

[20] **2.** Consider the type

```
type 'a btree = Leaf of 'a | Node of ('a btree * 'a * 'a btree)
```

together with the functions

```
let hd(x::_) = x
let rec leftmost = function Leaf x      -> x
                          | Node(l,_,_) -> leftmost l
let rec flatten = function Leaf x      -> [x]
                         | Node(l,x,r) -> flatten l @ (x :: flatten r)
```

Prove by induction that `hd(flatten` t`)` $=$ `leftmost` t for all trees t. You may use the fact

$$\texttt{hd}(\texttt{flatten}\ t\ \texttt{@}\ xs) = \texttt{hd}(\texttt{flatten}\ t) \qquad (\star)$$

for all trees t and lists xs.

Solution.

Base Case ($t = \texttt{Leaf}\ x$). The base case concludes by the derivation

$$\begin{aligned} \texttt{hd}(\texttt{flatten}(\texttt{Leaf}\ x)) &= \texttt{hd}(\texttt{[}x\texttt{]}) && \text{(def. of } \texttt{flatten}) \\ &= x && \text{(def. of } \texttt{hd}) \\ &= \texttt{leftmost}\ t && \text{(def. of } \texttt{leftmost}) \end{aligned}$$

Step Case ($t = \texttt{Node}(l, x, r)$). By IH we may assume the following two equations:

$$\texttt{hd}(\texttt{flatten}\ l) = \texttt{leftmost}\ l$$
$$\texttt{hd}(\texttt{flatten}\ r) = \texttt{leftmost}\ r$$

The step case concludes by the derivation

$$\begin{aligned} \texttt{hd}(\texttt{flatten}(\texttt{Node}(l, x, r))) &= \texttt{hd}(\texttt{flatten}\ l \mathbin{\texttt{@}} (x :: \texttt{flatten}\ r)) && \text{(def. of } \texttt{flatten}) \\ &= \texttt{hd}(\texttt{flatten}\ l) && \text{(by } (\star)) \\ &= \texttt{leftmost}\ l && \text{(by IH)} \\ &= \texttt{leftmost}\ t && \text{(def. of } \texttt{leftmost}) \end{aligned}$$

3. Consider the OCaml function `replicate`, defined by:

```
let rec replicate m n = if n < 1 then [] else m :: replicate m (n-1)
```

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

Solution.

```
let replicate m n =
 let rec replicate n acc =
  if n < 1 then acc else replicate (n-1) (m::acc)
 in
 replicate n []
```

[10] (b) Implement the function `split` that splits a list into two lists, where the first contains all elements satisfying the given predicate and the second all the others, e.g.,

```
split (fun x -> x <> 0) [1;2;0;3] = ([1;2;3],[0])
```

Solution.

```
let rec split p = function
 | []    -> ([],[])
 | x::xs -> let (l,r) = split p ys in if p x then (x::l,r)
                                              else (l,x::r)
```

4. Consider the λ-term $t = (\lambda x.x)\ (\lambda x.x)\ (\lambda x.x)$.

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

Solution. $t \rightarrow_\beta (\lambda x.x)\ (\lambda x.x) \rightarrow_\beta (\lambda x.x)$

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

Solution. $\mathcal{FV}\mathsf{ar}(t) = \varnothing$

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

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

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

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

[10] **5.** (a) Transform the type inference problem $\varnothing \rhd \lambda x.x\ x : \alpha_0$ into a unification problem.

Solution.

$$\varnothing \rhd \lambda x.x\ x : \alpha_0$$
$$\stackrel{\mathsf{abs}}{\Rightarrow}$$
$$x : \alpha_1 \rhd x\ x : \alpha_2; \alpha_0 \approx \alpha_1 \to \alpha_2$$
$$\stackrel{\mathsf{app}}{\Rightarrow}$$
$$x : \alpha_1 \rhd x : \alpha_3 \to \alpha_2; x : \alpha_1 \rhd x : \alpha_3; \alpha_0 \approx \alpha_1 \to \alpha_2$$
$$\stackrel{\mathsf{cons}}{\Rightarrow}$$
$$\alpha_1 \approx \alpha_3 \to \alpha_2; x : \alpha_1 \rhd x : \alpha_3; \alpha_0 \approx \alpha_1 \to \alpha_2$$
$$\stackrel{\mathsf{cons}}{\Rightarrow}$$
$$\alpha_1 \approx \alpha_3 \to \alpha_2; \alpha_1 \approx \alpha_3; \alpha_0 \approx \alpha_1 \to \alpha_2$$

[10] (b) Solve the following unification problem (if possible).

$$\alpha_1 \approx \alpha_3 \to \alpha_2$$
$$\alpha_1 \approx \alpha_3$$
$$\alpha_0 \approx \alpha_1 \to \alpha_2$$

Solution.

$$\alpha_1 \approx \alpha_3 \to \alpha_2; \alpha_1 \approx \alpha_3; \alpha_0 \approx \alpha_1 \to \alpha_2$$
$$\Rightarrow^{(\mathsf{v}_1)}_{\{\alpha_1/\alpha_3 \to \alpha_2\}}$$
$$\alpha_3 \to \alpha_2 \approx \alpha_3; \alpha_0 \approx (\alpha_3 \to \alpha_2) \to \alpha_2$$

At this point the occur-check fails and hence given unification problem has no solution.