universität innsbruck

Functional Programming | WS 2017/2018 | LVA 703024

EXAM 1 | February 2, 2018

NAME: **MATRICULATION NUMBER:**

1 (a) *base case*

We employ structural induction over xs. The base case, where $xs = \texttt{[]}$, is taken care of by the derivation:

$$\begin{aligned}
\texttt{concat}\ (\texttt{[]} \mathbin{\texttt{++}} ys) &= \texttt{concat}\ ys && \text{(definition of ++)} \\
&= \texttt{[]} \mathbin{\texttt{++}} \texttt{concat}\ ys && \text{(definition of ++)} \\
&= \texttt{concat}\ \texttt{[]} \mathbin{\texttt{++}} \texttt{concat}\ ys && \text{(definition of concat)}
\end{aligned}$$

(b) *induction hypothesis and statement to show in step case*

In the step case $xs = z : zs$ for some element z and list zs. The induction hypothesis (IH) is

$$\texttt{concat}\ (zs \mathbin{\texttt{++}} ys) = \texttt{concat}\ zs \mathbin{\texttt{++}} \texttt{concat}\ ys$$

and the statement we have to show in order to conclude the step case is

$$\texttt{concat}\ ((z : zs) \mathbin{\texttt{++}} ys) = \texttt{concat}\ (z : zs) \mathbin{\texttt{++}} \texttt{concat}\ ys$$

(c) *step case*

We prove the step case by the following derivation

$$\begin{aligned}
\texttt{concat}\ ((z : zs) \mathbin{\texttt{++}} ys) &= \texttt{concat}\ (z : (zs \mathbin{\texttt{++}} ys)) && \text{(definition of ++)} \\
&= z \mathbin{\texttt{++}} \texttt{concat}\ (zs \mathbin{\texttt{++}} ys) && \text{(definition of concat)} \\
&= z \mathbin{\texttt{++}} (\texttt{concat}\ zs \mathbin{\texttt{++}} \texttt{concat}\ ys) && \text{(IH)} \\
&= (z \mathbin{\texttt{++}} \texttt{concat}\ zs) \mathbin{\texttt{++}} \texttt{concat}\ ys && \text{(associativity of ++)} \\
&= \texttt{concat}\ (z : zs) \mathbin{\texttt{++}} \texttt{concat}\ ys && \text{(definition of concat)}
\end{aligned}$$

2 (a) *reduction to normal form*

The contracted redex of each β-step is underlined:

$$\begin{aligned} \underline{A\ K}\ y\ \Omega &\to_\beta \underline{(\lambda x.\, K\ x)\ y}\ \Omega \\ &\to_\beta \underline{K\ y}\ \Omega \\ &\to_\beta \underline{(\lambda y'.\, y)\ \Omega} \\ &\to_\beta y \end{aligned}$$

(b) *weak head normal forms*

The terms A, K, and y are in weak head normal form. The term Ω is not, since it contains a β-redex (Ω itself).

(c) *alpha equivalence*

Complete the following table:

$\equiv_\alpha$	A	K	Ω	y
t_1	✗	✗	✗	✔
t_2	✗	✗	✗	✗
t_3	✔	✗	✗	✗
t_4	✗	✔	✗	✗

3 (a) *equational reasoning*

```
diffs [1,2] = diffs [2] - 1
            = (diffs [] - 2) - 1
            = (0 - 2) - 1
            = -3

rev [1,2,3] [1,2,3] = rev [1,1,2,3] [2,3]
                    = rev [2,1,1,2,3] [3]
                    = rev [3,2,1,1,2,3] []
                    = [3,2,1,1,2,3]

downto 1 2 = []
```

(b) *kinds of recursion*

The only tail recursive function above is `rev`. Moreover, the function `downto` is not tail recursive but guardedly recursive (the recursive call is "guarded" by one application of the list constructor '`:`'). Finally, `diffs` is neither tail recursive nor guardedly recursive.

(c) *a tail recursive implementation*

```
downto m n = go [] m n
  where
    go acc m n
      | m < n = acc
      | otherwise = go (n:acc) m (n + 1)
```

4 (a) *type checking*

1	$x :: \mathsf{Int}$	assumption
2	$y :: \mathsf{Int}$	assumption
3	$1 :: \mathsf{Int}$	ins E
4	$\lambda y.\, 1 :: \mathsf{Int} \to \mathsf{Int}$	abs 2–3
5	$\lambda xy.\, 1 :: \mathsf{Int} \to \mathsf{Int} \to \mathsf{Int}$	abs 1–4

(b) *type inference part 1 – typing constraints*

$$\underline{E \rhd \lambda xy.\, 1 :: \alpha_0}$$
$$\Rightarrow^{(\mathsf{asb})}$$
$$\alpha_0 \approx \alpha_1 \to \alpha_2;$$
$$\underline{E, x :: \alpha_1 \rhd \lambda y.\, 1 :: \alpha_2}$$
$$\Rightarrow^{(\mathsf{asb})}$$
$$\alpha_0 \approx \alpha_1 \to \alpha_2;$$
$$\alpha_2 \approx \alpha_3 \to \alpha_4;$$
$$\underline{E, x :: \alpha_1, y :: \alpha_3 \rhd 1 :: \alpha_4}$$
$$\Rightarrow^{(\mathsf{con})}$$
$$\alpha_0 \approx \alpha_1 \to \alpha_2;$$
$$\alpha_2 \approx \alpha_3 \to \alpha_4;$$
$$\mathsf{Int} \approx \alpha_4$$

(c) *type inference part 2 – unification*

$$\alpha_0 \approx \alpha_1 \to \alpha_2;$$
$$\underline{\alpha_2 \approx \alpha_3 \to \alpha_4};$$
$$\mathsf{Int} \approx \alpha_4$$
$$\Rightarrow^{(\mathsf{v}_1)}_{\{\alpha_2 \mapsto \alpha_3 \to \alpha_4\}}$$
$$\alpha_0 \approx \alpha_1 \to \alpha_3 \to \alpha_4;$$
$$\underline{\mathsf{Int} \approx \alpha_4}$$
$$\Rightarrow^{(\mathsf{v}_2)}_{\{\alpha_4 \mapsto \mathsf{Int}\}}$$
$$\alpha_0 \approx \alpha_1 \to \alpha_3 \to \mathsf{Int};$$
$$\Rightarrow^{(\mathsf{v}_1)}_{\{\alpha_0 \mapsto \alpha_1 \to \alpha_3 \to \mathsf{Int}\}}$$
$$\square$$

The resulting mgu is: $\{\alpha_0 \mapsto \alpha_1 \to \alpha_3 \to \mathsf{Int}, \alpha_2 \mapsto \alpha_3 \to \mathsf{Int}, \alpha_4 \mapsto \mathsf{Int}\}$