universität innsbruck

Functional Programming | WS 2018/2019 | LVA 703024

EXAM 1 | February 1, 2019

NAME: **MATRICULATION NUMBER:**

1 (a) *equational reasoning*

```
countLess 3 [1,2,3,1] = length $ filter (< 3) [1,2,3,1]
                      = length $ 1 : filter (< 3) [2,3,1]          (since 1 < 3)
                      = 1 + length (filter (< 3) [2,3,1])
                      = 1 + length (2 : filter (< 3) [3,1])        (since 2 < 3)
                      = 1 + 1 + length (filter (< 3) [3,1])
                      = 1 + 1 + length (filter (< 3) [1])      (since not 3 < 3)
                      = 1 + 1 + length (1 : filter (< 3) [])       (since 1 < 3)
                      = 1 + 1 + 1 + length (filter (< 3) [])
                      = 1 + 1 + 1 + length []
                      = 1 + 1 + 1 + 0
                      = 3
```

(b) *kinds of recursion*

Neither `length` nor `filter` is tail recursive. According to the lecture `filter` is not be guardedly recursive, since *all* recursive calls have to be guarded by constructors. However, the intention was to have an extension of tail recursion (thus also the alternative name *tail recursion modulo cons*) that allows us to avoid stack overflows. Therefore, `filter` should be guardedly recursive. Due to this mismatch between definition and intention, full points (for this half of the question) are obtained for both answers: that `filter` is guardedly recursive and that it is not.

(c) *a tail recursive implementation*

```
countLess y xs = go 0 y xs
  where
    go acc y [] = acc
    go acc y (x:xs)
      | x < y     = go (1 + acc) y xs
      | otherwise = go acc y xs
```

2 (a)

base case

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

$$\begin{aligned}
\texttt{all}\ p\ (\texttt{[]}\ \texttt{++}\ ys) &= \texttt{all}\ p\ ys && \text{(definition of ++)}\\
&= \texttt{True}\ \texttt{\&\&}\ \texttt{all}\ p\ ys && \text{(definition of \&\&)}\\
&= \texttt{all}\ p\ \texttt{[]}\ \texttt{\&\&}\ \texttt{all}\ p\ ys && \text{(definition of all)}
\end{aligned}$$

(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

$$\texttt{all}\ p\ (zs\ \texttt{++}\ ys) = \texttt{all}\ p\ zs\ \texttt{\&\&}\ \texttt{all}\ p\ ys$$

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

$$\texttt{all}\ p\ ((z : zs)\ \texttt{++}\ ys) = \texttt{all}\ p\ (z : zs)\ \texttt{\&\&}\ \texttt{all}\ p\ ys$$

(c)

step case

We prove the step case by the following derivation

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

3 (a) *conventions*

Using the notational conventions from the lecture we obtain

$$(\lambda x.\, x\ x)\ \lambda yz.\, y\ z$$

(b) *beta-reduction and alpha-renaming*

$$\begin{aligned}
\underline{(\lambda x.\, x\ x)\ (\lambda yz.\, y\ z)} &\to_\beta \underline{(\lambda yz.\, y\ z)\ (\lambda yz.\, y\ z)} \\
&\to_\beta \lambda z.\, (\lambda y\underline{z}.\, y\ \underline{z})\ z \\
&\equiv_\alpha \lambda z.\, \underline{(\lambda yx.\, y\ x)\ z} && \text{(avoid variable capture)} \\
&\to_\beta \lambda z.\, \lambda x.\, z\ x = \lambda zx.\, z\ x
\end{aligned}$$

(c) *weak head normal form*

A λ-term is in WHNF if it is either an abstraction (that is, of shape $\lambda x.\, t$) or the application of a variable to a number (possibly zero) of terms (that is, of shape $x\ t_1\ \ldots\ t_n$).
Therefore, x is a WHNF but $(\lambda x.\, x)\ x$ is not.

(d) *evaluation strategies*

Applicative order reduction reduces $(\lambda x.\, y)\ ((\lambda x.\, x\ x)\ (\lambda x.\, x\ x))$ to itself, resulting in a nonterminating reduction, whereas normal order reduction derives the normal form y.

4 (a) *type checking*

1	$1 :: \mathsf{Int}$	ins E
2	$0 :: \mathsf{Int}$	ins E
3	$x :: \mathsf{Int}$	assumption
4	$\mathsf{True} :: \mathsf{Bool}$	ins E
5	$\lambda x.\, \mathsf{True} :: \mathsf{Int} \to \mathsf{Bool}$	abs 3–4
6	$(\lambda x.\, \mathsf{True})\ 1 :: \mathsf{Bool}$	app 5, 1
7	$\textbf{if}\ (\lambda x.\, \mathsf{True})\ 1\ \textbf{then}\ 1\ \textbf{else}\ 0 :: \mathsf{Int}$	ite 6, 1, 2

(b) *type inference – unification*

$$\mathsf{Bool} \to \mathsf{Bool} \approx \mathsf{Bool} \to \alpha_2;$$
$$\underline{\mathsf{Bool} \to \alpha_0 \approx \alpha_2 \to \alpha_1 \to \alpha_2}$$
$$\Rightarrow^{(\mathsf{d}_2)}$$
$$\mathsf{Bool} \to \mathsf{Bool} \approx \mathsf{Bool} \to \alpha_2;$$
$$\underline{\mathsf{Bool} \approx \alpha_2};$$
$$\alpha_0 \approx \alpha_1 \to \alpha_2$$
$$\Rightarrow^{(\mathsf{v}_2)}_{\{\alpha_2 \mapsto \mathsf{Bool}\}}$$
$$\underline{\mathsf{Bool} \to \mathsf{Bool} \approx \mathsf{Bool} \to \mathsf{Bool}};$$
$$\alpha_0 \approx \alpha_1 \to \mathsf{Bool}$$
$$\Rightarrow^{(\mathsf{t})}$$
$$\underline{\alpha_0 \approx \alpha_1 \to \mathsf{Bool}};$$
$$\Rightarrow^{(\mathsf{v}_1)}_{\{\alpha_0 \mapsto \alpha_1 \to \mathsf{Bool}\}}$$
$$\square$$

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

(c) *type inference – typing constraints*

$$\underline{E \rhd \texttt{filter}\ (\lambda y.\, 1) :: \alpha_0}$$
$$\Rightarrow^{(\mathsf{app})}$$
$$\underline{E \rhd \texttt{filter} :: \alpha_1 \to \alpha_0};$$
$$E \rhd \lambda y.\, 1 :: \alpha_1$$
$$\Rightarrow^{(\mathsf{con})}$$
$$(\alpha \to \mathsf{Bool}) \to \mathsf{List}(\alpha) \to \mathsf{List}(\alpha) \approx \alpha_1 \to \alpha_0;$$
$$\underline{E \rhd \lambda y.\, 1 :: \alpha_1}$$
$$\Rightarrow^{(\mathsf{abs})}$$
$$(\alpha \to \mathsf{Bool}) \to \mathsf{List}(\alpha) \to \mathsf{List}(\alpha) \approx \alpha_1 \to \alpha_0;$$
$$\alpha_1 \approx \alpha_2 \to \alpha_3;$$
$$\underline{E, y :: \alpha_2 \rhd 1 :: \alpha_3}$$
$$\Rightarrow^{(\mathsf{con})}$$
$$(\alpha \to \mathsf{Bool}) \to \mathsf{List}(\alpha) \to \mathsf{List}(\alpha) \approx \alpha_1 \to \alpha_0;$$
$$\alpha_1 \approx \alpha_2 \to \alpha_3;$$
$$\mathsf{Int} \approx \alpha_3$$