

8. Aufgabenblatt vom Samstag, den 14. Dezember 2019 zur Vorlesung

## ALP I: Funktionale Programmierung

**Bearbeiter:** A. Rudolph und F. Formanek

**Tutor:** Stephanie Hoffmann

### Tutorium 06

Abgabe: bis Montag, den 06. Januar 2020, 10:10 Uhr

#### 1. Aufgabe (24 Punkte)

(a) *Behauptung:*  $\text{reverse}(\text{reverse } xs) = xs$

Induktion über Liste  $xs$  der Länge  $n$

**Induktionsanfang:**  $xs = []$

$$\text{reverse}(\text{reverse } []) \stackrel{\text{rev.1}}{=} \text{reverse } [] \stackrel{\text{rev.1}}{=} []$$

**Induktionsvoraussetzung:** für  $xs = xs'$  gilt:

$$\text{reverse}(\text{reverse } xs') = xs'$$

**Induktionsschritt:** Sei  $xs = (x:xs')$

$$\text{reverse}(\text{reverse } (x:xs')) \stackrel{\text{rev.2}}{=} \text{reverse}(\text{reverse } xs' ++ [x])$$

$$\equiv (\text{reverse } [x]) ++ \text{reverse } (\text{reverse } xs')$$

$$\stackrel{\text{rev.2}}{\Leftrightarrow} (\text{reverse } ([ ]) ++ [x]) ++ \text{reverse}(\text{reverse } xs') \stackrel{\text{rev.1}}{\Leftrightarrow}$$

$$([ ] ++ [x]) ++ \text{reverse}(\text{reverse } xs') \stackrel{(++).1}{=} [x] ++ \text{reverse}(\text{reverse } xs')$$

$$[x] ++ \text{reverse}(\text{reverse } xs') \stackrel{\text{nach IV}}{=} [x] ++ xs'$$

$$[x] ++ xs' \equiv (x:xs')$$

Das bedeutet, dass die Behauptung für alle  $xs$  (endliche Listen) gilt.

(b) *Behauptung:*  $\text{reverse}(xs ++ ys) = \text{reverse } ys ++ \text{reverse } xs$

**Induktionsanfang:**  $xs = []$

$$\text{reverse}([ ] ++ ys) = \text{reverse } ys ++ \text{reverse } [ ] \stackrel{\text{rev.1}}{=}$$

$$\text{reverse}([ ] ++ ys) = \text{reverse } ys ++ [ ] \stackrel{(++).1}{=}$$

$$\text{reverse } ys = \text{reverse } ys$$

**Induktionsvoraussetzung:** für  $xs = xs'$  gilt:

$$\text{reverse}(xs' ++ ys) = \text{reverse } ys ++ \text{reverse } xs'$$

**Induktionsschritt:** Sei  $xs = (x:xs')$

$$\text{reverse}((x:xs') ++ ys) = \text{reverse } ys ++ \text{reverse}(x:xs') \stackrel{\text{rev.2}}{=}$$

$$\begin{aligned}
\text{reverse}((x:xs') ++ ys) &= \text{reverse } ys ++ (\text{reverse } xs' ++ [x]) \stackrel{(++).2}{=} \\
\text{reverse}(x:(xs' ++ ys)) &= \text{reverse } ys ++ (\text{reverse } xs' ++ [x]) \stackrel{rev.2}{=} \\
\text{reverse}(xs' ++ ys) ++ [x] &= \text{reverse } ys ++ (\text{reverse } xs' ++ [x]) \stackrel{nachIV}{=} \\
\text{reverse } ys ++ \text{reverse } xs' ++ [x] &= \text{reverse } ys ++ (\text{reverse } xs' ++ [x]) \equiv \\
\mathbf{\text{reverse } ys ++ \text{reverse } xs' ++ [x] = \text{reverse } ys ++ \text{reverse } xs' ++ [x]}
\end{aligned}$$

Das bedeutet, dass die Behauptung für alle xs (endliche Listen) gilt.

(c) *Behauptung:*  $\text{elem } a (xs ++ ys) = \text{elem } a xs \parallel \text{elem } a ys$

**Induktionsanfang:**  $xs = []$

$$\begin{aligned}
\text{elem } a ([] ++ ys) &= \text{elem } a [] \parallel \text{elem } a ys \stackrel{(++).1}{=} \\
\text{elem } a ys &= \text{elem } a [] \parallel \text{elem } a ys \stackrel{elem.1}{=} \\
\text{elem } a ys &= \text{False} \parallel \text{elem } a ys \equiv \\
\text{elem } a ys &= \text{elem } a ys
\end{aligned}$$

**Induktionsvoraussetzung:** für  $xs = xs'$  gilt:

$$\text{elem } a (xs' ++ ys) = \text{elem } a xs' \parallel \text{elem } a ys$$

**Induktionsschritt:** Sei  $xs = (x:xs')$

$$\begin{aligned}
\text{elem } a ((x:xs') ++ ys) &= \text{elem } a (x:xs') \parallel \text{elem } a ys \stackrel{(++).1}{=} \\
\text{elem } a (x:(xs' ++ ys)) &= \text{elem } a (x:xs') \parallel \text{elem } a ys \stackrel{elem.3}{=} \\
\text{elem } a (x:(xs' ++ ys)) &= \text{elem } a xs' \parallel \text{elem } a ys \equiv \\
\text{elem } a (x:(xs' ++ ys)) &= \text{elem } a xs' \equiv \\
\text{elem } a [x] \parallel \text{elem } a (xs' ++ ys) &= \text{elem } a xs' \parallel \text{elem } a ys \stackrel{nachIV}{=} \\
\text{elem } a [x] \parallel (\text{elem } a xs' \parallel \text{elem } a ys) &= \text{elem } a xs' \parallel \text{elem } a ys \equiv \\
\text{elem } a (x:[]) \parallel (\text{elem } a xs' \parallel \text{elem } a ys) &= \text{elem } a xs' \parallel \text{elem } a ys \stackrel{elem.1}{=} \\
\text{False} \parallel (\text{elem } a xs' \parallel \text{elem } a ys) &= \text{elem } a xs' \parallel \text{elem } a ys \equiv \\
\text{elem } a xs' \parallel \text{elem } a ys &= \text{elem } a xs' \parallel \text{elem } a ys
\end{aligned}$$

TODO: Überarbeiten. Ergebnis ist falsch

Das bedeutet, dass die Behauptung für alle xs (endliche Listen) gilt.

(d) *Behauptung:*  $(\text{takeWhile } p \ xs) ++ (\text{dropWhile } p \ xs) = xs$

**Induktionsanfang:**  $xs = []$

$$\begin{aligned}
(\text{takeWhile } p \ []) ++ (\text{dropWhile } p \ []) &= [] \stackrel{takeW.1}{=} \\
[] ++ (\text{dropWhile } p \ []) &= [] \stackrel{dropW.1}{=} \\
[] ++ [] &= [] \stackrel{(++).1}{=}
\end{aligned}$$

$$[] = []$$

**Induktionsvoraussetzung:** für  $xs = xs'$  gilt:  
 $(takeWhile\ p\ xs') ++ (dropWhile\ p\ xs') = xs'$

**Induktionsschritt:** Sei  $xs = (x:xs')$   
 $(takeWhile\ p\ (x:xs')) ++ (dropWhile\ p\ (x:xs')) = (x:xs') \stackrel{takeW.2}{=}$   
 $(x:(takeWhile\ p\ xs')) ++ (dropWhile\ p\ (x:xs')) = (x:xs') \stackrel{dropW.2}{=}$   
 $(x:(takeWhile\ p\ xs')) ++ (dropWhile\ p\ xs') = (x:xs') \stackrel{(++).2}{=}$   
 $x:((takeWhile\ p\ xs') ++ (dropWhile\ p\ xs')) \stackrel{nachIV}{=} (x:xs')$   
 $(x:xs') = (x:xs')$

Das bedeutet, dass die Behauptung für alle  $xs$  (endliche Listen) gilt.

(e) *Behauptung:*  $map\ (f.\ g)\ xs = map\ f\ xs.\ map\ g\ xs$

**Induktionsanfang:**  $xs = []$   
 $map\ (f.\ g)\ [] = map\ f\ [].\ map\ g\ [] \stackrel{map.1}{=}$   
 $[] = []$

**Induktionsvoraussetzung:** für  $xs = xs'$  gilt:  
 $map\ (f.\ g)\ xs' = map\ f\ xs' . map\ g\ xs'$

**Induktionsschritt:** Sei  $xs = (x:xs')$   
 $map\ (f.\ g)\ (x:xs') = map\ f\ (x:xs') . map\ g\ (x:xs') \stackrel{map.2}{=}$   
 $g(f(x)):map(f.\ g)\ xs' = g(f(x)):(map\ f\ xs' . map\ g\ xs') \stackrel{nachIV}{=}$   
 $g(f(x)):(map\ f\ xs' . map\ g\ xs') = g(f(x)):(map\ f\ xs' . map\ g\ xs')$

Das bedeutet, dass die Behauptung für alle  $xs$  (endliche Listen) gilt.