

ALP I: Funktionale Programmierung

Bearbeiter: A. Rudolph und F. Formanek

Tutor: Stephanie Hoffmann

Tutorium 06

Abgabe: bis Montag, den 13. Januar 2020, 10:10 Uhr

1. **Aufgabe** (5 Punkte) *Behauptung:* $\text{foldl } g \ z \ xs = \text{foldr } f \ z \ (\text{reverse } xs)$

Induktionsanfang: Sei $xs = []$

$$\begin{aligned} \text{foldl } g \ z \ [] &= \text{foldr } f \ z \ (\text{reverse } []) \\ &\stackrel{\text{rev.1}}{=} \text{foldl } g \ z \ [] \text{foldr } f \ z \ [] \\ &\stackrel{\text{foldl.1}}{=} z &= \text{foldr } f \ z \ [] \\ &\stackrel{\text{foldr.1}}{=} z &= z \end{aligned}$$

Induktionsvoraussetzung: für xs' gilt:

$$\text{foldl } g \ z \ xs' = \text{foldr } f \ z \ (\text{reverse } xs')$$

Induktionsschritt: Sei $xs = (z:xs')$

$$\begin{aligned} \text{foldl } g \ z \ (x : xs') &= \text{foldr } f \ z \ (\text{reverse } (x : xs')) \\ &\stackrel{\text{foldl.2}}{=} \text{foldl } g \ (g \ z \ x) \ xs' = \text{foldr } f \ z \ (\text{reverse } (x : xs')) \\ &\stackrel{\text{rev.2}}{=} \text{foldl } g \ (g \ z \ x) \ xs' = \text{foldr } f \ z \ (\text{reverse } xs' ++ [x]) \\ &= \text{foldl } g \ (g \ z \ x) \ xs' = f \ (f \ (\text{foldr } f \ z \ (\text{reverse } xs')) \ x) \ z \\ &\stackrel{\text{Def.f}}{=} \text{foldl } g \ (g \ z \ x) \ xs' = g \ z \ (g \ x \ (\text{foldr } f \ z \ (\text{reverse } xs'))) \\ &\stackrel{\text{nach IV}}{=} \text{foldl } g \ (g \ z \ x) \ xs' = g \ z \ (g \ x \ (\text{foldl } g \ z \ xs')) \\ &\stackrel{*}{=} \text{foldl } g \ (g \ z \ x) \ xs' = \text{foldl } g \ (g \ z \ x) \ xs' \end{aligned}$$

* = Ich weiß das sieht ein bisschen komisch aus, aber der Ausdruck ist wirklich äquivalent. Da das aber eher an Beispielen ersichtlich wird, als daran, dass ich hier mindestens 10 Zeilen mehr beweise, lasse ich das so im Raum stehen und akzeptiere, dass ich diese Aufgabe nicht mit voller Punktzahl schaffe.

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

2. **Aufgabe** (5 Punkte) *Behauptung:* $h\ x\ (foldl\ g\ y\ xs) = foldl\ g\ (h\ x\ y)\ xs$

Induktionsanfang: Sei $xs = []$

$$\begin{aligned} h\ x\ (foldl\ g\ y\ []) &= foldl\ g\ (h\ x\ y)\ [] \\ &\stackrel{foldl.1}{=} h\ x\ y &= h\ x\ y \end{aligned}$$

Induktionsvoraussetzung: für xs' gilt:

$$h\ x\ (foldl\ g\ y\ xs') = foldl\ g\ (h\ x\ y)\ xs'$$

Induktionsschritt: Sei $xs = (z:xs')$

$$\begin{aligned} h\ x\ (foldl\ g\ y\ (z : xs')) &= foldl\ g\ (h\ x\ y)\ (z : xs') \\ &\stackrel{foldl.2}{=} h\ x\ (foldl\ g\ (g\ y\ z)\ xs') &= foldl\ g\ (g\ (h\ x\ y)\ z)\ xs' \\ &\equiv h\ x\ (foldl\ g\ (g\ y\ z)\ xs') &= foldl\ g\ (h\ x\ (g\ y\ z))\ xs' \end{aligned}$$

Ab hier weiß ich auch nicht mehr weiter

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

3. **Aufgabe** (4 Bonuspunkte) *Behauptung:* $span\ p\ xs = (takeWhile\ p\ xs, dropWhile\ p\ xs)$

Induktionsanfang: Sei $xs = []$

$$\begin{aligned} span\ p\ [] &= (takeWhile\ p\ [], dropWhile\ p\ []) \\ &\stackrel{span.1}{=} ([], []) &= (takeWhile\ p\ [], dropWhile\ p\ []) \\ &\stackrel{tW.1}{=} ([], []) &= ([], dropWhile\ p\ []) \\ &\stackrel{dW.1}{=} ([], []) &= ([], []) \end{aligned}$$

Induktionsvoraussetzung: für xs' gilt:

$$span\ p\ xs' = (takeWhile\ p\ xs', dropWhile\ p\ xs')$$

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

Fall 1 ($p(x) == \text{False}$):

$$\begin{aligned} span\ p\ (x : xs') &= (takeWhile\ p\ (x : xs'), dropWhile\ p\ (x : xs')) \\ &\stackrel{span.3}{=} ([], (x : xs')) &= (takeWhile\ p\ (x : xs'), dropWhile\ p\ (x : xs')) \\ &\stackrel{tW.3}{=} ([], (x : xs')) &= ([], dropWhile\ p\ (x : xs')) \\ &\stackrel{dW.3}{=} ([], (x : xs')) &= ([], (x : xs')) \end{aligned}$$

Fall 2 ($p(x) == \text{True}$):

$$\begin{aligned}
& \text{span } p (x : xs') &= (\text{takeWhile } p (x : xs'), \text{dropWhile } p (x : xs')) \\
\stackrel{\text{span.2}}{\equiv} (x : ys, zs) &= (\text{takeWhile } p (x : xs'), \text{dropWhile } p (x : xs')) \\
\stackrel{\text{nachIV}}{\equiv} (x : \text{takeWhile } p xs', \text{dropWhile } p xs') &= (\text{takeWhile } p (x : xs'), \text{dropWhile } p (x : xs')) \\
\stackrel{tW.2}{\equiv} (x : \text{takeWhile } p xs', \text{dropWhile } p xs') &= (x : \text{takeWhile } p xs', \text{dropWhile } p (x : xs')) \\
\stackrel{dW.2}{\equiv} (x : \text{takeWhile } p xs', \text{dropWhile } p xs') &= (x : \text{takeWhile } p xs', \text{dropWhile } p xs')
\end{aligned}$$

Falls unklar ist, warum $(x:ys,zs) == (x:\text{takeWhile } p \text{ xs}', \text{dropWhile } p \text{ xs}')$ ist: Laut Definition von span ist $(ys,zs) = \text{span } xs'$ und da span xs' unserer IV entspricht können wir diese dafür einsetzen.

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

4. **Aufgabe** (6 Punkte) *Behauptung:* $\text{map } f(\text{tree2List } t) = \text{tree2List}(\text{mapTree } f t)$

Induktionsanfang 1.1: Sei $t = \text{Nil}$

$$\begin{aligned}
& \text{map } f (\text{tree2List } \text{Nil}) = \text{tree2List}(\text{mapTree } f \text{ Nil}) \\
\stackrel{t2L.1}{\equiv} \text{map } f [] &= \text{tree2List}(\text{mapTree } f \text{ Nil}) \\
\stackrel{mT.1}{\equiv} \text{map } f [] &= \text{tree2List } \text{Nil} \\
\stackrel{t2L.1}{\equiv} \text{map } f [] &= [] \\
\stackrel{\text{map.1}}{\equiv} [] &= []
\end{aligned}$$

Induktionsanfang 1.2: Sei $t = (\text{Leaf } x)$

$$\begin{aligned}
& \text{map } f(\text{tree2List}(\text{Leaf } x)) = \text{tree2List}(\text{mapTree } f (\text{Leaf } x)) \\
\stackrel{t2L.2}{\equiv} \text{map } f [x] &= \text{tree2List}(\text{mapTree } f (\text{Leaf } x)) \\
\stackrel{mT.2}{\equiv} \text{map } f [x] &= \text{tree2List}(\text{Leaf } (f(x))) \\
\stackrel{t2L.2}{\equiv} \text{map } f [x] &= [f(x)] \\
\stackrel{\text{map.2}}{\equiv} [f(x)] &= [f(x)]
\end{aligned}$$

Induktionsvoraussetzung: für $t = (\text{Node } x \text{ lt } \text{rt})$ gilt:

$\text{map } f (\text{tree2List } \text{lt}) = \text{tree2List}(\text{mapTree } f \text{ lt})$

und

$\text{map } f (\text{tree2List } \text{rt}) = \text{tree2List}(\text{mapTree } f \text{ rt})$

Induktionsschritt: Sei $t = (\text{Node } x \text{ } lt \text{ } rt)$

$$\text{map } f \text{ (tree2List (Node } x \text{ } lt \text{ } rt))} = \text{tree2List}(\text{mapTree } f \text{ (Node } x \text{ } lt \text{ } rt))$$

$$\begin{aligned} & \stackrel{LHS}{=} \text{map } f \text{ (tree2List (Node } x \text{ } lt \text{ } rt))} \\ & \stackrel{t2L.3}{=} \text{map } f \text{ (tree2List } lt \text{ } ++ [x] \text{ } ++ \text{tree2List } rt) \\ & \equiv \text{map } f \text{ (tree2List } lt) \text{ } ++ (\text{map } f [x]) \text{ } ++ \text{map } f \text{ (tree2List } rt) \\ & \stackrel{nachIV}{=} \text{tree2List}(\text{mapTree } f \text{ } lt) \text{ } ++ (\text{map } f [x]) \text{ } ++ \text{tree2List}(\text{mapTree } f \text{ } rt) \\ & \stackrel{map.2}{=} \text{tree2List}(\text{mapTree } f \text{ } lt) \text{ } ++ [f(x)] \text{ } ++ \text{tree2List}(\text{mapTree } f \text{ } rt) \\ & \stackrel{RHS}{=} \text{tree2List}(\text{mapTree } f \text{ (Node } x \text{ } lt \text{ } rt)) \\ & \stackrel{mT.3}{=} \text{tree2List}(\text{Node } (f(x)) \text{ (mapTree } f \text{ } lt) \text{ (mapTree } f \text{ } rt)) \\ & \stackrel{t2L.3}{=} \text{tree2List}(\text{mapTree } f \text{ } lt) \text{ } ++ [f(x)] \text{ } ++ \text{tree2List}(\text{mapTree } f \text{ } rt) \end{aligned}$$

Das bedeutet, dass die Behauptung für alle t (endliche Binärbäume) gilt.

5. **Aufgabe** (3 Punkte)

- 1) $\lambda x. \lambda y. (zv)(\lambda x. xz)$
 Inkorrekt, da $\lambda y. (zv)$ keine gültige Aussage ist. Würde hingegen $\lambda y. (zv)$ da stehen, wäre die Aussage richtig.
- 2) $\lambda(x.xzx)x.xyz$
 Inkorrekt, da $\lambda(x.xzx)x$ keine gültigen Variablennamen sind. Würden wir die gesamte Klammer $(x.xzx)$ entfernen, hätten wir einen gültigen Ausdruck.
- 3) $\lambda zy.zxyb.y$
 Inkorrekt, da $b.y$ keine Aussage bilden können. Es fehlt also ein λ vor dem b oder man entfernt den Punkt, um es zu einer gültigen Aussage zu machen.

6. **Aufgabe** (8 Punkte)

- 1) $\lambda a. (\lambda y. ay)$
 In $\lambda y. ay$ ist y gebunden und a frei
 Im gesamten Ausdruck ist a gebunden.
- 2) $(\lambda z. (\lambda x. xz(\lambda x. xz)))z$
 Im Ausdruck $\lambda x. xz$ ist x gebunden und z frei.
 Im Ausdruck $\lambda x. xz(\lambda x. xz)$ ist x gebunden und z frei (wobei es in der Klammer beim Einsetzen nicht ersetzt wird, weil er dort über eine andere Variable gebunden ist).
 Im Ausdruck $\lambda z. \lambda x. xz(\lambda x. xz)$ ist z gebunden.
 Und zum Schluss steht dann einfach noch ein freies z , welches nicht zum ersten Ausdruck gehört.

3) $z((\lambda x.xz)x)(\lambda x.xwz)$

Im Ausdruck $(\lambda x.xwz)$ ist x gebunden und w und z sind frei.

Im Ausdruck $(\lambda x.xz)$ ist x gebunden und z frei

Im Ausdruck $((\lambda x.xz)x)$ ist das rechte x frei.

Im gesamten Ausdruck ist das linke z frei.

4) $\lambda c.zy.zxy(\lambda c.bz)$

Im Ausdruck $(\lambda c.bz)$ ist c gebunden, obwohl es in der expression nicht auftaucht und b und z sind frei Im gesamten Ausdruck sind c , z und y gebunden und x frei.

7. Aufgabe (4 Punkte)

$$\begin{aligned} 1) & (\lambda zy.z(\lambda abc.b(abc))y) (\lambda sz.z) (\lambda sz.s(z)) \\ &= (\lambda zy.z(\lambda bc.b(ybc))) (\lambda sz.z) (\lambda sz.s(z)) \\ &= (\lambda y.(\lambda sz.z) (\lambda bc.b(ybc))) (\lambda sz.s(z)) \\ &= (\lambda y.(\lambda z.z)) (\lambda sz.s(z)) \\ &= (\lambda z.z) \end{aligned}$$

$$\begin{aligned} 2) & (\lambda zy.zy(\lambda ab.b)) (\lambda ab.a) (\lambda ab.b) z y \\ &= (\lambda y.(\lambda ab.a) y (\lambda ab.b)) (\lambda ab.b) z y \\ &= (\lambda y.(\lambda b.y) (\lambda ab.b)) (\lambda ab.b) z y \\ &= (\lambda y.y) (\lambda ab.b) z y \\ &= (\lambda ab.b) z y \\ &= (\lambda b.b) y \\ &= y \end{aligned}$$

8. Aufgabe (2 Punkte)

- 1) Inkorrekt, da die linke Seite sich nicht weiter auflösen lässt und die rechte Seite eine unendliche Verkettung von y wird
- 2) Korrekt, da beide Seiten sich auf eine unendliche Verkettung von y reduzieren lassen.