9. Aufgabenblatt vom Freitag, den 03. Januar 2020 zur Vorlesung

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: foldl g z xs = foldr f z (reverse xs)*

Induktionsanfang: Sei xs = []

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

Induktionsvoraussetzung: für xs' gilt:
foldl g z xs' = foldr f z (reverse xs')

Indukionsschritt: Sei xs = (z:xs')

$$\begin{aligned} & foldl\ g\ z\ (x:xs') = foldr\ f\ z\ (reverse\ (x:xs')) \\ \overset{foldl.2}{\equiv} & foldl\ g\ (g\ z\ x)\ xs' = foldr\ f\ z\ (reverse\ (x:xs')) \\ \overset{rev.2}{\equiv} & foldl\ g\ (g\ z\ x)\ xs' = foldr\ f\ z\ (reverse\ xs' +\!+ [x]) \\ \equiv\ & foldl\ g\ (g\ z\ x)\ xs' = f\ (f\ (foldr\ f\ z\ (reverse\ xs'))\ x)\ z \\ \overset{Def.f}{\equiv} & foldl\ g\ (g\ z\ x)\ xs' = g\ z\ (g\ x\ (foldr\ f\ z\ (reverse\ xs'))) \\ \overset{nachIV}{\equiv} & foldl\ g\ (g\ z\ x)\ xs' = g\ z\ (g\ x\ (foldl\ g\ z\ xs')) \\ \overset{*}{\equiv} & foldl\ g\ (g\ z\ x)\ xs' = foldl\ g\ (g\ z\ x)\ xs' \end{aligned}$$

* = 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 = []

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

Induktionsvoraussetzung: für xs' gilt:
h x (foldl g y xs') = foldl g (h x y) xs'

Indukionsschritt: Sei xs = (z:xs')

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

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,dropWhle p xs)*

Induktionsanfang: Sei xs = []

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

Induktionsvoraussetzung: für xs' gilt:
span p xs' = (takeWhile p xs',dropWhile p xs')

Indukionsschritt: Sei xs = (x:xs')

Fall 1 (p(x) == False):

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

Fall 2 (p(x) == True):

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

Falls unklar ist, warum (x:ys,zs) == (x:takeWhile p xs', dropWhile p xs') ist: Laut Definition von span ist (ys,zs) = span xs' und da san 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: map f(tree2List t) = tree2List(mapTree f t)*

Induktionsanfang 1.1: Sei t = Nil

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

Induktionsanfang 1.2: Sei t = (Leaf x)

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

Induktionsvoraussetzung: für t = (Node x lt rt) gilt:
map f (tree2List lt) = tree2List(mapTree f lt)
und
map f (tree2List rt) = tree2List(mapTree f rt)

Indukionsschritt: Sei t = (Node x lt rt)

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

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

$$\begin{aligned}
&\overset{RHS}{\equiv} tree2List(mapTree\ f\ (Node\ x\ lt\ rt)) \\
&\overset{mT.3}{\equiv} tree2List(Node\ (f(x))\ (mapTree\ f\ lt)\ (mapTree\ f\ rt) \\
&\overset{t2L.3}{\equiv} tree2List(mapTree\ f\ lt)\ +\!+\ [f(x)]\ +\!+\ tree2List(mapTree\ f\ rt)
\end{aligned}$$

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

5. **Aufgabe** (3 Punkte)

   1) λx.λy(zv)(λx.xz)
   Inkorrekt, da λy(zv) keine gültige Aussage ist. Würde hingegen λy.(zv) da stehen, wäre die Aussage richtig.
   2) λ(x.xzx)x.xyz
   Inkorrekt, da λ(x.xzx)x keine gültigen Variablennamen sind. Würden wir die gesamte Klammer (x.xzx) entfernen, hätten wir einen gültigen Ausdruck.
   3) λzy.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) λa.(λy.ay)
   In λy.ay ist y gebunden und a frei
   Im gesamten Ausdruck ist a gebunden.
   2) (λz.(λx.xz(λx.xz)))z
   Im Ausdruck λx.xz ist x gebunden und z frei.
   Im Ausdruck λx.xz(λx.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 λz.λx.xz(λx.xz)) ist z gebunden.
   Und zum Schluss steht dann einfach noch ein freies z, welches nicht zum ersten Ausdruck gehört.

3) z((λx.xz)x)(λx.xwz)
   Im Ausdruck (λx.xwz) ist x gebunden und w und z sind frei.
   Im Ausdruck (λx.xz) ist x gebunden und z frei
   Im Ausdruck ((λx.xz)x) ist das rechte x frei.
   Im gesamten Ausdruck ist das linke z frei.

4) λczy.zxy(λc.bz)
   Im Ausdruck (λc.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)

1) (λzy.z(λabc.b(abc))y) (λsz.z) (λsz.s(z))
   = (λzy.z(λbc.b(ybc))) (λsz.z) (λsz.s(z))
   = (λy.(λsz.z) (λbc.b(ybc))) (λsz.s(z))
   = (λy.(λz.z)) (λsz.s(z))
   = (λz.z)

2) (λzy.zy(λab.b)) (λab.a) (λab.b) z y
   = (λy.(λab.a) y (λab.b)) (λab.b) z y
   = (λy.(λb.y) (λab.b)) (λab.b) z y
   = (λy.y) (λab.b) z y
   = (λab.b) z y
   = (λb.b) y
   = y

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.