

Programmierung

06. Übungsblatt

Zeitraum: 23. – 27. Mai 2016

Übung 1 (AGS 12.4.29 ★)

- Geben Sie einen Kombinator A an, so dass $A t s u \Rightarrow^* s$ für alle Lambdaerme t, s und u .
- Geben Sie einen Kombinator B an, so dass $B t s \Rightarrow^* s t$ für alle Lambdaerme t und s .
- Geben Sie einen Kombinator C an, so dass $C C \Rightarrow_{\beta} C C$.
- Geben Sie einen Kombinator D an, so dass $D \Rightarrow_{\beta} D$.
- Geben Sie einen Kombinator E an, so dass $E E t \Rightarrow^* E t E$ für jeden Lambdaerme t .

Übung 2 (AGS 12.4.32)

- Gegeben sei der λ -Term

$$\langle F \rangle = \left(\lambda f x y z. \langle \text{ite} \rangle (\langle \text{iszero} \rangle (\langle \text{sub} \rangle x y)) \right. \\ \left. \langle \text{add} \rangle y z \right. \\ \left. \langle \text{succ} \rangle (f (\langle \text{pred} \rangle x) (\langle \text{succ} \rangle y) (\langle \text{mult} \rangle \langle 2 \rangle z)) \right)$$

Berechnen Sie schrittweise die Normalform des Terms $\langle Y \rangle \langle F \rangle \langle 6 \rangle \langle 5 \rangle \langle 3 \rangle$. Schreiben Sie für jeden Aufruf von $\langle F \rangle$ jeweils zwei Zeilen: eine in der Sie die Werte der Parameter des Aufrufs protokollieren, und eine in der Sie ihre Auswertung skizzieren. Führen Sie im Rechenprozess zweckmäßige Abkürzungen der λ -Terme ein.

- Gegeben sei die folgende Haskell-Funktion:

```
g :: Int -> Int -> Int
g 0 y = 2 * (y + 1)
g x 0 = 2 * (x + 1)
g x y = 4 + g (x - 1) (y - 1)
```

Geben Sie einen λ -Term $\langle G \rangle$ an, so dass $g = \langle Y \rangle \langle G \rangle$ gilt. Sie dürfen dabei die in der Vorlesung vorgestellten Terme nutzen.

Übung 3 (AGS 12.4.30)

- Berechnen Sie die Normalform des untenstehenden λ -Terms, indem Sie ihn *schrittweise* reduzieren. Geben Sie dabei vor jedem Schritt für die relevanten Teilausdrücke die Mengen der gebunden bzw. frei vorkommenden Variablen an.

$$(\lambda f x. f (f x)) (\lambda y. x) y$$

- Gegeben sei die folgende Haskell-Funktion:

```
g :: Int -> Int -> Int
g m 0 = m
g m 1 = m + 1
g m n = g m (n - 2) + g m (n - 1)
```

Geben Sie einen λ -Term $\langle G \rangle$ an, so dass $g = \langle Y \rangle \langle G \rangle$ gilt.

(c) Gegeben sei der λ -Term

$$\langle F \rangle = \lambda f x y. \langle \text{ite} \rangle (\langle \text{iszero} \rangle y) \\ \langle 1 \rangle \\ \left(\langle \text{mult} \rangle x (f x (\langle \text{pred} \rangle y)) \right).$$

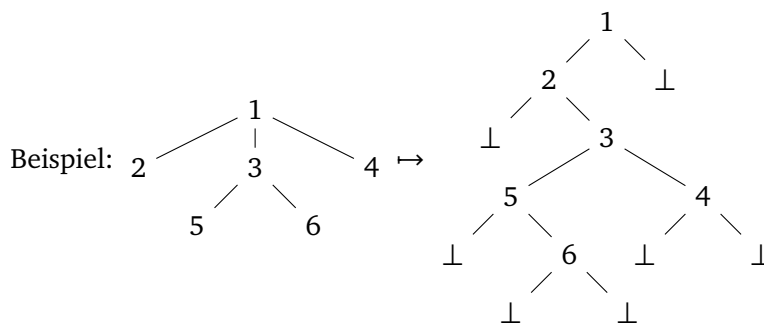
Berechnen Sie die Normalform des Terms $\langle Y \rangle \langle F \rangle \langle 2 \rangle \langle 1 \rangle$. Protokollieren Sie dabei die Berechnung wie in Aufg. 2 angegeben.

Zusatzaufgabe 1

Gegeben sei der folgende Haskell-Code zur Darstellung von Bäumen mit beliebigem Rang, sowie von Binärbäumen.

```
1 data UTree a = UNode a [UTree a]
2 data BTree a = BNil | BNode a (BTree a) (BTree a)
```

Implementieren Sie eine Funktion, welche einen Baum $t :: \text{UTree } a$ als Binärbaum in $\text{BTree } a$ mithilfe der first-child-next-sibling-Methode kodiert. Es soll also jeder Knoten x von t durch einen Knoten y des Binärbaums kodiert werden. Der erste Teilbaum von y soll die Kindknoten von x kodieren, der zweite Teilbaum von y den nächsten Geschwisterknoten von x . Existiert kein Kind, oder kein nächstes Geschwister, wird dies durch BNil kodiert.



Im obigen Beispiel steht \perp für BNil .

Zusatzaufgabe 2 (AGS 12.3.5 ★)

Zeigen Sie durch Induktion über Listen, dass

$$\text{map } g (ys ++ zs) = \text{map } g \text{ ys } ++ \text{map } g \text{ zs}$$

für alle $g :: a \rightarrow b$, und $ys, zs :: [a]$. Nutzen Sie dazu die folgenden Definitionen:

```
1 map :: (a -> b) -> [a] -> [b]
2 map f [] = []
3 map f (x:xs) = f x : map f xs
4
5 (++) :: [a] -> [a] -> [a]
6 [] ++ ys = ys
7 (x:xs) ++ ys = x : (xs ++ ys)
```