Übungsblatt 9

Abgabe der Lösungen: Do, 11.07., 12:00 im Briefkasten am blauen Hochhaus

PRÄSENZAUFGABEN

Einige Annahmen

Wir nehmen einen Typ **Nat** von Konstanten 0,1,2,3,... und die die üblichen Grundoperationen $+,-,*$,**max**,**min**,==,... für **Nat** als gegeben an. Weiter nehmen wir einen Typ **Bool** mit den Konstanten *True* und *False* sowie den Grundoperationen $\wedge,\vee$,*not* etc. und dem **if** _ **then** _ **else** _-Konstrukt als gegeben an. Mit () bezeichnen wir den Typ *unit*; er enthält nur einen einzigen Wert, den wir ebenfalls mit () bezeichnen.

Weiterhin schreiben wir in Beweisen der Gleichheit zweier gegebener Terme s und t die Aussage $s =_{\beta\delta} t$ abgekürzt als $s = t$. Beachten Sie, dass $=_{\beta\delta}$ hierbei die Konvertierbarkeitsrelation bezüglich β- und δ-Reduktionen ist und auch mit $\leftrightarrow^*_{\beta\delta}$ bezeichnet wird.

Übung 1 Listen natürlicher Zahlen

Wir betrachten die folgende algebraische Definition eines Datentyps für Listen natürlicher Zahlen:

data NatList where
NNil : () → **NatList**
NCons : **Nat** → **NatList** → **NatList**

1. Beschreiben Sie in eigenen Worten die durch die folgenden Terme gegebenen Listen natürlicher Zahlen:
   - *NNil*
   - *NCons* 5 *NNil*
   - *NCons* 5 (*NCons* 5 *NNil*)
   - *NCons* 1 (*NCons* 2 (*NCons* 3 (*NCons* 4 *NNil*)))
2. Es können nun Funktionen induktiv über der Struktur von **NatList** definiert werden, beispielsweise:

   sum *NNil* = 0
   sum (*NCons x xs*) = *x* + **sum** *xs*

   (a) Welchen Typ hat **sum**?
   (b) Werten Sie den Term **sum** (*NCons* 4 (*NCons* 89 (*NCons* 21 *NNil*))) aus.
3. Schreiben Sie eine Funktion *element* : **Nat** → **NatList** → **Bool**, so dass *element a xs* = *True* wenn *a* in *xs* vorkommt, und andernfalls *element a xs* = *False*.

Übung 2 Folds über Datentypen

1. Einer der grundlegendsten Datentypen sind die booleschen Wahrheitswerte:

   data Bool where
   True: () → **Bool**
   False: () → **Bool**

   Geben Sie die zu diesem Typ gehörige *fold*-Funktion an; welchen Typ hat die Funktion? Welche wohlbekannte Kontrollstruktur wird durch die Funktion realisiert?

2. Betrachten wir nun den Datentyp der natürlichen Zahlen wie in der Vorlesung:

   data Nat where
   0: () → **Nat**
   Suc: **Nat** → **Nat**

   Geben Sie erneut die zugehörige *fold*-Funktion (die wir ab jetzt mit *foldn* bezeichnen) und ihren Typ an; implementieren Sie nun die Funktionen *add*, *mult*, und *exp* zur Addition, Multiplikation und Exponentiation von natürlichen Zahlen mittels *foldn*.

3. Gegeben seien die Datentypen von einfachen binären Bäumen mit Blättern über einem beliebigen Typ a, gegeben durch die Signatur

   data Tree a **where**
   Leaf: $a \to$ **Tree** a
   Node: **Tree** $a \to$ **Tree** $a \to$ **Tree** a

   sowie allgemeiner Listen (siehe Übung 3) und deren zugehörige *fold*-Funktionen (von nun an *foldt* und *foldL*).

   (a) Geben Sie explizit die Definitionen von *foldt* und *foldL* und ihre Typen an.

   (b) Um ein besseres Verständnis dafür zu bekommen, wie *foldL* funktioniert, bietet es sich an, eine Funktion *scanL* zu schreiben, die alle Zwischenergebnisse, die *foldL* berechnet, in einer Liste sammelt; implementieren Sie *scanL* : $b \to (a \to b \to b) \to$ **List** $a \to$ **List** b.

   (c) Drücken Sie die Funktionen *length* : **List** $a \to$ **Nat** zur Bestimmung der Länge einer Liste und *reverse* : **List** $a \to$ **List** a zum Umdrehen einer Liste mittels *foldL* aus.

   (d) Definieren Sie eine Funktion *front*, die die Blätter eines Baumes in einer Liste speichert.

HAUSAUFGABEN

Übung 3 Allgemeine Listen (10 Punkte)

Wir könnten ähnlich wie in Übung 1 algebraische Datentypen *BoolList*, *NatListList* etc. definieren; das wäre allerdings extrem redundant. Anstelle dessen definieren wir einen *parametrischen* Typ **List** a, wobei a eine Typvariable ist.

data List a **where**
Nil : () $\rightarrow$ **List** a
Cons : $a \rightarrow$ **List** $a \rightarrow$ **List** a

1. Entscheiden Sie für jeden der folgenden Terme, ob er typisierbar ist, und geben Sie gegebenenfalls den zugehörigen Prinzipaltyp an (ohne Berechnung).

   - *Cons True* (*Cons True Nil*)
   - *Cons True* (*Cons* 35 *Nil*)
   - *Cons True*
   - *Cons Nil* (*Cons* (*Cons* 35 *Nil*) *Nil*)
   - *Cons Nil* (*Cons* 35 *Nil*)
   - *Cons Nil*

   Notation: Wir schreiben beispielsweise anstelle von *Cons x* (*Cons y* (*Cons z* (*Cons v Nil*))) kurz [*x*,*y*,*z*,*v*]. Insbesondere entspricht [] der leeren Liste *Nil*.

2. Definieren Sie induktiv die folgenden Funktionen über Listen:

   (a) *length* : **List** $a \rightarrow$ **Nat**, so dass:
   length [] = 0, *length* [*x*] = 1, *length* [*x*,*y*] = 2, ...

   (b) *snoc* : **List** $a \rightarrow a \rightarrow$ **List** a, so dass *snoc xs x* das Element *x* an das rechte Ende von *xs* anhängt; z.B.:
   snoc [] *x* = [*x*], *snoc* [*y*] *x* = [*y*,*x*], *snoc* [*y*,*z*] *x* = [*y*,*z*,*x*], ...

   (c) *reverse* : **List** $a \rightarrow$ **List** a, so dass:
   reverse [] = [], *reverse* [*x*] = [*x*], *reverse* [*x*,*y*] = [*y*,*x*], *reverse* [*x*,*y*,*z*] = [*z*,*y*,*x*], ...
   Hinweis: Verwenden Sie *snoc*.

   (d) *drop* : $a \rightarrow$ **List** $a \rightarrow$ **List** a, so dass:
   drop x [] = [], *drop x* [*x*,*y*] = [*y*], *drop x* [*x*,*y*,*z*,*x*] = [*y*,*z*], ...

   (e) *elem* : $a \rightarrow$ **List** $a \rightarrow$ **Bool**, so dass:
   elem x [] = *False*, *elem x* [*y*,*z*,*q*,*v*] = *False*, *elem x* [*y*,*z*,*z*,*q*,*x*,*z*] = *True*, ...

   (f) *minimum* : **List Nat** $\rightarrow$ **Nat**, so dass:
   minimum [] = 0, *minimum* [3] = 3, *minimum* [3,5,2,3] = 2, ...

   Hinweis: Sie dürfen die Funktionen auch in Haskell implementieren.

Übung 4 Folds über Natürlichen Zahlen (10 Punkte)

Viele Funktionen über den natürlichen Zahlen lassen sich mittels *foldn* (siehe Übung 2.2) im folgenden Schema implementieren:

$$f\ x = g\ (foldn\ \ c\ \ h\ x)$$

wobei *foldn* nicht in g vorkommt. Geben Sie die entsprechenden g, c und h für die folgenden Funktionen an:

1. die Funktion *sqr*, die eine natürliche Zahl quadriert; *Hinweis: Verwenden Sie einen ähnlichen Trick wie bei der Fakultätsfunktion aus der Vorlesung und finden Sie eine geeignete Darstellung des Quadrats einer Zahl!*

2. $last\ \ p$ für ein Prädikat p vom Typ $\mathbf{Nat} \to \mathbf{Bool}$, wobei wir annehmen, dass $p\ 0 = True$; $last\ \ p\ \ n$ berechnet die größte Zahl $m \leq n$, für die $p\ \ m = True$; *Hinweis:* Sie dürfen die Notation $\mathbf{if}\ \ a\ \ \mathbf{then}\ b\ \mathbf{else}\ \ c$ für die Funktion *foldb* von **Bool** verwenden.