Übungen zur "Theorie der Programmierung" - SoSe2015

Freitag 12:15-13:45, Martenstr. 3, Raum 02.133-113

Daniel Hausmann

daniel.hausmann@fau.de

Friedrich-Alexander-Universität Erlangen
Department Informatik

Lehrstuhl 8

June 12, 2015

Übungsblatt 2 - Häufige Fehler

- Übung 12.3):
  Herleitung von $PT(\Gamma, t, a)$ durch falsche Klammerung fehlerhaft; z.B.

  $(((\ \textit{if\_then\_else}\ \ b)\ t)\ e)$, **nicht** $\textit{if\_then\_else}\ \ (b\ (t\ e))$
  $(\textit{le}\ \ a)\ b$, **nicht** $\textit{le}\ \ (a\ b)$;

  Menge von Typgleichungen 'widersprüchlich'; z.B.

  $\{a \to b \doteq nat, \ldots\}$
  $\{a \doteq nat \to nat, a \to b \doteq nat \to nat, \ldots\}$

  Als Folge davon: Unifikation unmöglich.
- Übungen 13.1) und 13.3): Zu kurze Begründungen!
- Übung 13.2): Siehe Tafel.

Primitiv-rekursive Definitionen

Eine **primitiv-rekursive Definition** besteht aus Gleichungen der Form

$$f(\underbrace{c(x_1, \ldots, x_n)}_{\textbf{Konstruktor-Pattern}}) = g(f(x_1), \ldots, f(x_n), x_1, \ldots, x_n)$$

für jeden Konstruktor $c_{/n} \in \Sigma$.

Die Funktion f ist somit als Σ-Homomorphismus von der initialen Σ-Algebra in die (durch primitiv-rekursive Definition von f) gegebene Σ-Algebra eindeutig definiert.

Übung 1 - Listen natürlicher Zahlen

data NatList where

$NNil \;:\; () \rightarrow \mathbf{NatList}$

$NCons : \mathbf{Nat} \rightarrow \mathbf{NatList} \rightarrow \mathbf{NatList}$

1. Beschreiben Sie die folgenden Listen natürlicher Zahlen:
   - $NNil$
   - $NCons\ 5\ NNil$
   - $NCons\ 5\ (NCons\ 5\ NNil)$
   - $NCons\ 1\ (NCons\ 2\ (NCons\ 3\ (NCons\ 4\ NNil)))$

$sum\ NNil = 0$

$sum\ (NCons\ x\ xs) = x + sum\ xs$

2.
   1. Welchen Typ hat sum?
   2. Werten Sie den Term $sum\ (NCons\ 4\ (NCons\ 89\ (NCons\ 21\ NNil)))$ aus.
3. Schreiben Sie eine Funktion $element : \mathbf{Nat} \rightarrow \mathbf{NatList} \rightarrow \mathbf{Bool}$, mit
   $element\ a\ xs = True$ wenn a in xs vorkommt, und
   $element\ a\ xs = False$ sonst.

Übung 2 - Allgemeine Listen

data **List** a **where**

$Nil \;\; : () \rightarrow \textbf{List}\ a$

$Cons : a \rightarrow \textbf{List}\ a \rightarrow \textbf{List}\ a$

1. Entscheiden Sie für jeden der folgenden Terme, ob er typisierbar ist, und geben Sie gegebenenfalls den zugehörigen Prinzipaltyp an.
   - $Cons\ True\ (Cons\ True\ Nil)$
   - $Cons\ True\ (Cons\ False\ \ Nil)$
   - $Cons\ True\ (Cons\ 35\ \ Nil)$
   - $Cons\ True$
   - $Cons\ Nil\ \ (Cons\ (Cons\ 35\ \ Nil)\ \ Nil)$
   - $Cons\ Nil\ \ (Cons\ 35\ \ Nil)$
   - $Cons\ Nil$

   $[a, b, c, d]$ anstelle von $Cons\ a\ (Cons\ b\ (Cons\ c\ (Cons\ d\ Nil)))$.
   Insbesondere: $[]$ anstelle Nil.

Übung 2 - Allgemeine Listen

2. Definieren Sie induktiv die folgenden Funktionen über Listen:
   1. $length\ :\ \mathbf{List}\ \ a \to \mathbf{Nat}$, so dass:

      $length\ \ [] \ = 0,\ length\ \ [a]\ = 1,\ length\ \ [a,b]\ = 2,\ \ldots$
   2. $snoc\ :\ \mathbf{List}\ \ a \to a \to \mathbf{List}\ \ a$, so dass:

      $snoc\ []\ \ x = [x],\ snoc\ [a]\ \ x = [a,x],\ snoc\ [a,b]\ \ x = [a,b,x]$
   3. $reverse\ :\ \mathbf{List}\ \ a \to \mathbf{List}\ \ a$, so dass:

      $reverse\ \ []\ = [],\ reverse\ \ [a,b,c]\ = [c,b,a],\ \ldots$
   4. $drop : a \to \mathbf{List}\ a \to \mathbf{List}\ a$, so dass:

      $drop\ x\ [] = [],\ drop\ x\ [x,y,z,x] = [y,z],\ \ldots$
   5. $elem\ :\ a \to \mathbf{List}\ \ a \to \mathbf{Bool}$, so dass:

      $elem\ x\ [y,z,q,v]\ = False,\ elem\ x\ [y,z,z,q,x,z]\ = True,\ \ldots$
   6. $maximum : \mathbf{List\ Nat} \to \mathbf{Nat}$, so dass:

      $maximum\ [] = 0,\ maximum\ [3] = 3,\ maximum\ [3,5,2,3] = 5$

Übung 3 - Beweise mittels Struktureller Induktion

Beweisen Sie die folgenden Eigenschaften jeweils durch Induktion über der Struktur der Argumentliste. Rechtfertigen Sie hierbei Ihre Schritte und geben Sie jeweils Ihre Induktions-Hypothese klar an.

1. $\forall x\ xs,\ \mathit{length}\ (\mathit{snoc}\ xs\ x) = 1 + \mathit{length}\ xs$
2. $\forall xs,\ \mathit{length}\ (\mathit{reverse}\ xs) = \mathit{length}\ xs$
3. $\forall x\ xs,\ \mathit{reverse}\ (\mathit{snoc}\ xs\ x) = \mathit{Cons}\ x\,(\mathit{reverse}\ xs)$
4. $\forall x\ xs,\ \mathit{reverse}\ (\mathit{reverse}\ xs) = xs$

Hinweis: Wir erinnern daran, dass $s = t$ als $s =_{\beta\delta} t$ zu lesen ist. Ausserdem können Sie jederzeit zuvor bereits bewiesene Eigenschaften verwenden.

Übung 4 - Eine binäre Funktion: Listenkonkatenation

$Nil \oplus ys = ys$
$(Cons\ x\ xs) \oplus ys = Cons\ x\ (xs \oplus ys)$

Wir möchten mittels struktureller Induktion beweisen, dass

$$\forall xs\ \ ys,\ \ length\ \ (xs \oplus ys) = length\ xs + length\ ys$$

1. Über welche Liste(n) sollten wir induzieren, über das erste Argument von $(\_ \oplus \_)$, über das zweite, oder über beide? Warum?
2. Beweisen Sie die oben angegebene Eigenschaft.
3. Beweisen Sie die folgenden Eigenschaften mittels struktureller Induktion.
    1. $\forall xs,\ \ xs \oplus Nil = xs$
    2. $\forall x\ \ xs,\ \ xs \oplus [x] = snoc\ xs\ x$
    3. $\forall a\ \ xs\ \ ys,\ \ snoc\ (xs \oplus ys)\ a = xs \oplus (snoc\ ys\ a)$
    4. $\forall xs\ \ ys,\ \ reverse\ \ (xs \oplus ys) = (reverse\ \ ys) \oplus (reverse\ \ xs)$
    5. $\forall xs\ \ ys\ \ zs,\ \ (xs \oplus ys) \oplus zs = xs \oplus (ys \oplus zs)$
    6. $\forall xs\ \ ys\ \ x,\ \ elem\ x\ (xs \oplus ys) = (elem\ x\ xs)\ \ lor\ \ (elem\ x\ ys)$

Übung 5 - Higher-order-Programmierung

$id\ x = x$
$f\ .\ g = \lambda x\ .\ f\ (g\ x)$
$map\ f\ Nil = Nil$
$map\ f\ (Cons\ x\ xs) = Cons\ (f\ x)\ (map\ f\ xs)$

1. Geben Sie die Prinzipaltypen von id, $(\_\ .\ \_)$ und map an.
2. Was berechnet der Term $maximum\ .\ (map\ length)$? Geben Sie den Prinzipaltypen dieses Terms an.
3. Beweisen Sie die folgenden Eigenschaften mittels struktureller Induktion:
   1. $\forall xs,\ map\ id\ xs = xs$
   2. $\forall f\ g\ xs,\ (map\ f\ .\ map\ g)\ xs = map\ (f\ .\ g)\ xs$
   3. $\forall xs\ ys\ f.\ map\ f\ (xs \oplus ys) = (map\ f\ xs) \oplus (map\ f\ ys)$
   4. $\forall x\ y\ xs.\ x = y \Rightarrow map\ (drop\ x)\ (map\ (Cons\ y)\ xs) = map\ (drop\ x)\ xs$
   5. $\forall x\ y\ xs.\ x \neq y \Rightarrow map\ (drop\ x)\ (map\ (Cons\ y)\ xs) = map\ (Cons\ y)\ (map\ (drop\ x)\ xs)$

Übung 6 - Binäre Bäume

data BinTree a **where**

$Leaf : () \to \mathbf{BinTree}\ a$

$Bin : \mathbf{BinTree}\ a \to a \to \mathbf{BinTree}\ a \to \mathbf{BinTree}\ a$

$mirror\ \ Leaf = Leaf$

$mirror\ \ (Bin\ \ l\ \ a\ \ r) = Bin\,(mirror\ \ r)\ \ a\ \ (mirror\ \ l)$

$inorder\ \ Leaf = Nil$

$inorder\ \ (Bin\ \ l\ \ a\ \ r) = inorder\ \ l \oplus Cons\ a\,(inorder\ \ r)$

Beweisen Sie mittels struktureller Induktion die folgenden Eigenschaften.
Hinweis: Sie sollten jeweils zwei Induktions-Hypothesen aufstellen!
Ausserdem werden Sie vermutlich in vorherigen Übungen bereits bewiesene Eigenschaften benötigen.

1. $\forall t,\ \ mirror\ \ (mirror\ \ t) = t.$
2. $\forall t,\ \ inorder\ \ (mirror\ \ t) = reverse\ \ (inorder\ \ t).$