Formale Grundlagen der Informatik 3

Prof. Dr. Christoph Walther / Visar Januzaj, Nathan Wasser
Technische Universität Darmstadt — Wintersemester 2011/12

Übung 1

Version 1 vom 28.10.2011

Aufgabe 1.1 (Arithmetik)

Aus der Vorlesung kennen Sie die folgende Datenstruktur für natürliche Zahlen:

`structure` $\mathbb{N}$ `<=`
0,
$^{+}(^{-} : \mathbb{N})$

Implementieren Sie die folgenden arithmetischen Operationen:

(a) Addition: `function [infixr,10]` $+(a, b : \mathbb{N}) : \mathbb{N}$

Hinweis (vgl. L 1.0 Primer): `[infixr]` bedeutet, dass a + b anstelle von $+(a, b)$ geschrieben werden muss und dass a + b + c als a + (b + c) interpretiert wird.

(b) Multiplikation: `function [infixr,20]` $\times(a, b : \mathbb{N}) : \mathbb{N}$

Hinweis (vgl. L 1.0 Primer): Da $\times$ mit Priorität 20 stärker bindet als + mit Priorität 10, können Sie $a \times b$ + c statt ($a \times b$) + c schreiben.

(c) Fakultät: `function [postfix]` $!(a : \mathbb{N}) : \mathbb{N}$

Hinweis (vgl. L 1.0 Primer): `[postfix]` bedeutet, dass $(a)!$ anstelle von $!(a)$ geschrieben werden muss.

Aufgabe 1.2 (Listen)

Wir betrachten die aus der Vorlesung bekannte Datenstruktur für Listen:

`structure list[@I] <=`
$\emptyset$,
`[infixr,100]` $::$(`hd : @I, tl : list[@I]`)

Implementieren Sie die folgenden Listen-Operationen:

(a) Das Aneinanderhängen von Listen: `function append`(l : `list[@I]`, k : `list[@I]`) : `list[@I]`

(b) Die Berechnung der Länge einer Liste: `function length`(k : `list[@I]`) : $\mathbb{N}$

(c) Das Spiegeln einer Liste: `function mirror`(k : `list[@I]`) : `list[@I]`.

Beispiel: `mirror`($a :: a :: b :: c :: \emptyset$) = $a :: a :: b :: c :: c :: b :: a :: a :: \emptyset$.

Aufgabe 1.3 (Terme und Typen)

(a) Geben Sie die Typen der folgenden Terme an.

a) `1::ø`

b) `length(ø)`

c) `(1::ø)::ø`

d) `(1::ø)::ø::ø`

e) `((1::ø)::ø)::ø`

f) `length(1::ø)::ø`

(b) Geben Sie die Längen der folgenden Listen an.

a) `1::ø`

b) `ø`

c) `ø::ø`

d) `(1::ø)::ø`

e) `(1::ø)::ø::ø`

f) `((1::ø)::ø)::ø`

g) `length(1::ø)::ø`

Aufgabe 1.4 (Spezifikationen)

Wir wollen nun die Spiegelung einer Liste aus der vorherigen Aufgabe formal spezifizieren. Offenbar ist das Ergebnis der Spiegelung ein Palindrom, d. h. für jede Liste k gilt:

$$\texttt{reverse}(\texttt{mirror}(k)) = \texttt{mirror}(k) \tag{1}$$

Die Prozedur `reverse` dreht dabei die Liste um und kann wie in der ersten praktischen Übung implementiert werden.

Außerdem verdoppelt sich durch die Spiegelung die Länge der Liste, also gilt für jede Liste k:

$$\texttt{length}(\texttt{mirror}(k)) = 2 \times \texttt{length}(k)) \tag{2}$$

Schließlich sollte die ursprüngliche Liste ein Präfix der Ergebnisliste sein. Um dies zu formulieren benutzen wir eine Prozedur `isPrefix`, die testet, ob eine Liste k Präfix einer Liste l ist, d. h. $\texttt{isPrefix}(k, l)$ liefert `true`, falls k ein Präfix von l ist und `false` sonst. Für jede Liste k gilt damit:

$$\texttt{isPrefix}(k, \texttt{mirror}(k)) \tag{3}$$

Untersuchen Sie, welche Kombinationen der Eigenschaften (1), (2) und (3) die Spiegelung einer Liste korrekt spezifizieren. Geben Sie für die anderen Kombinationen jeweils ein Beispiel einer Prozedur, die zwar die Spezifikation erfüllt, aber nicht die Spiegelung einer Liste berechnet.