

Aufgabe 7-1

Diese Aufgabe führt das PVS-Konzept der semantische Subtypen ein. Bitte lesen Sie sich dazu den entsprechenden Abschnitt der PVS-Kurzeinführung durch (Seite 16f).

- a) Definieren Sie in PVS den Typ `even` der geraden natürlichen Zahlen als Subtyp von `nat`.
- b) Definieren Sie eine rekursive Funktion `half(n:even) : RECURSIVE nat`, die eine gerade natürliche Zahl halbiert.
- c) Überprüfen Sie mit `M-x tc` ihre Theorie auf Typkorrektheit. Lassen Sie sich mit `M-x show-tccs` die entstandenen Typkorrektheitsbedingungen anzeigen. Es treten unterschiedliche Arten von TCCs auf: welche? Erklären Sie, warum diese TCCs generiert wurden. Beweisen Sie die TCCs mit `M-x pr`.
- d) Zeigen Sie:

```
half_halves : THEOREM
  FORALL (n:nat): half(2*n) = n
```

Aufgabe 7-2

In PVS ist der Datentyp der linearen Listen über einem beliebigen Elementtyp `T` wie folgt vordefiniert (siehe auch `M-x view-prelude-file`):

```
list [T: TYPE] : DATATYPE
BEGIN
  null : null?
  cons(car:T, cdr:list) : cons?
END list
```

Wir betrachten in dieser Aufgabe lineare Listen über natürlichen Zahlen:

```
NList : TYPE = list[nat]
```

- a) Definieren Sie eine rekursive Funktion `length(l:NList) : RECURSIVE nat`, die die Länge der Liste `l` berechnet.
- b) Definieren Sie eine rekursive Funktion `app(l,k:NList) : RECURSIVE NList`, die zwei Listen `l` und `k` aneinanderhängt (vgl. Vorlesung).

Zeigen Sie, dass die Länge einer mit `app` zusammengesetzten Liste der Summe der Längen der beiden Teillisten entspricht.

- c) Definieren Sie eine rekursive Funktion $\text{map}(f: [\text{nat} \rightarrow \text{nat}], l: \text{NList}) : \text{RECURSIVE NList}$, die eine neue Liste erzeugt und dabei die Funktion f auf jedes Element der Liste l anwendet.

Zeigen Sie, dass die Länge einer Liste bei der Anwendung von map erhalten bleibt.

- d) Definieren Sie eine rekursive Funktion $\text{sum}(l: \text{NList}) : \text{RECURSIVE nat}$, die die Summe der Elemente der Liste l errechnet.

Definieren Sie ferner eine Funktion $\text{double}(l: \text{NList}) : \text{NList}$, die aus der Liste l eine neue Liste erzeugt, in der der Wert jedes Elements verdoppelt ist.

Zeigen Sie, dass die Summe der Elemente der so „verdoppelten“ Liste gleich der zweifachen Summe der Elemente der ursprünglichen Liste ist.

Hinweis: Beweisen Sie jeweils auch die entstehenden TCCs!

Aufgabe 7-3

Definieren Sie die in der Vorlesung vorgestellte Funktion rev zur Listenumkehr in PVS. Sie können sich dabei auf den in PVS vordefinierten Listentyp $\text{list}[T]$ beziehen.

Beweisen Sie die entstehenden TCCs!

- a) Zeigen Sie:

```
rev_rev : PROPOSITION
  FORALL (l: list[T]): rev(rev(l)) = l
```

- b) Definieren Sie wie in der Vorlesung die iterative Variante rev_2 . Zeigen Sie die Gleichheit der beiden Funktionen.

Aufgabe 7-4

- a) Definieren Sie in PVS eine rekursive Funktion $\text{mod}(n: \text{nat}, k: \text{posnat})^1$ zur Berechnung des Rests einer natürlichen Zahl n bei der Division durch k .

- b) Zeigen Sie:

```
n, m : VAR nat
k    : VAR posnat

mod_defn : PROPOSITION
  EXISTS m: n = m*k + mod(n, k)
```

¹Beachte den Typ des zweiten Arguments!