

Aufgabe 8-1

Diese Aufgabe behandelt einfache rekursive Funktionen auf natürlichen Zahlen sowie deren Eigenschaften, die mit Hilfe einfacher Induktionsbeweise gezeigt werden sollen.

Geben Sie bei der Definition der rekursiven Funktionen jeweils eine geeignete Maßfunktion an. Beweisen Sie auch alle auftretenden Typkorrektheitsbedingungen!

- a) Definieren Sie eine Funktion $\text{sum}(n:\text{nat}) : \text{RECURSIVE nat}$, die die Summe der natürlichen Zahlen von 0 bis n berechnet:

$$\text{sum}(n) = \sum_{i=0}^n i$$

Zeigen Sie: $\text{sum}(n) = \frac{1}{2}n * (n + 1)$

- b) Definieren Sie nun eine Funktion $\text{sumF}(n:\text{nat}, f : [\text{nat} \rightarrow \text{nat}]) : \text{RECURSIVE nat}$, die die Summe der Funktionswerte von f an den Stellen von 0 bis n berechnet:

$$\text{sumF}(n, f) = \sum_{i=0}^n f(i)$$

Zeigen Sie:

$$\begin{aligned} \text{sumF}(n, \text{id}) &= \text{sum}(n) \\ \text{sumF}(n, \lambda x. f(x) + g(x)) &= \text{sumF}(n, f) + \text{sumF}(n, g) \end{aligned}$$

id ist die Identitätsfunktion ($\text{id}(n) = n$) und in PVS vordefiniert.

- c) Definieren Sie die Funktionen $\text{square}(n:\text{nat}) : \text{nat}$ und $\text{cube}(n:\text{nat}) : \text{nat}$ mit $\text{square}(n) = n^2$ und $\text{cube}(n) = n^3$.

Zeigen Sie:

$$\begin{aligned} \text{sumF}(n, \text{square}) &= \frac{1}{6}n * (n + 1) * (2n + 1) \\ \text{sumF}(n, \text{cube}) &= \text{square}(\text{sum}(n)) \end{aligned}$$

- d) Definieren Sie eine Funktion $\text{exp2}(n:\text{nat}) : \text{RECURSIVE nat}$, mit $\text{exp2}(n) = 2^n$.
Zeigen Sie: $\text{sumF}(n, \text{exp2}) = \text{exp2}(n + 1) - 1$.
- e) Verallgemeinern Sie die Funktion sumF und definieren in PVS eine rekursive Funktion sumFI mit der Signatur $\text{sumFI}(m:\text{nat}, n:\text{nat}, f : [\text{nat} \rightarrow \text{nat}]) : \text{RECURSIVE nat}$, so dass $\text{sumFI}(m, n, f)$ die Summe der Funktionswerte von f in dem Intervall zwischen m und n bildet:

$$\text{sumFI}(m, n, f) = \sum_{i=m}^n f(i)$$

Zeigen Sie $\text{sumFI}(0, n, f) = \text{sumF}(n, f)$.

Beweisen Sie ferner:

$$\begin{aligned}\text{sumFI}(m, n, \lambda i. x * f(i)) &= x * \text{sumFI}(m, n, f) \\ \text{sumFI}(m, n, \lambda i. f(i + x)) &= \text{sumFI}(m + x, n + x, f) \\ |\text{sumFI}(m, n, f)| &\leq \text{sumFI}(m, n, \lambda i. |f(i)|)\end{aligned}$$

Hinweis: Bei der letzten Aussage benötigen Sie vermutlich als Hilfslemma die sogenannte „Dreiecksungleichung“:

$$|x + y| \leq |x| + |y|$$

Aufgabe 8-2

Definieren Sie eine rekursive Funktion $\text{iterate}(f: [\text{nat} \rightarrow \text{nat}], n: \text{nat}) : \text{RECURSIVE} [\text{nat} \rightarrow \text{nat}]$ in PVS, die eine gegebene Funktion f genau n -mal iteriert:

$$\text{iterate}(f, n)(x) = \underbrace{f(\dots f(f(x))\dots)}_{n\text{-mal}}$$

- a) Beweisen Sie nach der Typprüfung die entstandenen TCCs.
b) Zeigen Sie (\circ ist die Funktionskomposition und in PVS bereits vordefiniert):

```
iterate_compose : THEOREM
  (iterate(f,n) o iterate(f,m))(x) = iterate(f,n+m)(x)
```

- c) Zeigen Sie: Die m -fache Iteration der Funktion $\lambda x. x + n$ entspricht der Multiplikation $n \cdot m$.

```
iterate_add : THEOREM
  iterate((LAMBDA (x): x + n), m)(0) = n * m
```

- d) Durch Iteration welcher Funktion erhält man die Exponentiation n^m ? Formulieren Sie eine entsprechende Aussage und beweisen Sie diese. (Die Exponentiation n^m ist in PVS ebenfalls vordefiniert.)

Aufgabe 8-3

- a) Definieren Sie in PVS eine rekursive Funktion $ggT(x, y)$, die den größten gemeinsamen Teilers zweier *positiver* natürlicher Zahlen nach dem Differenzen-Algorithmus berechnet:

$$ggT(x, y) = \begin{cases} x & \text{falls } x = y \\ ggT(x - y, y) & \text{falls } x > y \\ ggT(x, y - x) & \text{sonst} \end{cases}$$

- b) Zeigen Sie mittels *Maßinduktion*: $ggT(x, y) = ggT(y, x)$

Hinweise:

- Wagemutige können die in PVS vordefinierte lexikographische Ordnung `lex2` zusammen mit der auf Ordinalzahlen definierten Relation `<` als `MEASURE` benutzen. Schlaue definieren eine einfachere Maßfunktion!
- Beweise durch Maßinduktion werden in PVS mit dem Befehl

(MEASURE-INDUCT+ `measure vars`)

eingeleitet, wobei `measure` eine geeignete Maßfunktion ist, die freie Variablen aus `vars` enthält. Genauerer siehe `HELP` Kommando.

Aufgabe 8-4

Die transitive Hülle einer binären Relation über einem Typ `T` kann in PVS wie folgt auch als *induktive Relation* definiert werden:

```
x, y : VAR T
R    : VAR [T, T -> bool]

TC(R)(x, y): INDUCTIVE bool =
  R(x, y) OR (EXISTS z: TC(R)(x, z) AND TC(R)(z, y))
```

- a) Zeigen Sie, dass diese Formalisierung der transitiven Hülle tatsächlich die gewünschten Eigenschaften besitzt, nämlich (i) transitiv ist, (ii) `R` umfasst und (iii) die kleinste solche Relation ist (vgl. Aufgabe 7-2).
- b) Zeigen Sie, dass diese Formalisierung der transitiven Hülle äquivalent zu der aus Aufgabe 7-2 ist.