

Konzepte höherer Programmiersprachen Sommersemester 2005

Übungsblatt 6

Aufgabe 1

Viele numerische Verfahren basieren darauf, dass man den Grenzwert einer unendlichen Folge approximiert. Dies lässt sich mit Hilfe von streams realisieren.

- a. Definieren Sie eine Funktion

```
iterate: ('a -> 'a) -> 'a -> 'a stream
```

die zu einer Funktion f und einem Anfangswert x die Folge $x, f(x), f(f(x)), \dots$ liefert.

- b. Definieren Sie eine Funktion

```
within: float -> float stream -> float
```

die zu einer reellen Zahl $\epsilon > 0$ und einer Folge x_0, x_1, \dots die erste Zahl x_i mit $|x_{i+1} - x_i| < \epsilon$ berechnet.

- c. Benutzen Sie `iterate` und `within` zur Implementierung einer Funktion

```
sqroot: float -> float
```

die zu jeder Zahl $a > 0$ die Quadratwurzel \sqrt{a} approximiert: Es gilt $\sqrt{a} = \lim_{n \rightarrow \infty} x_i$, wobei $x_0 > 0$ und $x_{i+1} = (a/x_i + x_i)/2$.

- d. Implementieren Sie auf ähnliche Art eine Approximation für die Exponentialfunktion: Für alle $x \in \mathbb{R}$ gilt $e^x = \sum_{n=0}^{\infty} x^n/n!$.

Aufgabe 2

Manche streams kann man implementieren, indem man eine geeignete Rekursionsgleichung für den stream aufstellt. Sei etwa `powers_2` der stream aller Zweierpotenzen (in aufsteigender Reihenfolge). Dann gilt

```
powers_2 = cons 1 (add powers_2 powers_2)
```

wobei

```
add: int stream -> int stream -> int stream
```

eine Funktion ist, die zwei (unendliche) streams elementweise addiert.

- Implementieren Sie die Funktion `add`.
- Implementieren Sie den stream `powers_2` gemäß obiger Rekursionsgleichung.
- Stellen Sie eine ähnliche Rekursionsgleichung für den stream der Fibonaccizahlen auf und implementieren Sie ihn.
- Implementieren Sie eine Funktion

```
merge: int stream -> int stream -> int stream
```

die zwei echt aufsteigende Folgen so ‘vereinigt’, dass wieder eine echt aufsteigende Folge entsteht.

- Verwenden Sie die Funktion `merge`, um den stream aller Zahlen der Form $2^i 3^j 5^k$ in aufsteigender Reihenfolge zu generieren.

.

Aufgabe 3

Eine alternative Implementierung für streams erhält man durch

```
type 'a stream = Nil
              | Cons of unit -> 'a * 'a stream
```

Definieren Sie den stream der Primzahlen mit dieser alternativen Implementierung. Welche der beiden Implementierungen ist effizienter?