

Programmierung und Modellierung, SS 2008
Übungsblatt 8

Abgabe: bis Di 17. Juni 2008 (nachts)

Besprechung am Do-Di 5-10. Juni

Am Mittwoch, 18.06.2008, werden ab 14:30 Uhr in der Vorlesung das System- und das Softwareentwicklungspraktikum vorgestellt. Bitte unbedingt erscheinen!

Aufgabe 8-1 Warteschlangen

SML

Eine Warteschlange ist eine Datenstruktur, zu der man Elemente hinzufügen und von der man hinzugefügte Elemente wieder auslesen kann. Dabei wird das zuerst hinzugefügte Element auch zuerst wieder ausgegeben (engl. *first in first out* = FIFO). Definieren Sie einen Typ `'a queue` von Warteschlangen mit Elementen aus `'a`, der der folgenden Spezifikation genügt.

<code>empty : 'a queue</code>	Erzeugt eine leere Warteschlange.
<code>isEmpty : 'a queue -> bool</code>	Liefert wahr gdw. Warteschlange leer.
<code>enqueue : 'a * 'a queue -> 'a queue</code>	Fügt ein Element hinten an eine Warteschlange an.
<code>dequeue : 'a queue -> 'a * 'a queue</code>	Liefert ein Paar bestehend aus erstem Element und Rest der Warteschlange. Ist die Warteschlange leer wird die Ausnahme <code>EmptyQueue</code> ausgelöst.

- a) Wählen Sie zuerst **type** `'a queue = 'a list`.
- b) Was ist die Laufzeit von n enqueue- gefolgt von n dequeue-Operationen?
- c) Nun sollen die Operationen enqueue und dequeue im Durchschnitt in konstanter Zeit laufen, d.h., eine Folge von n enqueue/dequeue-Operationen soll in Zeit linear in n bearbeitet werden. Implementieren Sie die Warteschlange durch 2 Listen, also
type `'a queue = 'a list * 'a list`,
eine Liste für den vorderen, eine für den hinteren, umgedrehten Teil der Warteschlange.

Aufgabe 8-2 Polynome als Koeffizientenlisten

SML

Das Polynom mit Koeffizienten $k_n, k_{n-1}, \dots, k_2, k_1, k_0$ ist die Funktion

$$x \mapsto k_n \cdot x^n + k_{n-1} \cdot x^{n-1} + \dots + k_2 \cdot x^2 + k_1 \cdot x + k_0$$

die auch nach dem so genannten Horner-Schema berechnet werden kann:

$$x \mapsto (k_0 + x \cdot (k_1 + x \cdot (k_2 + x \cdot \dots (k_{n-1} + x \cdot (k_n + x \cdot 0)) \dots)))$$

Sei **type** `polynom = real list`

- a) Definieren Sie eine Funktion `appPoly` vom Typ `polynom -> real -> real`, die zu einer Liste $[k_0, k_1, \dots, k_{n-1}, k_n]$ von Koeffizienten die Funktion zur Berechnung des Polynoms liefert. Das Polynom mit der leeren Liste von Koeffizienten sei die Funktion $x \mapsto 0.0$.
- b) Definieren Sie `appPoly` noch einmal, diesmal **mit Hilfe der Faltungsfunktionen**.
- c) Definieren Sie eine Funktion `derivePoly` vom Typ `polynom -> polynom`, die die Ableitung eines Polynoms berechnet.
- d) Definieren Sie eine Funktion `newtonPoly` vom Typ `polynom * real -> real`, die, gegeben ein Polynom und einen Startwert, eine Nullstelle des Polynoms approximiert. (Siehe Aufgabe 6-4.)

Aufgabe 8-3 Gesetze für Listenoperationen

Beweis

Seien $f : 'a \rightarrow 'b$ und $p : 'b \rightarrow \text{bool}$ reine Funktionen, d.h., ohne Seiteneffekte wie Ausgabe oder Manipulation von Referenzen und Ausnahmen. Zeigen Sie durch Induktion über die Länge der Liste `l`, dass

$$\text{filter } p \ (\text{map } f \ l) = \text{map } f \ (\text{filter } (p \circ f) \ l).$$

Betrachten Sie dazu die Programme im Substitutionsmodell, applikative Auswertungsreihenfolge.