

Informatik II, SoSe 2008
Übungsblatt 11

Abgabe: keine. Das Blatt dient der persönlichen Klausurvorbereitung.

Besprechung 10.07.08 - 15.07.08

Aufgabe 11-1 SML
 Suchbäume

Die Datei `11-1.sml` enthält folgende Definition des Datentyps `tree`, zur Representation von binären Suchbäumen mit `int` Knotenmarkierungen.

```
datatype tree = Node of int * tree * tree  
              | Leaf;
```

- a) Schreiben Sie eine Funktion `insertInTree` vom Typ `int * tree -> tree`, die einen Knoten (mit der `int` Markierung womit die Funktion aufgerufen wird) in einen binären Suchbaum einfügt.
- b) Schreiben Sie ausserdem eine Funktion `listToTree` vom Typ `int list -> tree` die aus einer Liste von Integers einen binären Suchbaum erzeugt. Verwenden Sie dabei die Funktion `insertInTree`.
- c) Ist Ihre Implementierung von `listToTree` (direkt oder indirekt) endrekursiv? Wenn nicht, schreiben Sie ihre Implementierung entsprechen um.
- d) Implementieren Sie `listToTree` nochmal, diesmal als Aufruf einer Faltungsfunktion.
- e) Wie können Sie Ihr Programm erweitern, so dass es Listen sortieren kann? Implementieren Sie die fehlenden Funktionen!

Aufgabe 11-2 Papier
 Strukturelle Induktion

Aufbauend auf die vorige Aufgabe: Wir schreiben $k \in t$ falls k eine Knotenmarkierung des binären Suchbaums t ist. Wir wollen beweisen, dass `insertInTree k t` tatsächlich k in t einfügt und nicht etwas verschlampt.

Zeigen Sie durch strukturelle Induktion über t , dass $k \in \text{insertInTree } k \ t$. Verwenden Sie darin die applikative Auswertung.

Aufgabe 11-3 SML
 Funktionale while-Schleife

In SML kann man mit Hilfe von Funktionen höherer Ordnung und Rekursion Schleifenkonstrukte selbst definieren. Die **while**-Schleife implementiert man als

```
fun fwhile condition step state = ...
```

Dabei ist `state` zu Beginn der Startzustand, z.B. ein Tupel von Werten. Ist die Bedingung `condition` auf dem Zustand erfüllt, wird der Schleifenrumpf `step` ausgeführt, der den nächsten Zustand erzeugt. Ist die Bedingung nicht erfüllt, wird die Schleife nicht weiter ausgeführt sondern der Zustand zurückgegeben.

Betrachten Sie folgendes imperatives Programm mit **while**-Schleife.

```
val zero = let val i = ref 255
           in while !i >= 17 do i := !i - 17;
           !i
           end;
```

Um dieses Programm funktional mit `fwhile` zu implementieren, genügt als Zustand der in der Referenz `i` abgelegte Wert. Dies ist die funktionale Implementierung von obigem Programm:

```
val zero = fwhile (fn i => i >= 17)
                  (fn i => i - 17)
                  255
```

a) Implementieren Sie `fwhile`.

b) Implementieren Sie mit Hilfe von `fwhile` die Funktion $f(a) = \min \left\{ n \mid a < \sum_{i=1}^n \frac{1}{i} \right\}$ in SML. $f(a)$ berechnet die Länge des minimalen Anfangsstücks der harmonischen Reihe, das a überschreitet.

Aufgabe 11-4 SML (schwierig!) Listenverarbeitung

Das kartesische Produkt `cartesian` zweier Listen ist eine Liste von Paaren. Jedes Paar besteht aus einem Element der ersten und einem Element der zweiten Liste. Jede mögliche Paarkombination ist im kartesischen Produkt enthalten. Z.B. ist

```
cartesian ["a", "b"] [1, 2, 3]
= [ ("a", 1), ("a", 2), ("a", 3), ("b", 1), ("b", 2), ("b", 3) ]
```

Implementieren Sie `cartesian` ohne Rekursion oder Schleifen, allein unter Verwendung der Listenverarbeitungsfunktionen. [Hinweis: das Kommando **signature** `L = LIST`; verschafft Ihnen im SML-Interpreter einen schnellen Überblick über die Listenverarbeitungsfunktionen.]

Aufgabe 11-5 Papier Typinferenz

Inferieren Sie per Hand oder im Kopf den allgemeinsten Typ der folgenden Funktionen:

```
fun curry f x y = f (x, y)
```

```
fun uncurry f (x, y) = f x y
```

```
fun K x y = x
```

```
fun S x y z = (x z) (y z)
```

```
fun fix f x = f (fix f) x
```

```
fun fix' f = f (fix' f)
```