8. Übungsblatt zu Programmierung (WS 1999/2000)

Andreas Rossberg, Prof. Gert Smolka

Abgabe: Mittwoch, 15. Dezember 1999 vor der Vorlesung

Hinweis: Die Aufgabe 8.3 braucht nur schriftlich abgegeben zu werden. Die Aufgaben 8.5 bis 8.12 brauchen *nicht* abgegeben zu werden.

Aufgabe 8.1: Indizierter Listenhomomorphismus (4 Punkte)

Schreiben Sie eine Prozedur

```
mapi: (int * 'a -> 'b) -> int -> 'a list -> 'b list
```

die ähnlich `map` eine Prozedur auf jedes Element einer Liste, gepaart mit einem fortlaufenden Index, anwendet. Der Startindex kann dabei frei gewählt werden. So soll beispielsweise

```
mapi f 3 [a,b,c,d] = [f(3,a), f(4,b), f(5,c), f(6,d)]
```

sein.

Aufgabe 8.2: Terme (5 Punkte)

Gegeben sei folgende Deklaration für Terme über beliebigen Typen:

```
datatype 'a term = T of 'a * 'a term list
```

(a) Schreiben Sie eine Prozedur `postfixp: 'a term -> 'a list`, die die Postfixprojektion eines Terms berechnet.

(b) Schreiben Sie eine Prozedur `domain: 'a term -> int list list`, die zu einem Term den zugeordneten Baumbereich als Liste in Präfixordnung liefert. Tipp: Benutzen Sie `mapi` aus Aufgabe 8.1.

Aufgabe 8.3: Ausnahmen (7 Punkte)

Bestimmen Sie für folgende Ausdrücke, ob sie zu einem Wert oder zu einem Ausnahme-Packet auswerten. Geben Sie zudem den jeweiligen Wert mit allgemeinstem Typ, oder die entsprechende Ausnahme des Packets an.

(a) `(raise Empty) handle Overflow => "schmonz" | Empty => "schmunk"`

(b) `4 + ((raise Empty) handle _ => 4)`

(c) `(raise Subscript)::(raise Empty) handle Overflow => []`

(d) `(fn x => fn y => raise Empty) 3`

(e) `(raise Empty) handle _ => raise Overflow`

(f) `(fn e => raise e) Subscript`

(g) `(fn e => e) Subscript`

Aufgabe 8.4: Suchbäume (8 Punkte)

Gegeben sei folgende Typdeklaration für binäre Suchbäume:

```
datatype 'a stree = E | N of 'a * 'a stree * 'a stree
```

(a) Zeichnen Sie den Baum

```
val t = N(5, N(3, N(1,E,N(2,E,E)), N(4,E,E)), N(7,N(6,E,E),E))
```

(b) Schreiben Sie eine Prozedur

```
append: 'a stree * 'a stree -> 'a stree
```

Sie soll zwei sortierte Suchbäume, bei denen alle Elemente des zweiten Baumes größer sind als jedes Element des ersten, zu einem sortierten Suchbaum zusammenhängen. Sie soll dies tun, indem sie den zweiten Baum am Stück an die richtige Stelle des ersten Baumes einfügt.

Zum Beispiel: $1 \swarrow 3 \searrow 5 \qquad 6 \swarrow 7 \searrow 9 \quad \rightarrow \quad 1 \swarrow 3 \searrow 5 \searrow 7\,(6 \swarrow 7 \searrow 9)$

(c) Schreiben Sie eine Prozedur

```
delete: ('a * 'a -> order) -> 'a * 'a stree -> 'a stree
```

die ein Element aus einen sortierten Suchbaum löscht. Sie soll dazu eine frei wählbare Ordnung benutzen. Falls das Element nicht enthalten ist, soll eine Ausnahme `Delete` geworfen werden, die Sie geeignet deklarieren müssen. Hinweis: Verwenden Sie `append` aus Aufgabe (b).

(d) Schreiben Sie eine weitere Prozedur

```
delete': ('a * 'a -> order) -> 'a * 'a stree -> 'a stree
```

die ebenfalls ein Element löscht, die aber den Baum einfach unverändert zurückgibt, falls das Element nicht gefunden wird. Schreiben Sie diese Prozedur ohne Rekursion, nur mit Hilfe von `delete`!

Aufgabe 8.5: Wiederholung: Induktive Definition

Die Menge M sei durch folgende Regeln definiert:

- $2 \in M$, $3 \in M$, $5 \in M$
- wenn $k,m,n \in M$, dann ist $k+m+n \in M$

(a) Geben Sie M mit Hilfe einer rekursiven Gleichung an.

(b) Definieren Sie M durch Inferenzregeln.

(c) Geben Sie eine Ableitung für die Aussage $16 \in M$ und den dazu gehörigen Ableitungsbaum an.

(d) Welche Zahlen $n \in \mathbb{N}$ sind nicht Element von M?

Aufgabe 8.6: Wiederholung: Einfache Listenoperationen

(a) Schreiben Sie eine Prozedur `last: 'a list -> 'a`, die das letzte Element einer Liste liefert. Im Falle der leeren Liste soll die Ausnahme `Empty` geworfen werden.

(b) Schreiben Sie eine ähnliche Prozedur `last': 'a list -> 'a option`, die ebenfalls das letzte Element einer Liste liefert. Im Falle der leeren Liste soll das Ergebnis `NONE` sein.

Aufgabe 8.7: Wiederholung: Typisierung von Ausdrücken

Welche der folgenden Deklarationen sind typkorrekt und welcher allgemeinste Typ ist jeweils für die deklarierten Bezeichner herleitbar? Sagen Sie bei nicht korrekten Deklarationen in einem Satz, was der Fehler ist.

(a) `val (a,b) = ([#2 (3.4, "Slurp", 2<0)], fn x => (x,x))`

(b) `fun f(a,b) = (a+1, [false,b,true], a>b)`

(c) `fun f(a,b,c) = (a@[], a@b, [c], "")`

(d) `val x = fn (a,b) => a-1 < 2 andalso b`

(e) `fun f g (x,y) = g x y`

Aufgabe 8.8: Wiederholung: Typen und Werte

Geben Sie jeweils einen SML-Ausdruck (keine Deklaration) an, die den folgenden *allgemeinsten* Typ hat. Verwenden Sie dabei nur Konstanten, Tupel, Listen und Funktionsabstraktion (`fn`) – keine Typconstraints oder Bezeichner aus Standardmodulen!

(a) `string list list * (bool * (real * bool))`

(b) `(int -> bool) list`

(c) `'a * 'a list -> 'a list`

(d) `'a -> 'b -> 'a list * 'b list`

(e) `'a * 'b -> ('a * int) list`

Aufgabe 8.9: Wiederholung: Falten

Erinnern Sie sich an die Standardprozedur

```
foldl: ('a * 'b -> 'b) -> 'b -> 'a list -> 'b
```

zum Falten von Listen. Schreiben Sie damit eine nicht-rekursive Prozedur

```
count: ''a * ''a list -> int
```

die die Anzahl der Auftreten eines Elements in der Liste berechnet.

Aufgabe 8.10: Wiederholung: Mergesort

Gegeben seien die Prozeduren

```
split: int list -> int list * int list
merge: int list * int list -> int list
```

Die Prozedur `split` zerlegt Listen mit mindestens zwei Elementen in zwei kürzere Listen, während `merge` zwei sortierte Listen zu einer ebenfalls sortierten Liste verschmilzt. Schreiben Sie mit Hilfe dieser beiden Prozeduren eine Prozedur

```
sort: int list -> int list
```

die Listen ganzer Zahlen sortiert.

Aufgabe 8.11: Wiederholung: Induktionsbeweis

Wir betrachten Listen als mathematische Objekte gemäß

$$\mathcal{L}(X) = \{\emptyset\} \cup X \times \mathcal{L}(X)$$

Wir schreiben nil für $\emptyset$ und $x :: w$ für (x, w) und definieren induktiv die Funktion

$$\begin{aligned} \_ @ \_ :\ & \mathcal{L}(X) \times \mathcal{L}(X) \to \mathcal{L}(X) \\ \text{nil} @ v \quad &= \quad v \\ (x :: w) @ v \quad &= \quad x :: (w @ v) \end{aligned}$$

Beweisen Sie durch Induktion, dass

$$w @ \text{nil} = w$$

für alle $w \in \mathcal{L}(X)$ gilt.

Aufgabe 8.12: Wiederholung: Graphen

Gegeben sei der Graph $G = (V, E)$ mit

$$V = \{1, 2, 3, 4, 5, 6, 7\}$$
$$E = \{(1,5), (1,7), (2,1), (2,3), (3,5), (6,2), (6,3), (6,4), (6,7)\}$$

(a) Zeichnen Sie G. Was sind die Quellen und Senken?

(b) Ist G zyklisch, zusammenhängend, stark zusammenhängend, ein Wald, ein Baum?

(c) Was sind die Vorgänger und Nachfolger des Knoten 2? Was ist sein Ingrad und Ausgrad?

(d) Geben Sie den von 2 aus erreichbaren Teilgraphen G_2 formal an.