12. Übungsblatt zu Programmierung (WS 1999/2000)

Andreas Rossberg, Prof. Gert Smolka

Abgabe: Freitag, 28. Januar 2000 vor der Vorlesung

Hinweis: Für die Aufgaben 12.1 bis 12.2 arbeiten Sie bitte mit SML/NJ, für die restlichen mit Moscow ML.

Alle Signaturen und vorgegebenen Strukturen finden Sie unter `http://www.ps.uni-sb.de/courses/info1-ws99/uebung/sml/blatt12/`.

Aufgabe 12.1: Endliche Funktionen (8 Punkte)

In der Programmierung werden endliche Funktionen (*maps*) benutzt, um beispielsweise das Nachschlagen bestimmter Informationen auf Grund eines eindeutigen *Schlüssels* zu ermöglichen.

Im folgenden sollen Sie endliche Funktionen als abstrakten Typ implementieren. Verwenden Sie dazu die folgende Signatur, die einen abstrakten Typkonstruktor für erweiterbare endliche Funktionen $A \to B$ beschreibt:

```
signature MAP =
  sig
    type (''a,'b) map
    val empty:   (''a,'b) map
    val insert:  (''a,'b) map * ''a * 'b -> (''a,'b) map
    val lookup:  (''a,'b) map * ''a -> 'b option
    val domain:  (''a,'b) map -> ''a list
  end
```

Die Operationen sollen folgendes Verhalten haben:

$$
\begin{aligned}
empty &= \emptyset \\
insert(f,a,b) &= f + \{a \mapsto b\} \\
lookup(f,a) &= \text{if } a \in \mathrm{Dom} f \text{ then SOME}(f(a)) \text{ else NONE} \\
domain(f) &= xs
\end{aligned}
$$

wobei *xs* eine Liste ist, deren Elemente gerade die Elemente von Dom$f$ sind. Die Reihenfolge dieser Elemente ist unbestimmt.

(a) Schreiben Sie eine Struktur `Map` mit der Signatur `MAP`, die endliche Funktionen zur Verfügung stellt.

(b) Deklarieren Sie einen Bezeichner `m:(int,int) Map.map`, der die Funktion

$$\{1 \mapsto 1,\ 5 \mapsto 3,\ 6 \mapsto 0,\ 7 \mapsto 0,\ 9 \mapsto 2\}$$

repräsentiert.

(c) Schreiben Sie eine Struktur `Set` mit der Signatur `SET` (siehe Skript), die Mengen mit

```
type ''a set = (''a,unit) Map.map
```

implementiert.

Aufgabe 12.2: Funktoren (12 Punkte)

Betrachten Sie folgende abgewandelte Signatur für endliche Funktionen:

```
signature MAP' =
  sig
    type key
    type 'a map
    val empty:   'a map
    val insert:  'a map * key * 'a -> 'a map
    val lookup:  'a map * key -> 'a option
    val domain:  'a map -> key list
  end
```

(a) Schreiben Sie einen Funktor `MakeMap`:

```
(type t val compare: t * t -> order)  -->  MAP' where type key = t
```

der endliche Funktionen mit Suchbäumen implementiert.

(b) Gegeben sei die Signatur:

```
signature SET' =
  sig
    type elem
    type set
    val empty:   set
    val insert:  set * elem -> set
    val member:  set * elem -> bool
  end
```

Schreiben Sie einen Funktor `MakeSet`:

```
(type t val compare: t * t -> order)  -->  SET' where type elem = t
```

der ähnlich wie in Aufgabe 12.1 Mengen mit Hilfe von Maps – in diesem Fall also dem Funktor `MakeMap` – implementiert. Tipp: SML erlaubt Strukturdeklarationen innerhalb von Strukturen.

(c) Gegeben sei eine Signatur für Multimengen:

```
signature MULTISET' =
  sig
    type elem
    type mset
    val empty:   mset
    val insert:  mset * elem -> mset
    val count:   mset * elem -> int
  end
```

Schreiben Sie mit Hilfe von `MakeMap` einen Funktor `MakeMultiset`:

```
(type t val compare: t * t -> order)  -->  MULTISET' where type elem = t
```

Aufgabe 12.3: Große natürliche Zahlen (12 Punkte)

Unter `http://www.ps.uni-sb.de/courses/info1-ws99/uebung/sml/` finden Sie die Signatur und eine unvollständige Implementierung der Struktur `BigNat`, die natürliche Zahlen beliebiger Größe verwirklicht.

(a) Vollziehen Sie die Funktionsweise der Implementierung nach und vervollständigen Sie die Prozeduren `sub` (Subtraktion mit Übertrag) und exp (Exponentiation). Achten Sie bei `sub` auf die Elimination führender Nullen, um die Eindeutigkeit der Darstellung zu erhalten! Realisieren Sie `BigNat` mit Moscow ML als Komponente.

(b) Laden Sie die `BigNat`-Komponente in den Interpreter. Schreiben Sie eine Prozedur zur Berechnung der Fakultät unter Verwendung von `BigNat`. Schreiben Sie weiterhin eine Prozedur `facDigits: int -> int`, die für ein $n \in \mathbb{N}$ die Anzahl der Dezimalstellen von $n!$ berechnet. Wieviele Stellen hat die Fakultät von 333?

Aufgabe 12.4: Große ganze Zahlen (12 Punkte)

Schreiben Sie eine Komponente `BigInt`, die den abstrakten Typ `bigint` mit folgender Signatur implementiert:

```
eqtype bigint

val fromInt :       int    -> bigint
val toInt :         bigint -> int                  (* [Overflow] *)
val toString :      bigint -> string

val < :             bigint * bigint -> bool
val <= :            bigint * bigint -> bool
val isEven :        bigint          -> bool
val isNegative :    bigint          -> bool

val ~ :             bigint          -> bigint
val + :             bigint * bigint -> bigint
val - :             bigint * bigint -> bigint
val * :             bigint * bigint -> bigint
val div :           bigint * bigint -> bigint     (* [Div] *)
val mod :           bigint * bigint -> bigint     (* [Div] *)
val abs :           bigint          -> bigint
val exp :           bigint * bigint -> bigint     (* [Domain] *)
```

Implementieren Sie den Typ `bigint` als

```
type bigint = bool * BigNat.bignat
```

wobei die boolesche Markierung mit `true` anzeigt, dass die Zahl negativ ist. Die Prozedur `toString` soll entsprechend der SML-Syntax ~ als negatives Vorzeichen verwenden. Hinweis: Achten Sie darauf, dass `bigint` ein Typ mit Gleichheit sein soll, also muss insbesondere die Zahl 0 in eindeutiger Weise dargestellt werden.