2. Übungsblatt zu Programmierung WS 99/00

Andreas Rossberg, Prof. Gert Smolka

Abgabe: 5. November 1999 vor der Vorlesung

Allgemeiner Hinweis: Die Lösungen zu den Aufgaben 2.3 und 2.5 schicken Sie bitte zusätzlich per Email an Ihren Übungsgruppenleiter.

Aufgabe 2.1: Endliche partielle Funktion (6 Punkte)

Sei $X = \{0, 1, 2\}$. Betrachten Sie die endliche partielle Funktion

$$d \in X \times X \xrightarrow{fin} X$$
$$d(m,n) = \begin{cases} n - m & \text{falls } n > m \\ m - n & \text{sonst} \end{cases}$$

(a) Geben Sie d in verschiedenen Notationen an:

  (i) als Funktionsdefinition in If-then-else-Notation,
  (ii) in Lambda-Notation,
  (iii) in Mengenschreibweise,
  (iv) mit $\mapsto$,
  (v) in Tabellenform.

(b) Welche der folgenden Eigenschaften hat d: injektiv, surjektiv, bijektiv, total?

Aufgabe 2.2: Induktive Definition (6 Punkte)

Die Menge M ist durch folgende Regeln definiert:

- $3 \in M$, $7 \in M$,
- wenn $m, n \in M$, dann ist $m + n \in M$,
- wenn $k, m, n \in M$, dann ist $k \cdot m \cdot n \in M$.

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

(b) Definieren Sie M durch Inferenzregeln.

(c) Geben Sie eine Ableitung für die Aussage $55 \in M$ und den dazugehörigen Ableitungsbaum an.

Aufgabe 2.3: Listen (8 Punkte)

Schreiben Sie verschiedene SML-Prozeduren wie folgt:

(a) Die Prozedur `last: int list -> int` liefert das letzte Element der Argumentliste. Sie divergiert im Falle einer leeren Liste.

(b) Die Prozedur `lastl: int list -> int list` liefert eine einelementige Liste, die das letzte Element der Argumentliste enthält. Ist die Argumentliste leer, so ist auch die Ergebnisliste leer.

(c) Schreiben Sie analog zu `last` und `lastl` zwei Prozeduren `nth: int list * int -> int` und `nthl: int list * int -> int list`, die das n-te Element einer Liste liefern.

Aufgabe 2.4: Typisierung von Ausdrücken (9 Punkte)

Nehmen Sie an, es seien Bezeichner deklariert mit folgenden Typen:

```
val s: int * bool
val t: real * bool list * real * (int -> int)
val f: int * bool -> int
val g: int -> (real list -> real)
val h: (int -> int) -> (int -> int list)
```

Welche der folgenden SML-Ausdrücke sind dann typkorrekt und welchen Typ besitzen sie? Sagen Sie bei nicht korrekten Ausdrücken in einem Satz, was der Fehler ist.

(a) `( [[4], [3, f(5,true)], 0::[~1,~2], [], [23]], [3.4] )`

(b) `( h(4), false = true )`

(c) `( f, #2(t) @ [], f(s) )`

(d) `[ fn(x:int) => x, fn(y:int) => y + ~2, #4(t) ]`

(e) `fn(x:real, y:int) => x + 2.0*y`

(f) `fn(x:real, y:real) => fn(z:real) => x*y*z`

(g) `f(f(3, 4 <= 5), #1(#2(s), 3.4*0.0))`

(h) `g(let val k = 100 in k end)(nil)`

(i) `h(#4(t))`

Aufgabe 2.5: Typen und Werte (12 Punkte)

Geben Sie jeweils einen SML-Ausdruck (keine Deklaration) an, der den folgenden Typ hat. Verwenden Sie dabei nur Konstanten, Tupel, Listen und Abstraktionen (`fn`) – keine Prozeduranwendung oder Bezeichner aus Standardmodulen! Überprüfen Sie die Typen Ihrer Ausdrücke mit Moscow ML.

(a) `int list list list`

(b) `(real * bool list) list`

(c) `int * int * int -> int * int`

(d) `(real -> real) list`

(e) `bool * (int * int) * (real * (real * (real * real)))`

(f) `(int -> int list) * (int -> int) list`

(g) `int -> (real -> (int * real) list)`

(h) `(int -> int) -> int list`