Ludwig-Maximilians-Universität München WS 2005/2006
Institut für Informatik Blatt 7
Prof. Dr. F. Kröger, Dr. M. Hölzl, F. Hacklinger

Übungen zu Informatik I

Aufgabe 7-1 Typisierung von Funktionsanwendungen (keine Abgabe)

Funktionsanwendungen können mit der Regel

$$\text{(R5)} \qquad \Gamma \rhd t : typ_1 \to typ_2, \Gamma \rhd s : typ_1 \vdash \Gamma \rhd ts : typ_2$$

typisiert werden.

a) Geben Sie eine Herleitung folgender Typaussage an:

$$\emptyset \ \rhd \textbf{fn}\ x \Rightarrow \textbf{fn}\ y \Rightarrow x * y : \textbf{int} \to \textbf{int} \to \textbf{int}$$

b) Geben Sie eine Herleitung der Typaussage

$$\emptyset \ \rhd (\textbf{fn}\ x \Rightarrow \textbf{fn}\ y \Rightarrow x * y)\ 5 : \textbf{int} \to \textbf{int}$$

unter Verwendung der Typisierungsregel (R5) an.

Aufgabe 7-2 Ausnahmebehandlung (keine Abgabe)

In SML ist es möglich, auf Ausnahmen, die bei der Auswertung einer Funktion auftreten, zu reagieren: Ist t ein SML-Term, so hat der Term t **handle** $e \Rightarrow s$ den Wert von s, falls bei der Auswertung von t die Ausnahme e auftritt, ansonsten den Wert von t.

Wenn bei der Auswertung arithmetischer Ausdrücke die auftretenden Zahlen zu groß für das SML-System werden, wird die Auswertung mit der Standardausnahme *overflow* abgebrochen.

a) Die Berechnung der Funktion *fak*(n), definiert durch

```
fun fak n = if n=0 then 1 else n*fak(n-1)
```

wird mit der *overflow*-Ausnahme abgebrochen, wenn die Funktionswerte zu groß werden. Schreiben Sie ein SML-Programm *safe_fak*, das in diesen Fällen den Wert -1 und in allen anderen Fällen den gleichen Wert wie *fak* zurückgibt.

b) Für $n, m \in \mathbb{N}$ lässt sich die Exponentiation n^m nach folgendem Verfahren berechnen:

$$n^m = \begin{cases} 1 & \text{falls } m = 0 \\ (n^{m \text{ div } 2}) \cdot (n^{m \text{ div } 2}) & \text{falls } m \text{ gerade und } m > 0 \\ n \cdot n^{m-1} & \text{falls } m \text{ ungerade} \end{cases}$$

(Dabei wird 0^0 als 1 berechnet.) Schreiben Sie ein SML-Programm *safe_exp*, das die Exponentiation nach diesem Verfahren berechnet und -1 zurückgibt, falls die Berechnung mit einer *overflow*-Ausnahme abgebrochen wird.

Aufgabe 7-3 **Komplexe Zahlen** (keine Abgabe)

Komplexe Zahlen können durch ein Paar ganzer Zahlen dargestellt werden, wobei eine Zahl den *Realteil* und die andere den *Imaginärteil* darstellen.

a) Geben Sie eine SML-Typdeklaration *complex* an, die komplexe Zahlen durch Paare von ganzen Zahlen (Real- und Imaginärteil) darstellen.

b) Schreiben Sie SML-Programme *addcomplex* und *multcomplex*, die jeweils zwei Werte vom Typ *complex* als Parameter haben und als die Summe, beziehungsweise das Produkt der durch die Parameter dargestellten komplexen Zahlen als Ergebnis haben.

Aufgabe 7-4 **Typisierung mit lokalen Konstanten** (4 Punkte)

Die Typisierungsregel für Terme mit lokalen Konstanten lautet

$$\text{(R6)} \qquad \Gamma \rhd (\textbf{fn}\ x \Rightarrow s)(t) : \mathit{typ} \vdash \Gamma \rhd \textbf{let val}\ x = t\ \textbf{in}\ s\ \textbf{end}\ : \mathit{typ}$$

Geben Sie unter Verwendung dieser Typisierungsregel (und „offensichtlichen“ Typaxiomen) eine Herleitung der Typaussage

$$\{x : \textbf{int}\} \rhd \textbf{let val}\ x2 = x * 2\ \textbf{in}\ x2 + x + 1\ \textbf{end} : \textbf{int}$$

an.

Aufgabe 7-5 **Mustervergleich** (4 Punkte)

Schreiben Sie SML-Programme *qzt_pm*, *teilersumme_pm* und *hatteiler_pm*, die Mustervergleich verwenden und folgendes leisten:

a) Das Programm *qzt_pm* bestimmt für zwei natürliche Zahlen n und k mit $k \leq n$, ob n das Quadrat einer natürlichen Zahl i mit $0 \leq i \leq k$ ist oder nicht (vgl. Aufgabe 2-4 bzw. Aufgabe 5-4).

b) Das Programm *teilersumme_pm* bestimmt für zwei natürliche Zahlen n und k die Summe aller Teiler i von n mit $1 \leq i \leq k$ (vgl. Aufgabe 2-5).

c) Das Programm *hatteiler_pm* bestimmt für zwei natürliche Zahlen n und k mit $k \leq n$, ob n einen Teiler i mit $2 \leq i \leq k$ besitzt.

Aufgabe 7-6 **Brüche** (4 Punkte)

Ein Bruch besteht aus einem Zähler und einem Nenner, die beide ganze Zahlen sind.

a) Geben Sie eine SML-Typdeklaration *fraction* an, die Brüche durch Paare (Zähler und Nenner) von ganzen Zahlen darstellt.

b) Schreiben Sie ein SML-Programm *multfrac*, das zwei Werte vom Typ *fraction* als Parameter und als Ergebnis das Produkt der durch die Parameter dargestellten Brüche hat.

c) Schreiben Sie ein SML-Programm *addfrac*, das zwei Werte vom Typ *fraction* als Parameter und als Ergebnis die Summe der durch die Parameter dargestellten Brüche hat.

d) Schreiben Sie ein SML-Programm *divfrac*, das zwei Werte vom Typ *fraction* als Parameter und als Ergebnis die Division der durch die Parameter dargestellten Brüche hat.

Hinweis: Zur Vereinfachung können bei der Lösung der Aufgabe davon ausgehen, dass sowohl Zähler als auch Nenner der Parameter jeweils nicht 0 sind.

Abgabe: Montag, 19.12.2005, 8:00 Uhr.