Fachrichtung 6.2 — Informatik
Universität des Saarlandes
Tutorenteam der Vorlesung Programmierung 1

Programmierung 1 (Wintersemester 2012/13)

Lösungsblatt 1

(Kapitel 1)

Hinweis: Dieses Übungsblatt enthält von den Tutoren für die Übungsgruppe erstellte Aufgaben. *Die Aufgaben und die damit abgedeckten Themenbereiche sind für die Klausur weder relevant, noch irrelevant.*

1 Rekursive Prozeduren verstehen

Aufgabe 1.1
Welche mathematischen Funktionen berechnen die folgenden endrekursiven Prozeduren?

Beispiel: Die Prozedur

```
fun f (x:int) :int = x * x
```

berechnet die Funktion $f\colon \mathbb{N} \to \mathbb{N}, x \mapsto x^2$ und

```
fun f (x:int,y:int) :int = x*y
```

berechnet die Funktion $f\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}, (x, y) \mapsto x \cdot y$

(a)
```
fun mul(a:int, n:int, x:int) : int =
  if n=0
    then a
    else mul(a+x,n-1,x)
```

(b)
```
fun quer(a:int, x:int) : int =
  if x=0
    then a
    else quer(a + x mod 10,x div 10)
```

(c)
```
fun rev(m:int, n:int) : int =
  if n=0
    then m
    else rev(m*10 + n mod 10,n div 10)
```

Lösung 1.1:

(a) Die Prozedur berechnet die Funktion

$$\begin{aligned} f : \mathbb{Z} \times \mathbb{N} \times \mathbb{Z} &\to \mathbb{Z}, \\ (a, n, x) &\mapsto a + x \cdot n \end{aligned}$$

(b) Die Prozedur berechnet die Quersumme einer Zahl, also die Funktion

$$\begin{aligned} f : \mathbb{Z} \times \mathbb{N} &\to \mathbb{Z} \\ (a, x) &\mapsto \sum_{i=0}^{s(x)-1} \left\lfloor \frac{x}{10^i} \right\rfloor \bmod 10 \end{aligned}$$

wobei $s(x)$ die Stelligkeit der Zahl x darstellt.

(c) Die Prozedur reversiert eine Zahl, also berechnet die Funktion

$$\begin{aligned} f : \mathbb{Z} \times \mathbb{N} &\rightarrow \mathbb{Z} \\ (a, x) &\mapsto \sum_{i=0}^{s(x)-1} \left(\left\lfloor \frac{x}{10^i} \right\rfloor \bmod 10 \right) \cdot 10^{s(x)-i-1} \end{aligned}$$

wobei $s(x)$ die Stelligkeit der Zahl x darstellt.

Aufgabe 1.2
Stellen Sie das Ausführungsprotokoll für die folgenden Aufrufe der oben genannten Prozeduren auf:

(a) *rev*(*538*, *10*)

(b) *mul*(*17*, *3*, *10*)

(c) *quer*(*9273*, *0*)

Hinweis: Überlegen Sie sich gegebenfalls neu, was die obigen Prozeduren berechnen.

Lösung 1.2:

(a) Ausführungsprotokoll:

$$\begin{aligned} \underline{\text{rev(538,10)}} &= \text{if } \underline{10=0} \text{ then 538 else rev(538*10 + 10 mod 10 , 10 div 10)} \\ &= \underline{\text{if false then 538 else rev(538*10 + 10 mod 10 , 10 div 10)}} \\ &= \text{rev}(\underline{538*10 + 10 \bmod 10} \text{ , 10 div 10)} \\ &= \text{rev(5380 , } \underline{10 \text{ div } 10}\text{)} \\ &= \underline{\text{rev(5380 , 1)}} \\ &= \text{if } \underline{1=0} \text{ then 5380 else rev(5380*10 + 1 mod 10 , 1 div 10)} \\ &= \underline{\text{if false then 5380 else rev(5380*10 + 1 mod 10 , 1 div 10)}} \\ &= \text{rev}(\underline{5380*10 + 1 \bmod 10} \text{ , 1 div 10)} \\ &= \text{rev(53801 , } \underline{1 \text{ div } 10}\text{)} \\ &= \underline{\text{rev(53801 , 0)}} \\ &= \text{if } \underline{0=0} \text{ then 53801 else rev(53801*10 + 0 mod 10 , 0 div 10)} \\ &= \underline{\text{if true then 53801 else rev(53801*10 + 0 mod 10 , 0 div 10)}} \\ &= 53801 \end{aligned}$$

Verkürztes Ausführungsprotokoll:

$$\begin{aligned} \text{rev(538,10)} &= \text{rev(5380 , 1)} \\ &= \text{rev(53801 , 0)} \\ &= 53801 \end{aligned}$$

(b) Ausführungsprotokoll:

$$
\begin{aligned}
\underline{\text{mul(17,3,10)}} &= \text{if } \underline{3=0} \text{ then 17 else mul(17+10 , 3-1 , 10)} \\
&= \underline{\text{if false then 17 else mul(17+10 , 3-1 , 10)}} \\
&= \text{mul(}\underline{17+10}\text{ , 3-1 , 10)} \\
&= \text{mul(27 , }\underline{3\text{-}1}\text{ , 10)} \\
&= \underline{\text{mul(27 , 2 , 10)}} \\
&= \text{if } \underline{2=0} \text{ then 27 else mul(27+10 , 2-1 , 10)} \\
&= \underline{\text{if false then 27 else mul(27+10 , 2-1 , 10)}} \\
&= \text{mul(}\underline{27+10}\text{ , 2-1 , 10)} \\
&= \text{mul(37 , }\underline{2\text{-}1}\text{ , 10)} \\
&= \underline{\text{mul(37 , 1 , 10)}} \\
&= \text{if } \underline{1=0} \text{ then 37 else mul(37+10 , 1-1 , 10)} \\
&= \underline{\text{if false then 37 else mul(37+10 , 1-1 , 10)}} \\
&= \text{mul(}\underline{37+10}\text{ , 1-1 , 10)} \\
&= \text{mul(47 , }\underline{1\text{-}1}\text{ , 10)} \\
&= \underline{\text{mul(47 , 0 , 10)}} \\
&= \text{if } \underline{0=0} \text{ then 47 else mul(47+10 , 0-1 , 10)} \\
&= \underline{\text{if true then 47 else mul(47+10 , 0-1 , 10)}} \\
&= 47
\end{aligned}
$$

Verkürztes Ausführungsprotokoll:

$$
\begin{aligned}
\text{mul(17,3,10)} &= \text{mul(27,2,10)} \\
&= \text{mul(37,1,10)} \\
&= \text{mul(47,0,10)} \\
&= 47
\end{aligned}
$$

(c) Ausführungsprotokoll:

$$
\begin{aligned}
\underline{\text{quer(9273,0)}} &= \text{if } \underline{0=0} \text{ then 9273 else quer(9273 + 0 mod 10 , 0 mod 10)} \\
&= \underline{\text{if true then 9273 else quer(9273 + 0 mod 10 , 0 mod 10)}} \\
&= 9273
\end{aligned}
$$

Verkürztes Ausführungsprotokoll:

$$
\underline{\text{quer(9273,0)}} = 9273
$$

Aufgabe 1.3
Wie erkennt man endrekursive Prozeduren am Ausführungsprotokoll? Finden Sie mindestens zwei Charakterisierungen!

Lösung 1.3:
Hier einige Charakteristika endrekursiver Prozeduren:

- der rekursive Aufruf ist im Rumpf der Prozedur nicht von anderen Operationen umgeben
- in den meisten Fällen wird ein Akku verwendet

2 Sprache der Informatik

Aufgabe 1.4
Betrachten Sie das folgende Programm:

```
fun f (x:int) :int =
  if x=42
  then f(x-1) + 4
  else 10
fun g (x:real) :real =
  x * x + 2.0 * x + 1.0
val s = (f(1),g(1.0))
val t = (g(3.14), f(42))
fun g (x:real) : real =
  x * x * x + 3 * x * x + 3 * x + 1
```

Identifizieren Sie Schlüsselwörter, Konstanten, Bezeichner und Operatoren. Was tut das Programm? An was wird der Bezeichner *s* gebunden?

Aufgabe 1.5
Betrachten Sie die folgende Prozedurdeklaration:

```
fun f (x:int) :bool = f(x)
```

Identifizieren Sie den Prozedurtyp, den Argumenttyp und den Ergebnistyp der Prozedur. Was ist der Rumpf der Prozedur?

Hinweis: Ist die Prozedur wohlgetypt? Begründen Sie Ihre Antwort!

Aufgabe 1.6
Stellen Sie sich vor, Sie könnten eine Woche in die Vergangenheit reisen und treffen sich selbst. Sie kennen zu diesem Zeitpunkt das Konzept von Divergenz noch nicht. Wie würden Sie sich Divergenz erklären? Finden Sie ein anschauliches Beispiel!

Lösung 1.6:
Ein anschauliches Beispiel in SML für Divergenz ist zum Beispiel die Prozedur:

```
fun f (x:int) :int = f(x)
```

Aber auch das Verständnis der Rekursion kann mit Divergenz beschreiben: “Rekursion hat man erst verstanden, wenn man Rekursion verstanden hat.”

3 Kleine Programme schreiben

Aufgabe 1.7
Schreiben Sie ein kleines Programm, das den Bezeichner *s* an den Wert *5* bindet. Verwenden Sie dabei den Wert *5 nicht* explizit, d.h. die Konstante *5* darf in Ihrem Programm nicht vorkommen.

Lösung 1.7:

- Erste Möglichkeit:

```
fun f (x:int) :int = x + 2
val s = f(3)
```

- Zweite Möglichkeit:

```
val s = 3 + 2
```

- ...

Aufgabe 1.8
Schreiben Sie ein Programm, in dem jeder Typ, den Sie bisher kennengelernt haben sowohl als Argumenttyp einer Prozedur als auch als Ergebnistyp einer Prozedur mindestens einmal vorkommt.

Lösung 1.8:
Ein Beispiel:

```
fun f (x:unit) :int = 2
fun g (x:real) :bool = if f(()) < 4 then true else false
fun h (x:int) :unit = if g(3.0) then () else ()
fun i (x:bool) :real = if g(4.1) then 1.4 else 3.14
```

Aufgabe 1.9
Schreiben Sie ein Programm mit zwei Prozedurdeklarationen und fünf Prozeduranwendungen.

Lösung 1.9:

```
fun f(n:int) :int = if n = 0 then 1 else n * f(n-1)
fun g(n:int):int = if f(n)>f(n-1) then n + g(n-1) else n * g(n-1)
```

4 Rekursive Prozeduren schreiben

Aufgabe 1.10
Schreiben Sie eine rekursive Prozedur *fakultaet*: $int \to int$, die für eine natürliche Zahl n die Fakultät berechnet und für alle anderen Aufrufe divergiert. Geben Sie zunächst die Rekursionsgleichungen an.

Lösung 1.10:

```
fun fakultaet (n:int) :int = if n= 0 then 1 else n * (fakultaet(n-1))
```

Aufgabe 1.11
Geben Sie die Rekursionfolge und das Ausführungsprotokoll Ihrer Prozedur aus 1.10 für *fakultaet*(*3*) an.

Lösung 1.11:
Rekursionfolge: $fakultaet(3) \to fakultaet(2) \to fakultaet(1) \to fakultaet(0)$

Aufgabe 1.12
Schreiben Sie eine nicht-rekursive Prozedur *fakultaet3*: $int \to int * int * int$ die für eine natürliche Zahl n die einen Tupel zurückgibt, der in der ersten Komponente die Fakultät von $(n - 1)$, in der zweiten Komponente die Fakutlät von n und in der dritten Komponente die Fakultät von $(n + 1)$. Ist mindestens eine der Zahlen negativ, so ist es egal, was Ihre Prozedur zurückgibt.

Lösung 1.12:
fun fakultaet3 (n:int) :int*int*int = (fakultaet(n-1), fakultaet(n), fakultaet(n+1))

Aufgabe 1.13
Erklären Sie anhand der endrekursiven Variante der Prozedur Fakultät aus Aufgabe 1.10 das Konzept des Akkus.

Aufgabe 1.14
Sie haben einige Beispiele für rekursive Prozeduren bereits im Buch, auf dem Übungsblatt und in der Vorlesung gesehen. Schreiben Sie für mindestens drei Beispiele die zugehörigen Rekursionsgleichungen auf. Wandeln Sie diese Rekursionsgleichungen dann in endrekursive um.

Überlegen Sie sich dann ein allgemeines Vorgehen wie Sie aus Rekursiongleichungen SML-Prozeduren schreiben.

5 Knobelaufgaben

Aufgabe 1.15 (*Zum Üben von Syntax*)
Schreiben Sie in SML eine Prozedur *taschenrechner*, die natürliche Zahlen addieren, subtrahieren, dividieren und multiplizieren kann.

Hinweis: Überlegen Sie sich, wie man entscheiden kann, welche Operation ausgeführt werden soll.

Lösung 1.15:

```
fun taschenrechner (a:int, n:int, m:int) :int =
if a = 1
then n + m
else if a = 2
then n -m
else if a = 3
then n * m
else n / m
```

Aufgabe 1.16 (*Zu Rekursionsgleichungen*)
Überlegen Sie sich nicht-endrekursive Rekursionsgleichungen für die Berechnung von $\lfloor\sqrt{n}\rfloor$ für eine natürliche Zahl n. Ist die ohne Probleme möglich? Was fällt Ihnen auf?

Lösung 1.16:

```
fun wurzel (a, h) = if h*h > a then  1 else 1 + wurzel (a, h+1)
```

Aufgabe 1.17 (*Verzwickte Prozedur*)
Betrachten Sie den Anfang der folgenden Folge $(a_n)_{n\in\mathbb{N}}$ von Zahlen, die man auch *Tower of Twos* nennt: $a_0 = 2, a_1 = 2^2, a_2 = 2^{(2^2)}, a_3 = 2^{(2^{(2^2)})}, \ldots$
Schreiben Sie eine SML-Prozedur, die für eine natürliche Zahl n die n-te Zahl dieser Form a_n berechnet. Für welches $n \in \mathbb{N}$ können Sie diese Zahl noch berechnen? Mit welcher Exception wird die Berechnung abbrechen?

Lösung 1.17:

```
fun potenz(a:int, x:int,n:int) :int = if n<1 then a else potenz(a*x, x, n-1)
fun towerOfTwos (n:int) :int = if n = 0 then 1 else potenz(1, 2, towerOfTwos(n-1))
```