Christian Ikenmeyer, Peter Pfahler, Florian Schoppmann, Adrian Slowik, Patrick van Lück, Björn Zeiger

Vorlesung Grundlagen der Programmiersprachen Sommer 2004 - Lösung zu Blatt 7

Lösung zu Aufgabe 19

```
fun eins(x:int, y:int) = x < y;
```

Signatur: `int * int -> bool`

Die Funktion berechnet, ob die erste der beiden Zahlen kleiner als die zweite ist.

```
fun zwei z = if z=1 then 1
                    else z + (zwei(z-1));
```

Signatur: `int -> int`

Die Funktion berechnet bei Eingabe z die Summe der Zahlen von 1 bis z.

```
fun drei (e,(x,y)) = if e then (x,y) else (y,x);
```

Signatur: `bool * ('a * 'a) -> 'a * 'a`

Die Funktion liefert bei Eingabe (b, t) das Tupel t entweder unverändert oder getauscht zurück, je nachdem ob der Wahrheitswert b `true` oder `false` ist.

```
fun vier a = drei (eins a, a);
```

Signatur: `int * int -> int * int`

Die Funktion erhält ein Paar ganzer Zahlen und gibt das Paar zurück, wobei ggf. die Elemente vertauscht werden, damit sie in aufsteigender Reihenfolge sortiert sind.

```
fun fuenf (x,y) = if x=0 then y
                         else fuenf (x-1, y+1);
```

Signatur: `int * int -> int`

Die Funktion liefert die Summe der beiden übergebenen Zahlen. Das funktioniert allerdings nur, wenn die erste Zahl nicht negativ ist, ansonsten terminiert die Funktion nicht.

Lösung zu Aufgabe 20

a) Schreiben Sie unter Verwendung von `MkList` und `Sum` die Funktion `SumBis n`, die zu einer positiven ganzen Zahl `n` die Summe der Zahlen von `1` bis `n` berechnet.

```
fun SumBis n = Sum (MkList n);
```

b) Schreiben Sie eine Funktion `NonMember(l, m)`, die berechnet, ob `m` nicht Element von `l` ist.

```
fun NonMember(nil, m) = true
|   NonMember(h::t, m) = if h = m
                    then false
                    else NonMember(t, m);
```

c) Schreiben Sie eine Funktion `UniqueElems(l)`, die mit Hilfe der Funktion `Member` berechnet, ob jedes Element von `l` nur einmal in `l` enthalten ist.

```
fun UniqueElements(nil) = true
|   UniqueElements(h::t) = if Member(t, h)
                          then false
                          else UniqueElements(t);
```

d) Schreiben Sie eine Funktion `Filter(l, x)`, die aus einer Liste `l` eine neue Liste berechnet, die aus allen Elementen von `l` besteht außer denen, die den Wert `x` haben.

```
fun Filter(nil, x) = nil
|   Filter(h::t, x) = if h = x
                      then Filter(t, x)
                      else h::Filter(t, x);
```

e) Welche Signatur ermittelt das Moscow ML-System für die Funktion

```
fun guess ( a , nil ) = [a]
  | guess ( a, h::t ) = if ( a > h ) then a::h::t
                                     else h::guess(a,t);
```

Die Signatur ist

```
fn : int * int list -> int list
```

Welchen Wert liefert der folgende Aufruf?

```
guess(3, MkList 7);
```

Der Wert ist

```
[7, 6, 5, 4, 3, 3, 2, 1] : int list
```

f) Wie oft wird die `::`-Operation bei der Abarbeitung des Aufrufes

```
Append(MkList 7, MkList 10);
```

ausgeführt?

```
Append(MkList 7, MkList 10) ==
Append([7,6,5,4,3,2,1], [10,9,8,7,6,5,4,3,2,1]) ==
7::Append([6,5,4,3,2,1], [10,9,8,7,6,5,4,3,2,1]) ==
7::6::Append([5,4,3,2,1], [10,9,8,7,6,5,4,3,2,1]) ==
7::6::5::Append([4,3,2,1], [10,9,8,7,6,5,4,3,2,1]) ==
7::6::5::4::Append([3,2,1], [10,9,8,7,6,5,4,3,2,1]) ==
7::6::5::4::3::Append([2,1], [10,9,8,7,6,5,4,3,2,1]) ==
7::6::5::4::3::2::Append([1], [10,9,8,7,6,5,4,3,2,1]) ==
7::6::5::4::3::2::1::Append([], [10,9,8,7,6,5,4,3,2,1]) ==
7::6::5::4::3::2::1::[10,9,8,7,6,5,4,3,2,1] ==
```

Die `::`-Operation wird 7 mal ausgeführt. Dies entspricht der Länge des ersten Arguments von `append`.

Lösung zu Aufgabe 21

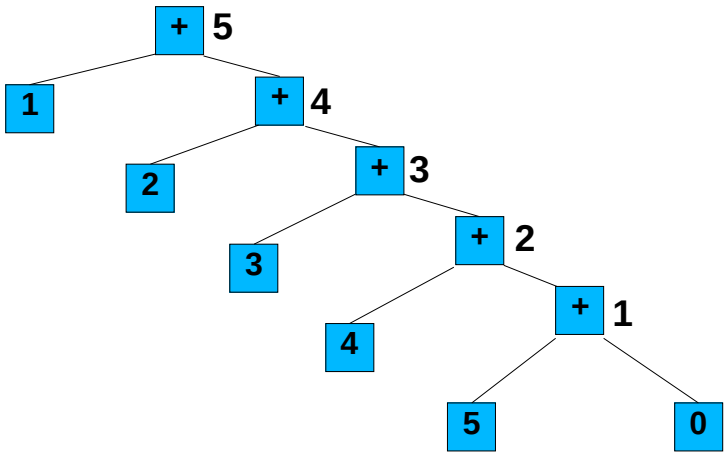
Die beiden Funktionen:

```
fun Sum nil = 0
|   Sum(h::t) = h + Sum t;

fun ASum(nil, a: int) = a
|   ASum(h::t, a) = ASum(t, a+h);
```

Auswertung von `Sum [1,2,3,4,5]`:

```
Sum [1,2,3,4,5] =
1 + (Sum [2,3,4,5]) =
1 + (2 + (Sum [3,4,5])) =
1 + (2 + (3 + (Sum [4,5]))) =
1 + (2 + (3 + (4 + (Sum [5])))) =
1 + (2 + (3 + (4 + (5 + (Sum []))))) =
1 + (2 + (3 + (4 + (5 + (0))))) =
1 + (2 + (3 + (4 + (5)))) =
1 + (2 + (3 + (9))) =
1 + (2 + (12)) =
1 + (14) =
15
```



Auswertung von `ASum([1,2,3,4,5], 0)`:

```
ASum([1,2,3,4,5], 0) =
ASum([2,3,4,5], 0 + 1) =
ASum([2,3,4,5], 1) =
ASum([3,4,5], 1 + 2) =
ASum([3,4,5], 3) =
ASum([4,5], 3 + 3) =
ASum([4,5], 6) =
ASum([5], 6 + 4) =
ASum([5], 10) =
ASum([], 10 + 5) =
15
```