TECHNISCHE UNIVERSITÄT MÜNCHEN
FAKULTÄT FÜR INFORMATIK

WS 11/12

Einführung in die Informatik II — **Übungsblatt 2**

Univ.-Prof. Dr. Andrey Rybalchenko, M.Sc. Ruslán Ledesma Garza — 8.11.2011

Dieses Blatt behandelt Kapitel 1.1 - 2.4 (exclusive 1.10) aus dem Buch zur Vorlesung. Lesen Sie diese Kapitel!

Aufgabe 1.23 (Reversion) Unter der Reversion `rev n` einer natürlichen Zahl n wollen wir die natürliche Zahl verstehen, die man durch Spiegeln der Dezimaldarstellung von n erhält. Beispielsweise soll `rev 1234 = 4321`, `rev 76 = 67` und `rev 1200 = 21` gelten.

a) Schreiben Sie zunächst eine endrekursive Prozedur `rev' : int * int → int`, die zu zwei natürlichen Zahlen m und n die Zahl liefert, die sich ergibt, wenn man die reversierte Dezimaldarstellung von n rechts an die Dezimaldarstellung von m anfügt. Beispielsweise soll `rev'(65,73) = 6537` und `rev'(0,12300) = 321` gelten. Die Arbeitsweise von `rev'` ergibt sich aus dem verkürzten Ausführungsprotokoll `rev'(65,73) = rev'(653,7) = rev'(6537,0) = 6537`.

b) Schreiben Sie mithilfe der Prozedur `rev'` eine Prozedur `rev`, die natürliche Zahlen reversiert.

c) Machen Sie sich klar, dass die entscheidende Idee bei der Konstruktion des Reversionsalgorithmus die Einführung einer Hilfsfunktion mit einem Akku ist. Überzeugen Sie sich davon, dass Sie `rev` nicht ohne Weiteres durch Rekursionsgleichungen bestimmen können.

Lösungsvorschlag 1.23:

a)
```
fun rev' (x:int, y:int) =
  if y=0 then x else rev' (x*10+(y mod 10),y div 10)
```

b)
```
fun rev (x:int) =
  rev' (0,x)
```

c)

$$\text{rev } x = 10^{\lfloor \log_{10} x \rfloor} * (x \bmod 10) + \text{rev } (x \text{ div } 10) \quad (x \neq 0)$$

Wie berechnen wir $10^{\lfloor \log_{10} x \rfloor}$?

```
fun rev (x:int) =
  if x=0 then 0 else Real.ceil (Math.pow (10.0,Real.fromInt (Real.floor
(Math.log10 (Real.fromInt x)))))*(x mod 10) + rev (x div 10)
```

PROBLEM? 

Aufgabe 1.24 Schreiben Sie eine Prozedur `int → int`, die für negative Argumente divergiert und für nicht-negative Argumente x das Ergebnis x liefert.

Lösungsvorschlag 1.24:

```
fun f (x:int) = if x<0 then f(x) else x
```

Aufgabe 1.27 Schreiben Sie eine Prozedur `p : real * real` → `real`, die die Funktion $f(x,y) = (x-3)(y+5)^2$ mit Gleitkommazahlen berechnet. Verwenden Sie eine lokale Deklaration, damit die Addition $y+5$ nur einmal berechnet werden muss.

Lösungsvorschlag 1.27:

```
fun p (x:real, y:real) =
  let val yplus5 = y+5.0 in
    (x-3.0)*yplus5*yplus5
  end
```

Aufgabe 1.28 Schreiben Sie eine rekursive Prozedur `power : real * int` → `real`, die zu einer reellen Zahl x und einer natürlichen Zahl n die Potenz x^n mittels Gleitkommaoperationen berechnet. Welche Zahl liefert `power(3.0,100)`? Handelt es sich dabei wirklich um die Zahl 3^{100}?

Lösungsvorschlag 1.28:

```
fun power (x:real, n:int) =
  if n=0 then 1.0 else x*power(x,n-1)
```

$$3^{100} = 515377520732011331036461129765621272702107522001 \neq 5.153775207E47 = power(3.0, 100)$$

Aufgabe 1.30 Deklarieren Sie Prozeduren des Typs `real` → `real`, die die folgenden Funktionen mit Gleitkommaoperationen berechnen:

a) $g(x) = 2x + 1,4e$

b) $h(x) = \sin x + \cos(2\pi x)$

Lösungsvorschlag 1.30:

```
fun g (x:real) = 2.0*x+1.4*Math.e

fun h (x:real) = Math.sin x + Math.cos (2.0*Math.pi*x)
```

Aufgabe 2.1 Geben Sie die Baumdarstellungen der folgenden durch Zeichendarstellun- gen beschriebenen Phrasen an.

a) `int * int` → `bool`

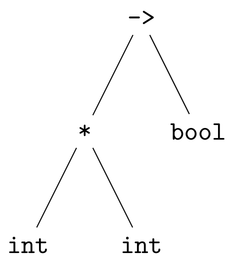
b) **if** `x<3` **then** `3` **else** `p 3`

c) **let val** `x = 2+y` **in** `x-y` **end**

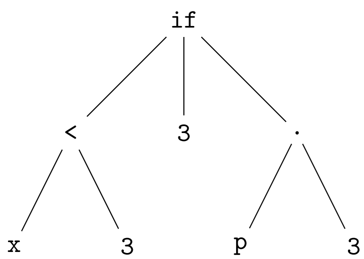
d) **fun** `p(x:int,n:int):int=`**if** `n>0` **then** `x*p(x,n-1)`**else** `1`

Lösungsvorschlag 2.1:

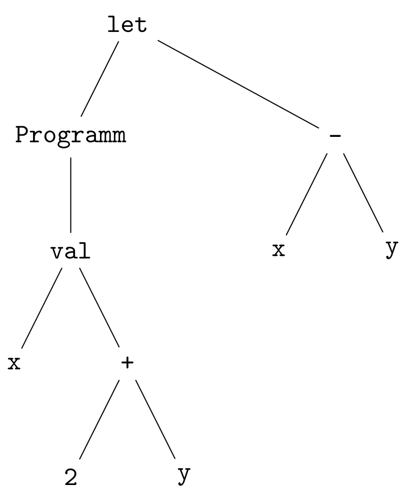
a) `int * int` → `bool`



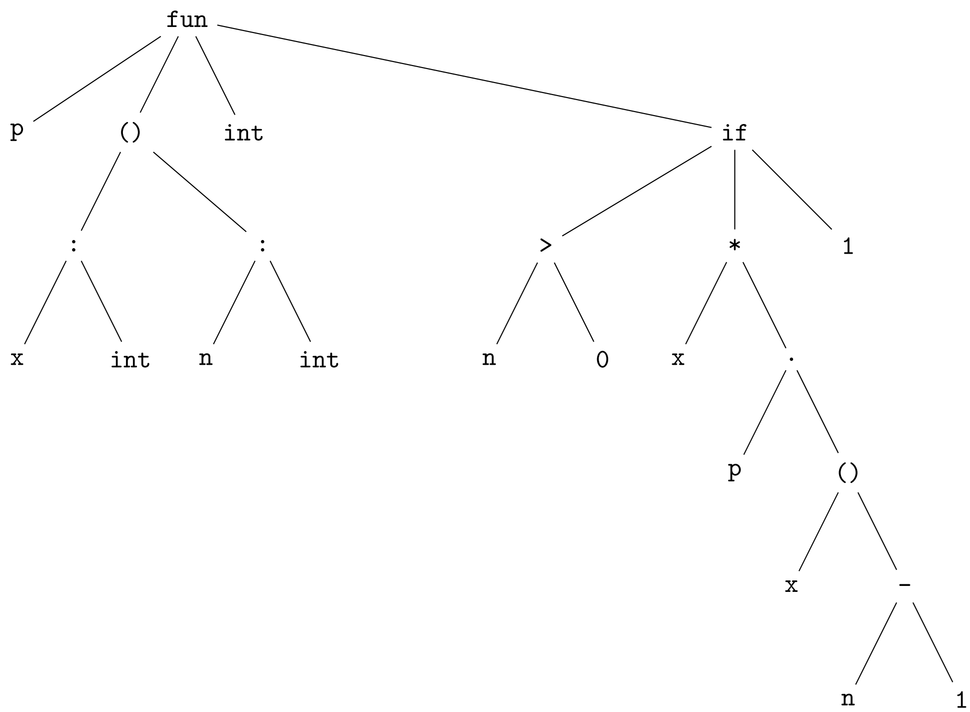
b) **if** `x<3` **then** `3` **else** `p 3`



c) **let val** `x = 2+y` **in** `x-y` **end**



d) **fun** `p(x:int,n:int):int`**=if** `n>0` **then** `x*p(x,n-1)`**else** `1`



Aufgabe 2.2 Geben Sie die Baumdarstellungen der folgenden durch Zeichendarstellungen beschriebenen Phrasen an.

a) `x+3*f x-4`

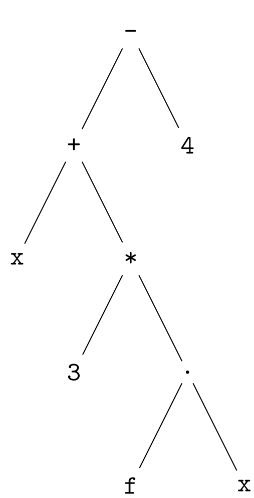
b) `1+2+3+4`

c) `1+2*x-y*3+4`

d) `int*(int*int)*int→int*int`

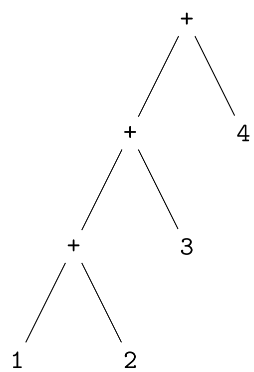
Lösungsvorschlag 2.2:

a) `(x+(3*(f x)))-4`

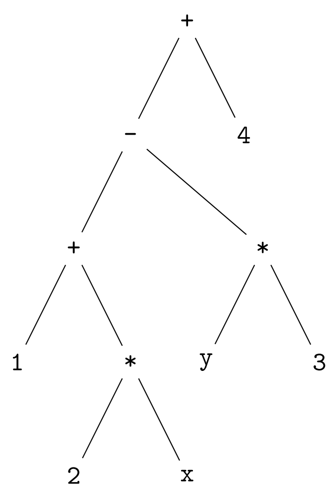


fun before * before +, all left associative

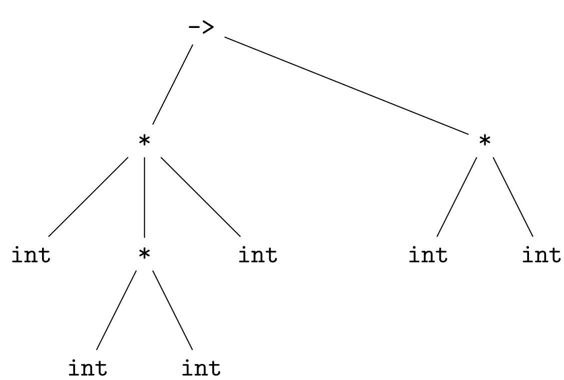
b) `((1+2)+3)+4`



c) `((1+(2*x))-(y*3))+4`



d) `int*(int*int)*int`$\rightarrow$`int*int`



Aufgabe sec2.4 Geben Sie für die folgenden Phrasen alle freien Bezeichner an.

a)
```
fun f (x:int) = x + y;
```

b)
```
fun sum (x:int) = if x <= 0 then 0 else x + sum (x - 1)
```

c)
```
fun pos (x:int) = x > 0;
fun abs (y:int) = if pos y then y else y * ~1;
```

d)
```
let val x = x in if g x then y else x
```

Lösungsvorschlag sec2.4:

a) `y`

b) $\emptyset$

c) $\emptyset$

d) `x, g, y`

Aufgabe sec2.6 In dieser Aufgabe müssen Sie eine Funktionsdefinition in Regelform wiedergeben. Zur Veranschaulichung dazu ein kurzes Beispiel: Gegeben sei ein Graph $G = (V, E)$ mit dem Startknoten $v_0 \in V$. Die Menge aller erreichbaren Knoten $R \subseteq V$ wird über folgende Regeln definiert:

$$\frac{}{v_0 \in R} \qquad \frac{v \in R \quad (v, w) \in E}{w \in R}$$

Nun die eigentliche Aufgabe: Die folgende Funktion $F : \mathbb{N} \to \mathbb{N}$ berechnet die n-te Fibonaccizahl:

$$F(n) = \begin{cases} 1 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ F(n-1) + F(n-2) & \text{otherwise} \end{cases}$$

Schreiben Sie diese Funktion als Regeln.

Lösungsvorschlag sec2.6:

$$\frac{}{(0,1) \in F} \qquad \frac{}{(1,1) \in F} \qquad \frac{(n-1, f') \in F \quad (n-2, f'') \in F}{(n, f' + f'') \in F}$$

Aufgabe **sec2.2.2 Sei $\mathcal{P}(\mathit{Bezeichner})$ die Potenzmenge von *Bezeichner*. Geben Sie eine Regelmenge an, die eine Funktion $B : \mathit{Ausdruck} \to \mathcal{P}(\mathit{Bezeichner})$ definiert, wobei diese Funktion für jeden Ausdruck a die Menge aller Bezeichner, die in a vorkommen, zurückgibt. Als Beispiel zeigen wir die Regeln für die Fälle wenn a ist eine Konstante, ein Bezeichner und ein Konditionalausdruck. Geben Sie bitte die Regeln für weitere Fälle an.

$$\frac{x \in \mathit{Bezeichner}}{(x, \{x\}) \in B} \qquad \frac{k \in \mathit{Konstante}}{(k, \emptyset) \in B} \qquad \frac{(a_1, b_1) \in B \quad (a_2, b_2) \in B \quad (a_3, b_3) \in B}{(\texttt{if } a_1 \texttt{ then } a_2 \texttt{ else } a_3, b_1 \cup b_2 \cup b_3) \in B}$$

Lösungsvorschlag **sec2.2.2: Follow the definitions in Figure 2.3 of page 33.