TECHNISCHE UNIVERSITÄT MÜNCHEN
FAKULTÄT FÜR INFORMATIK

WS 11/12

Einführung in die Informatik II **Übungsblatt 4**

Univ.-Prof. Dr. Andrey Rybalchenko, M.Sc. Ruslán Ledesma Garza 21.11.2011

Dieses Blatt behandelt Kapitel 2.7.1 - 3.3 aus dem Buch zur Vorlesung. Lesen Sie diese Kapitel!

Aufgabe 2.4

Welche Bindungen berechnet das folgende Programm? Geben Sie außerdem den Auswertungsbaum in der Umgebung `[ ]` für dieses Programm an.
Hinweis: Lesen Sie Kapitel 2.7.1 - 2.7.4 und das Skript von der Homepage um diese Aufgabe zu lösen.

```
fun f (x:bool) : int = if x then 1 else 0
val x = 5*7
fun g (z:int) : bool = f(z<x)<x
val x = g 5
```

Lösungsvorschlag 2.4:

a) Part A:

```
f := (fun f x = if x then 1 else 0,bool→int, [])
x := 35
g := ( fun g z = f(z<x)<x
     , int→bool
     , [ x:=35
       , f:=( fun f x = if x then 1 else 0
            , bool→int
            , [])])
x := true
```

b) Part B:

Auswertungsbaum in der Umgebung `[ ]` für das Programm **fun** `f (x:bool) : int = if x then 1 else 0`.

$$\frac{}{\texttt{[]} \mid\!>> \texttt{fun f (x:bool) : int = if x then 1 else 0} : \underbrace{\texttt{[f:= (fun f x = if x then 1 else 0, bool -> int, [])]}}_{V_f}}$$

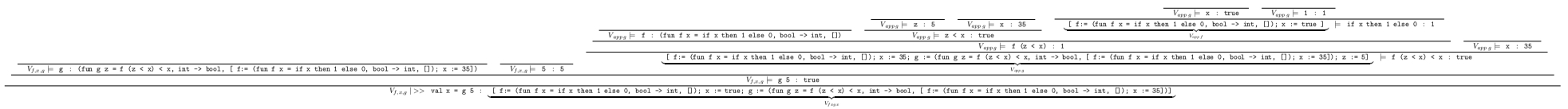
Auswertungsbaum in der Umgebung `[ f:= (fun f x = if x then 1 else 0, bool → int, [])]` für das Programm **val** `x = 5*7`.

$$\frac{\dfrac{\dfrac{}{V_f \models \texttt{5} : \texttt{5}} \quad \dfrac{}{V_f \models \texttt{7} : \texttt{7}}}{V_f \models \texttt{5 * 7} : \texttt{35}}}{V_f \mid\!>> \texttt{val x = 5 * 7} : \underbrace{\texttt{[f:= (fun f x = if x then 1 else 0, bool -> int, []); x := 35]}}_{V_{f,x}}}$$

Auswertungsbaum in der Umgebung `[ f:= (fun f x = if x then 1 else 0, bool → int, []); x := 35]` für das Programm **fun** `g z = f (z < x) < x`.

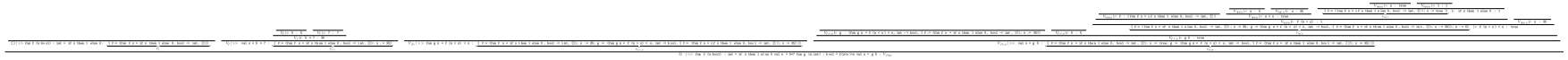
$V_{f,x}$ | >> `fun g z = f (z < x) < x :` $\underbrace{\texttt{[f:= (fun f x = if x then 1 else 0, bool -> int, []); x := 35; g := (fun g z = f (z < x) < x, int -> bool, [f:= (fun f x = if x then 1 else 0, bool -> int, []); x := 35])]}_{V_{f,x,g}}$

Auswertungsbaum in der Umgebung `[ f:= (fun f x = if x then 1 else 0, bool → int, []); x := 35; g := (fun g z = f (z < x) < x, int → bool, [ f:= (fun f x = if x then 1 else 0, bool → int, []); x := 35])]` für das Programm **val** `x = g 5`.



Auswertungsbaum in der Umgebung `[ ]` für das Programm:

```
fun f (x:bool) : int = if x then 1 else 0
val x = 5*7
fun g (z:int) : bool = f(z<x)<x
val x = g 5
```



Aufgabe 2.6 Betrachten Sie das folgende Programm:

```
val x = 3+2
fun f (y:int) : int = x+y
fun g (y:int) : int = if y<x then 0 else y+g(y-1)
```

a) Geben Sie die Umgebung an, die die Ausführung des Programms in der Umgebung [] liefert.

b) Geben Sie die Umgebung an, in der der Rumpf der Prozedur f bei der Ausführung des Aufrufs `f 3` mit Umgebung V = [x:=4] ausgeführt wird.

c) Geben Sie die Umgebung an, in der der Rumpf der Prozedur g bei der Ausführung des Aufrufs `g 13` mit Umgebung V = [y:=5,x:=3] ausgeführt wird.

d) Geben Sie den Auswertungsbaum in der Umgebung `[ ]` für das Programm an. **Hinweis:** Lesen Sie das Skript von der Homepage um diese Aufgabe zu lösen.

Lösungsvorschlag 2.6:

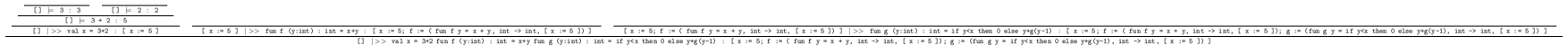
a) `[x:=5,f:=(fun f y=x+y,int→int,[x:=5]),g:=(fun g y = if y<x then 0 else y+g(y-1),int→int,[x:=5])]`

b) `([x:=5] + [f:=(fun f y = x+y,int→int,[x:=5])])+[y:=3] = ([x:=5,y:=3,f:=(fun f y = x+y,int→int,[x:=5])])`

c) `([x:=5] + [g:=(fun g y = if y<x then 0 else y+g(y-1),int→int,[x:=5])])+[y:=13] = [g:=(fun g y = if y<x then 0 else y+g(y-1),int→int,[x:=5]),x:=5,y:=13]`

d) Auswertungsbaum in der Umgebung `[ ]` für das Programm:

```
val x = 3+2
fun f (y:int) : int = x+y
fun g (y:int) : int = if y<x then 0 else y+g(y-1)
```



Aufgabe 2.7* Geben Sie einen geschlossenen Ausdruck (d. h., keine Deklaration) an, der eine Prozedur `int → int` beschreibt, die zu x das Ergebnis x^2 liefert. Geben Sie die Tripeldarstellung der durch Ihren Ausdruck beschriebenen Prozedur an. Hinweis: Verwenden Sie einen Let-Ausdruck.

Lösungsvorschlag 2.7*:

```
let fun f (x:int) = x*x in f end
```

Tripeldarstellung von `f`: (**fun** `f x = x*x`, `int→int`, [])

Aufgabe 3.1 Deklarieren Sie eine Prozedur `mul:int→int→int→int`, die das Produkt dreier Zahlen liefert. Deklarieren Sie mul auf 3 Arten: Mit einer kaskadierten Prozedurdeklaration, mit einer Prozedurdeklaration und zwei Abstraktionen, und mit einer Deklaration mit val und drei Abstraktionen.

Lösungsvorschlag 3.1:

a)
```
fun mul (x:int) (y:int) (z:int) = x*y*z
```

b)
```
fun mul (x:int) = fn (y:int) ⇒ fn (z:int) ⇒ x*y*z
```

c)
```
val mul = fn (x:int) ⇒ fn (y:int) ⇒ fn (z:int) ⇒ x*y*z
```

Aufgabe sec3.1.b Geben Sie der Typ von `f` im `fun f g h x = g (h x)` an.

Lösungsvorschlag sec3.1.b: `('a -> 'b) -> ('c -> 'a) -> 'c -> 'b`

Aufgabe 3.2 Deklarieren Sie zwei Prozeduren, die die Operation `div` (ganzzahlige Division) kartesisch und kaskadiert darstellen.

Lösungsvorschlag 3.2:

a)
```
fun div1 (x:int, y:int) = if x<y then 0 else 1 + div2 (x-y, y)
```

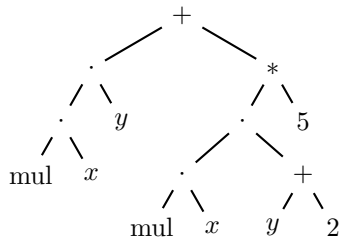
b)
```
fun div2 (x:int) (y:int) = if x<y then 0 else 1 + div2 (x-y) y
```

Aufgabe 3.3 Geben Sie die Baumdarstellung des Ausdrucks

```
mul x y + mul x (y + 2) * 5
```

an. Überprüfen Sie die Richtigkeit Ihrer Darstellung mit einem Interpreter.

Lösungsvorschlag 3.3:



Aufgabe 3.5 Geben Sie zu den folgenden Abstraktionen semantisch äquivalente Ausdrücke an, die ohne die Verwendung von Abstraktionen gebildet sind.

a) **fn** `(x : int)` ⇒ `x*x`

b) **fn** `(x : int)` ⇒ **fn** `(y : int)` ⇒ `x+y`

Hilfe: Verwenden Sie Let-Ausdrücke und Prozedurdeklarationen.

Lösungsvorschlag 3.5:

a) **let fun** `f (x : int) = x*x` **in** `f` **end**

b) **let fun** `g (x : int) (y : int) = x+y` **in** `g` **end**

Aufgabe 3.6 Geben Sie die Tripeldarstellung der Prozedur an, zu der der folgende Ausdruck auswertet:

```
(fn (x:int) ⇒ fn (b:bool) ⇒ if b then x else 7) (2+3)
```

Lösungsvorschlag 3.6: (**fn** `b` ⇒ **if** `b` **then** `x` **else** `7`, `bool`→`int`, [`x:=5`])

Aufgabe 3.7 Geben Sie die Tripeldarstellung der Prozedur an, zu der der folgende Ausdruck auswertet:

```
let val a = 7
    fun f (x:int) = a + x
    fun g (x:int) (y:int) : int = g (f x) y
in
    g (f 5)
end
```

Lösungsvorschlag 3.7:

```
( fn y = g (f x) y
, int→int
, [ f := (fun f x = a+x, int→int, [ a:= 7 ])
  , g := (fun g x y = g (f x) y, int→int→int, [ f:= ... ])
  , x := 12 ] )
```

Aufgabe 3.8.old Deklarieren Sie eine Prozedur `power : int → int → int`, die zu x und $n \geq 0$ die Potenz x^n liefert, wie folgt:

a) Mit einer kaskadierten Deklaration.

b) Mit einer Deklaration mit **val** und Abstraktionen.

Lösungsvorschlag 3.8.old:

a) `fun potenz (x : int) (n : int) = if n = 0 then 1 else x * potenz x (n - 1);`

b) `val rec potenz = fn (x : int) => fn (n : int) => if n = 0 then 1 else x * potenz x (n - 1);`

Aufgabe 3.8 Deklarieren Sie eine Prozedur `prod : (int→int)→int→int`, die für $n \geq 0$ die Gleichung prod $f\ n = 1 \cdot (f\ 1) \ldots \cdot (f\ n)$ erfüllt. Deklarieren Sie außerdem mithilfe von prod eine Prozedur `fac : int→int`, die für $n \geq 0$ die Fakultät $n!$ berechnet (siehe Aufgabe 1.26 auf S. 22). Die Prozedur `fac` soll nicht rekursiv sein.

Lösungsvorschlag 3.8:

```
fun prod (f:int→int) (n:int) = if n=0 then 1 else prod f (n-1) * f n

fun fac (x:int) = prod (fn (x:int) ⇒ x) x
```

Aufgabe 3.9 Deklarieren Sie mithilfe der höherstufigen Prozedur `sum` eine Prozedur `sum' : (int→int)→int→int→int`, die für $k \geq 0$ die Gleichung sum′ $f\ m\ k = 0 + f(m+1) \ldots + f(m+k)$ erfüllt. Die Prozedur `sum'` soll nicht rekursiv sein.

Lösungsvorschlag 3.9:

```
fun sum' (f:int→int) (m:int) (k:int) = sum (fn x⇒ f (m+x)) k
```

Aufgabe 3.10 Geben Sie die Baumdarstellungen der folgenden Typen an:

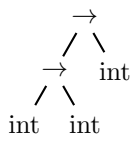
a) `(int→int)→int`

b) `int→(int*bool→int)→int`

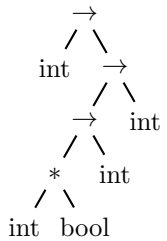
c) `(int→bool)→(bool→real)→int→real`

Lösungsvorschlag 3.10:

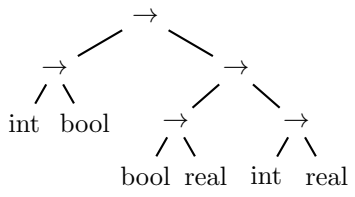
a)



b)



c)



Aufgabe 3.11 Geben Sie geschlossene Abstraktionen an, die die folgenden Typen haben:

a) `(int*int→bool)→int→bool`

b) `(int→bool→real)→int→bool→real`

c) `(int→bool)→(bool→real)→int→real`

d) `(int→real)→(bool→int)→int*bool→real*int`

Die Abstraktionen sollen nur mit Prozeduranwendungen, Tupeln und Bezeichnern gebildet werden. Konstanten und Operatoren sollen nicht verwendet werden.

Lösungsvorschlag 3.11:

a)
```
(fn (f:int*int→bool) ⇒ fn (x:int) ⇒ f (x,x))
```

b)
```
(fn (f:int→bool→real) ⇒ f)
```

c)
```
(fn (f:int→bool) ⇒ fn (g:bool→real) ⇒ fn (x:int) ⇒ g (f x))
(fn (f:int→bool) ⇒ fn (g:bool→real) ⇒ f o g)
```

d)
```
(fn (f:int→real) ⇒ fn (g:bool→int) ⇒ fn (x:int,y:bool) ⇒ (f x, g y
))
```

Aufgabe 3.12 Schreiben Sie zwei Prozeduren

a) `cas : (int*int→int)→int→int→int`

b) `car : (int→int→int)→int*int→int`

sodass `cas` zur kartesischen Darstellung einer zweistelligen Operation die kaskadierte Darstellung und `car` zur kaskadierten Darstellung die kartesische Darstellung liefert. Erproben Sie `cas` und `car` mit Prozeduren, die das Maximum zweier Zahlen liefern:

```
fun maxCas (x:int) (y:int) = if x<y then y else x
fun maxCar (x:int, y:int) = if x<y then y else x
val maxCas' = cas maxCar
val maxCar' = car maxCas
```

Wenn Sie `cas` und `car` richtig geschrieben haben, verhält sich `maxCas'` genauso wie `maxCas` und `maxCar'` genausowie `maxCar`. Hinweis:Die Aufgabe hat eine sehr einfache Lösung.

Lösungsvorschlag 3.12:

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

Aufgabe 3.13 Deklarieren Sie eine Prozedur `mul : int→int→int`, die das Produkt zweier Zahlen x und $n \geq 0$ gemäß der Gleichung

$$x \cdot n = 0 \underbrace{+x \cdots + x}_{n-mal}$$

durch Addieren berechnet. Die Prozedur `mul` soll mithilfe der Prozedur `iter` formuliert werden und nicht rekursiv sein.

Lösungsvorschlag 3.13:

```
fun mul (x:int) (y:int) = iter x 0 (fn (a:int) ⇒ a+y)
```

Aufgabe 3.14 Geben Sie eine Abstraktion `e` an, sodass die Ausführung des Ausdrucks `first x e` für alle `x` divergiert.

Lösungsvorschlag 3.14:

```
(fn (x:int) ⇒ false)
```

Aufgabe 3.15 Deklarieren Sie mit `first` eine Prozedur `divi : int→int→int`, die zu $x \geq 0$ und $m > 0$ dasselbe Ergebnis liefert wie Division mit dem Operator `div`.

Hinweis: Für $x \geq 0$ und $m > 0$ liefert `div` die größte ganze Zahl k mit $k \cdot m \leq x$.

Lösungsvorschlag 3.15:

```
fun divi (x:int) (m:int) = first 1 (fn k ⇒ k*m > x) - 1
```

Aufgabe 3.16 Deklarieren Sie mit `iter` eine Prozedur `fac`, die zu $n \geq 0$ die n-te Fakultät n! liefert.

Lösungsvorschlag 3.16: `fun fac (n:int) = #2 (iter n (1,1) (fn (c,f) ⇒ (c+1,c*f)) )`
oder
`fun fac (n:int) = let val (a,b)= iter n (1,1) (fn (c,f) ⇒ (c+1,c*f)) in b end`

Aufgabe 3.17 Deklarieren Sie mit `iter` eine Prozedur `sum`, die für n $\geq$ 0 die Gleichung `sum n f = 0 + f 1 ... + f n` erfüllt.

Lösungsvorschlag 3.17: `fun sum (n:int) (f:int → int) = #2(iter n (1,0) (fn (c,s) ⇒ ( c+1 ,s + f c )))`