



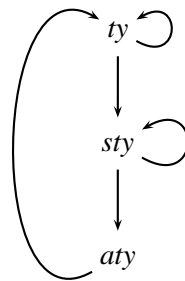
Programmierung: **Musterlösung zum 13. Übungsblatt**

Prof. Gert Smolka und Thorsten Brunklaus

Aufgabe 13.1: Grammatik und Parser für Typen mit Pfeil und Stern (3+9)

(a) (i)

$ty = sty \ [\ \text{"->"} \ ty \]$
 $sty = atty \ [\ \text{"*"} \ sty \]$
 $atty = \text{"int"} \mid \text{"(" } ty \ \text{"})"$



"->"

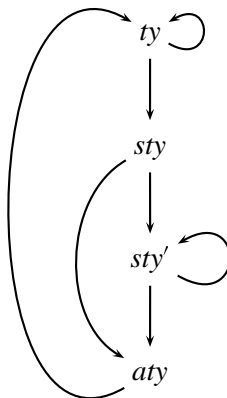
"*"

"int" "(" ")"

(ii)

(b) (i) i.

$ty = sty \ [\ \text{"->"} \ ty \]$
 $sty = atty \ [\ \text{"*"} \ sty' \]$
 $sty' = atty \ [\ \text{"*"} \ sty' \]$
 $atty = \text{"int"} \mid \text{"(" } ty \ \text{"})"$



"->"

"*"

"*"

"int" "(" ")"

ii.

```

(ii)  datatype token = INT | ARROW | STAR | LPAR | RPAR

datatype ty = Int
          | Arrow of ty * ty
          | Star of ty list

(*
   ty   = sty [ "->" ty ]
   sty  = aty [ "*" sty' ]
   sty' = aty [ "*" sty' ]
   aty  = "int" | "(" ty ")"
*)

exception Error

fun match (a,ts) t = if null ts orelse hd ts <> t
                    then raise Error
                    else (a, tl ts)

fun combine a ts p f = let val (a',tr) = p ts
                        in (f(a,a'), tr)
                        end

fun ty ts = case sty ts of
             (a, ARROW::tr) => combine a tr ty Arrow
             | ats         => ats

and sty ts = case atty ts of
             (a, STAR::tr) => combine a tr sty' (Star o op::)
             | ats         => ats

and sty' ts = case atty ts of
             (a, STAR::tr) => combine a tr sty' op::
             | (a, tr)    => ([a],tr)

and atty (INT::ts) = (Int,ts)
  | atty (LPAR::ts) = match (ty ts) RPAR
  | atty _         = raise Error

fun parse ts = case ty ts of
                (a, nil) => a
                | _      => raise Error

```

Aufgabe 13.2: Generatoren (2+2+2+3)

```

(a)  val nextSquare =
      let
        val r = ref 0
      in
        fn () => let
            val i = !r
          in
            (i * i) before r := i + 1
          end
        end
      end

```

```

(b) fun newNextSquare () =
    let
        val r = ref 0
    in
        fn () => let
            val i = !r
            in
                (i * i) before r := i + 1
            end
        end
    end

(c) fun newGenerator f =
    let
        val r = ref 0
    in
        fn () => f(!r) before r := !r+1
    end

(d) fun next y xs =
    if List.all (fn x => y mod x <> 0) xs then y
    else next (y+1) xs

fun newNextPrime () =
    let
        val r = ref []
    in
        fn () =>
            case !r of
                nil => (r := [2] ; 2)
            | x::xr => let val p = next (x+1) xr
                        in r := p:: !r ; p
                        end
    end
end

```

Aufgabe 13.3: Imperative Schlangen (7)

```

signature QUEUE =
sig
    type 'a queue
    val queue  : unit -> 'a queue
    val insert : 'a * 'a queue -> unit
    val head   : 'a queue -> 'a          (* Empty *)
    val remove : 'a queue -> unit        (* Empty *)
    val empty  : 'a queue -> bool
end

structure Queue :> QUEUE =
struct
    type 'a queue = 'a list ref
    fun queue () = ref nil
    fun insert (a,q) = q := !q @ [a]
    fun head q = hd(!q)
    fun remove q = q := tl(!q)
    fun empty q = null(!q)
end

```

Aufgabe 13.4: Rotieren von Reihungen (4+4)

(a)

```
fun rotate' a au i =
  if i<0 then Array.update(a,0,au)
  else (Array.update(a, i+1, Array.sub(a,i)) ;
        rotate' a au (i-1) )
```

```
fun rotate a =
  let
    val u = Array.length a - 1
  in
    rotate' a (Array.sub(a,u)) (u-1)
  end
```

(b)

```
fun rotate a =
  let
    val u = Array.length a - 1
    val au = Array.sub(a, u)
    val i = ref (u-1)
  in
    while !i>=0 do
      (Array.update(a, !i+1, Array.sub(a,!i)) ;
       i:= !i-1 ) ;
    Array.update(a,0,au)
  end
```

Aufgabe 13.5: Binäre Suche (8)

```
fun member' a x l u =
  if l>u then false
  else let val m = l+ (u-1) div 2
        in case Int.compare(x, Array.sub(a,m)) of
            LESS    => member' a x l (m-1)
          | EQUAL   => true
          | GREATER => member' a x (m+1) u
        end
```

```
fun member a x = member' a x 0 (Array.length a - 1)
```

Aufgabe 13.6: Statische Semantik von F mit Referenzen (6)

$$\frac{T \vdash e \Rightarrow t}{T \vdash \text{ref } e \Rightarrow t \text{ ref}} \quad \frac{T \vdash e \Rightarrow t \text{ ref}}{T \vdash !e \Rightarrow t} \quad \frac{T \vdash e_1 \Rightarrow t \text{ ref} \quad T \vdash e_2 \Rightarrow t}{T \vdash e_1 := e_2 \Rightarrow \text{unit}}$$