

Compilers: Assignment #2

Due on Sunday, November 29, 2015

Resubmit: Task 1, 2

Mirza Hasanbasic

Indhold

Task 1	2
Task 2	3

Task 1**a)**

$$([og][og])^* | ([og][og][og])^*$$

Her vil vi have et alfabet, hvor sekvensen af længden er delelig med 2 eller 3.

b)

$$T \rightarrow aTb$$

$$T \rightarrow R$$

$$R \rightarrow c$$

$$R \rightarrow cR$$

Vil producere en context-free grammer, hvor c er tilladt til at være alene, c vil altid være mellem a og b og der vil, for hvert a være et b .

c)**i)**

Når man har *% prec letprec*, så specificerer det en regel, som er associativ med *letprec* i dette tilfælde. I *% nonassoc* har man indikeret at *letprec* ikke er associativ, altså vores *LET* og *IN*.

ii)**FIX:**

Det som *let* gør, er at, der gives et eller flere udtryk, som deklarerer en værdi, hvor disse værdier så udgør et resultat til sidst.

Hvor den første variabel består af en streng, et udtryk og så positionen af angivelsen. Altså, hvis så kigger på `(Dec (#1 $2, $4, $3), $6, $1)`, så har vi, at `#1 $2` vil referere til den første parameter og andet symbol, ID, som er en streng der holder navnene på variablene.

`$4` vil så referere til parameteren af det fjerde i udtrykket som det vil blive båndet til. Hvor `$3` så er symbolet EQ, da det er på denne position, og dermed vil det være til et lighedstegn, som den vil blive båndet til.

For at kigge på de sidste to, kigger vi på hele udtrykket, altså `Let (Dec (#1 $2, $4, $3), $6, $1)`. Her har vi at `$6` referer til hvilken erklæring af variabelen, `(Dec)` vil blive brugt og hvor `$1` referer til positionen af *let* symbolet

Task 2

a)

For filter, vil det altså være

$$\forall \alpha \quad ((\alpha \rightarrow \text{bool}) * [\alpha]) \rightarrow [\alpha]$$

FIX:

b)

```

1 CheckExp(Exp, vtable, ftable) = case Exp of
2
3   filter(p, arr_exp) ⇒
4     tarr = CheckExp(arrexp, vtable, ftable)
5     tel = case tarr of
6       Array(t1) → t1
7       | other → error()
8
9     tf = lookup(ftable, name(p))
10    case tf of
11      (tin → tout) ⇒ if tin = tel and tout = bool
12                      then tarr
13                      else error()
14    | _ ⇒ error()
```

c)

```

1 CheckExp(Exp, vtable, ftable) = case Exp of
2
3   filter( Lambda(ret_type, id_type_lst, body), arr_exp) ⇒
4     tbody = CheckExp(body, vtable, ftable)
5     tarr = CheckExp(arrexp, vtable, ftable)
6     tel = case tarr of
7       Array(t1) → t1
8       | other → error()
9     tin = case id_type_lst of
10      [(_, t2)] → t2
11      | other → error()
12     if ret_type = bool and tbody = bool and tel = tin
13     then tarr
14     else error()
```