KØBENHAVNS UNIVERSITET

IPS - Assignment 2

Mikkel Willén, bmq419

2. maj 2023

Indhold

1 Task

1.1 Alphabet $\sum = \{o, g\}$

a)

$$([og] \cdot [og])^*$$

Inde i parentesen kan der vælges 2 karaktere fra alfabetet. Dette kan så vælges mindst 0 gange.

b)

$$[o] \cdot [og] \cdot ([og] \cdot [og])^*$$

Til at starte med, vil det første karakter være o, efterfuldt af et andet tal fra alfabetet. Derefter er det, det samme som opgave a).

c)

$$([o] \cdot [o])^* \cdot ([o] \cdot [g])? \cdot ([g] \cdot [g])^*$$

Der er først et lige antal o'er, efterfuldt af potentielt et og. Til sidst er der så et lige antal g'er.

1.2 Alphabet $\sum = \{a, b, c\}$

a)

$$\begin{aligned} T &\to R \\ T &\to aTb \\ R &\to cR \\ R &\to \end{aligned}$$

Det starter med T, som kan blive til et R eller et lige antal a'er og b'er. R'et bliver sat ind i midten af a'erne og b'erne og bliver til et antal c'er.

b)

$$\begin{aligned} A &\to aTbb \\ T &\to aTbb \\ T &\to \end{aligned}$$

Det starter med at der bliver sat a ind med b'er ind med et T i midten. T kan så blive til flere a'er og dobbelt så mange b'er, eller ingenting.

c)

$$\begin{aligned} A &\to aA \\ A &\to T \\ T &\to aTb \\ T &\to \end{aligned}$$

Det starter med et A, som endten kan blive til et a med et A bagefter, eller til et T. T'et kan så blive til et a med et b efter med et T i midten eller til ingenting.

1.3 Parser implementation snippet

a)

Det ville ikke umiddelbart gøre en forskel, hvis man ikke skrev %nonassoc letpret, da den øverste alligevel binder mindst. Dog vil det give en masse warnings, når man prøver at køre programmet.

b)

Hvis vi puttede det inde mellem de 2 %left, ville let binde tættere end DEQ og LTH. En funktion som:

```
let x = 0 in x == 1
```

Vil derfor ændre betydning.
Hvis vi ikke bytter rundt, vil let blive bundet til sandhedsværdien af x == 1. Hvis vi bytter rundt, vil let x = 0 in x blive sammenlignet med 1, og der vil derfor ikke blive bundet noget, men bare retuneret en sandshedsværdi.

c)

fst dec og let repræsentere funktionskald fra fsp filen, men de andre repræsentere det der står på linjen over. Altså vil $1 blive være det første der står i linjen over, altså LET, mens $2 vil være ID, osv.

2 Task

Svaret til denne opgave, kan findes i vedlagt .zip fil.

3 Task

a)

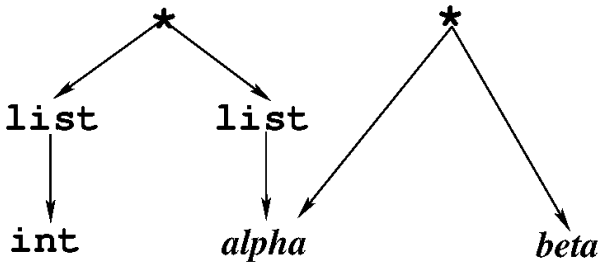
$$\text{filter } \forall a : (a \to bool) * [a] \to [a]$$

b)

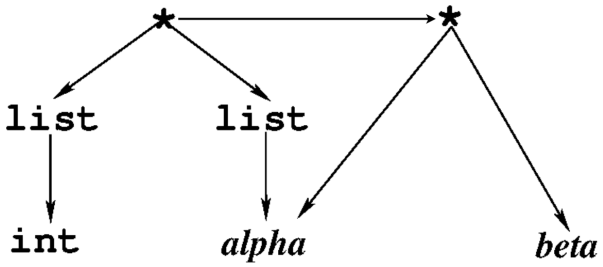
```
CheckExp(Exp, vtable, ftable) case Exp of
    filter(p, arrExp) =>
        tArr = CheckExp(arrExp, vtable, ftable)
        tEl = case tArr of
              Array(t1) -> t1
            | other -> error()
        tF = lookup(ftable, name(p))
            case tF of
                  unbound -> error()
                | (tIn -> tOut) =>
                    if tIn = tEl && tOut == bool then
                        Array(tEl)
                    else
                        error()
                | otherwise => error()
```

4 Task

Herunder er gennemgang af unification algoritmen.



Til starte med kigger algoritmen på de to op-nodes. Da de ikke er i samme set, og da de begge har 2 børn, bliver de sat sammen, og algoritmen bliver kaldt henholdsvis på deres venstre børn, og deres højre børn.



For det rekursive kald til de venstre børn, vil det falde ind i 4 case, da alpha er en variabel. De bliver derfor sat sammen, og algoritmen returnerer true. Alpha bliver derfor en int list. For det andet rekursive, vil det også falde ind i 4 case, da beta er en variabel. De bliver derfor også sat sammen, og algoritmen returnerer true. Beta bliver derfor en alpha list, som er en int list, og beta bliver derfor en int list lis.