Grammatica’s

Reguliere Expressies

Symbolen, invoertekens die herkend moeten worden: characters, digits, leesteken, etc

Operaties: [], |, +, *

Voorbeelden:

(0|1(0|1)*)[.(0|(0|1)*1)]

Reguliere expressies zijn equivalent met eindige automaten

Reguliere Expressies

Soms is het handig om een reguliere expressie op te delen en namen te gebruiken:

digit = 0|1|2|3|4|5|6|7|8|9

nzdig = 1|2|3|4|5|6|7|8|9

number = 0 | nzdig digit* [.(0|digit* nzdigit)]

Dit is alleen om het de lezer wat gemakkelijker te maken.

Probleem Reguliere Expressies

Een aantal nullen gevolgd door net zoveel enen kan niet worden uitgedrukt in een RE.

Poging: 01 | 0011 | 000111 | 00001111| …..

Probleem: dit eindigt niet!

Hoe op te lossen?

Maak gebruik van namen en recursie!

S = 01 | 0 S 1

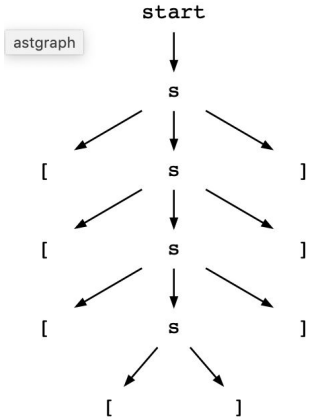
Grammatica’s: Recursieve Reguliere Expressies

S = [] | [S]

S komt voor in de definitie van S (recursie)

Invoer [[[[]]]]

Parsetree



Rekenkundige Expressies

```
3
3+5
4*(5+6-2)
```

Eerste poging zonder haakjes en zonder rekening te houden met prioriteiten

```
expressie = getal (oper getal)*
oper      = ‘+’ | ‘*’ | ‘/’ | ‘-’
```

Vraag: is dit een nog een reguliere expressie?

Rekenkundige Expressies

Tweede poging met haakjes

```
expressie = factor (oper factor)*
oper      = ‘+’ | ‘*’ | ‘/’ | ‘-’
factor    = getal | ‘(‘ expressie ‘)’
```

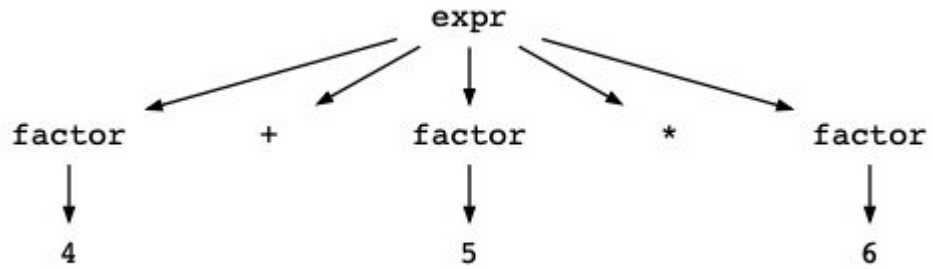
Vraag: is dit een nog een reguliere expressie?

Vraag: hoe zit het met prioriteiten?

Rekenkundige Expressies

Wat gaat er fout met de prio’s?

Teken de parsetree van: 4 + 5*6



Rekenkundige Expressies

Repareren met extra niveau

```
expressie = term   (plusoper term)*
term      = factor (multoper factor)*
plusoper  = ‘+’ | ‘-’
multoper  = ‘*’ | ‘/’
factor    = getal | ‘(‘ expressie ‘)’
```