Parsers, prioritet och associativitet

TDP019 Projekt: Datorspråk
Föreläsning 3

LINKÖPINGS UNIVERSITET

Pontus Haglund
Institutionen för datavetenskap
2024-02-17

Översikt

- Grundtanken med top-down-parsning
- Naiva grammatiken för aritmetik
- Omskrivning av regler
- Bottom-up-parsning?
- Unära operatorer och associativitet

Utvecklarblogg | Språkdagbok

- https://www.ida.liu.se/~TDP019/current/sprakdagbok/index.sv.shtml

top-down-parsning

- Strategi för att ”parsa”
- Har en ”hypotes” kontrollerar om strukturen passar
- Kan se på det som att varje regel är ett funktionsanrop

top-down-parsning

- Strategi för att ”parsa”
- Har en ”hypotes” kontrollerar om strukturen passar
- Kan se på det som att varje regel är ett funktionsanrop

```
E → E + T
  | E − T
  | T
```

top-down-parsning

- Strategi för att ”parsa”
- Har en ”hypotes” kontrollerar om strukturen passar
- Kan se på det som att varje regel är ett funktionsanrop

```
E → E + T   ”1+2”
  | E − T
  | T
```

top-down-parsning

- Strategi för att ”parsa”
- Har en ”hypotes” kontrollerar om strukturen passar
- Kan se på det som att varje regel är ett funktionsanrop

```
E → E + T   ”1+2”   E(1,+,2)
  | E – T
  | T
```

top-down-parsning

- Strategi för att ”parsa”
- Har en ”hypotes” kontrollerar om strukturen passar
- Kan se på det som att varje regel är ett funktionsanrop

```
E → E + T   ”1+2”   E(1,+,2)
  | E – T           E(+(1,2))
  | T
```

top-down-parsning

- Strategi för att ”parsa”
- Har en ”hypotes” kontrollerar om strukturen passar
- Kan se på det som att varje regel är ett funktionsanrop

```
E → E + T   ”1+2”   E(1,+,2)
  | E – T           E(+(1,2))
  | T               E(+(E(T(1)),E(T(2)))
```

top-down-parsning

- Strategi för att ”parsa”
- Har en ”hypotes” kontrollerar om strukturen passar
- Kan se på det som att varje regel är ett funktionsanrop

```
E → E + T   ”1+2”   E(1,+,2)
  | E – T           E(+(1,2))
  | T               E(+(E(T(1)),E(T(2)))
```

- I sin enklaste form utforskar denna typ av parser vänsterledet i sin helhet först

Utforska vänsterledet i sin helhet

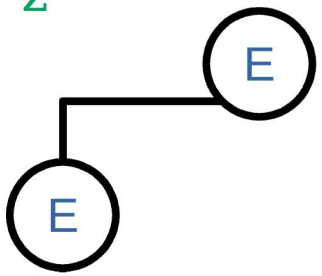

```
<E> ::= <E> + <E> | <E> * <E> | <V>
<V> ::= x | y | z
```


```
y + z
```

Utforska vänsterledet i sin helhet


```
<E> ::= <E> + <E> | <E> * <E> | <V>
<V> ::= x | y | z
```

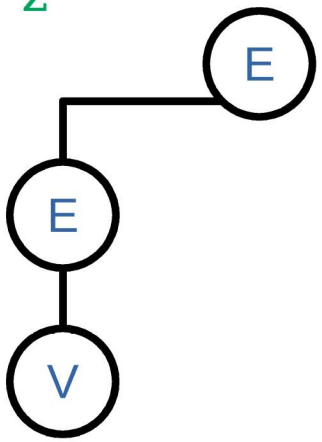
```
y + z
```



Utforska vänsterledet i sin helhet


```
<E> ::= <E> + <E> | <E> * <E> | <V>
<V> ::= x | y | z
```

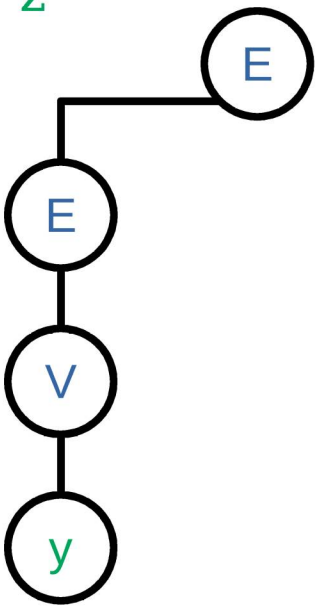
```
y + z
```



Utforska vänsterledet i sin helhet


```
<E> ::= <E> + <E> | <E> * <E> | <V>
<V> ::= x | y | z
```

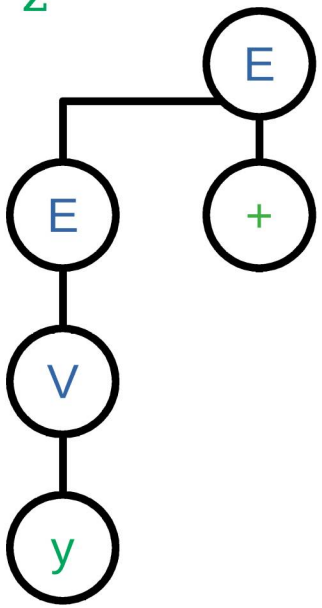
```
y + z
```



Utforska vänsterledet i sin helhet


```
<E> ::= <E> + <E> | <E> * <E> | <V>
<V> ::= x | y | z
```

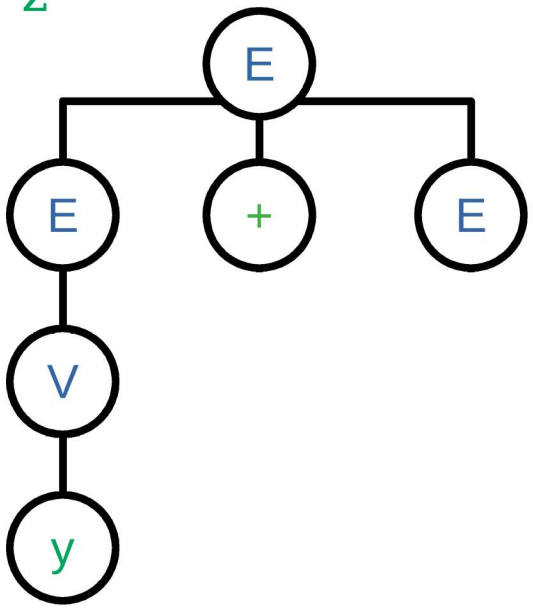
```
y + z
```



Utforska vänsterledet i sin helhet


```
<E> ::= <E> + <E> | <E> * <E> | <V>
<V> ::= x | y | z
```

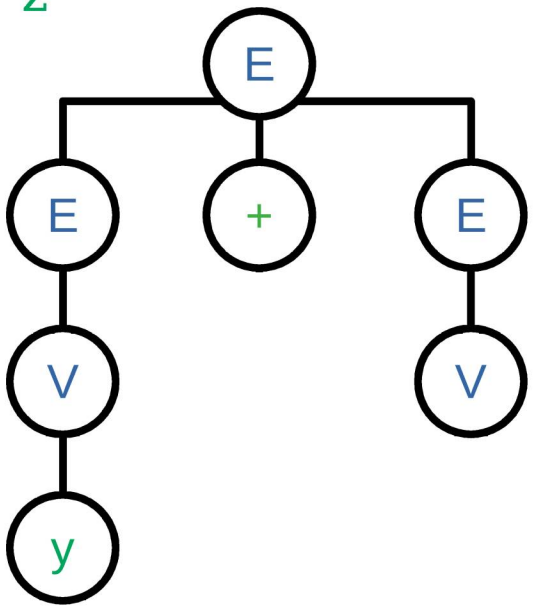
```
y + z
```



Utforska vänsterledet i sin helhet


```
<E> ::= <E> + <E> | <E> * <E> | <V>
<V> ::= x | y | z
```

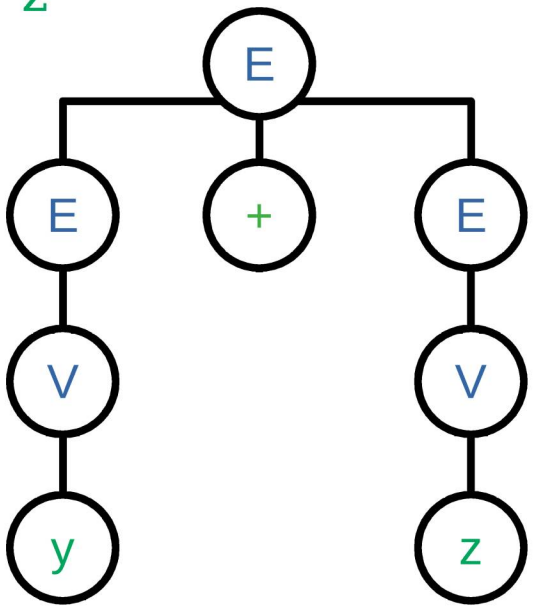
```
y + z
```



Utforska vänsterledet i sin helhet


```
<E> ::= <E> + <E> | <E> * <E> | <V>
<V> ::= x | y | z
```

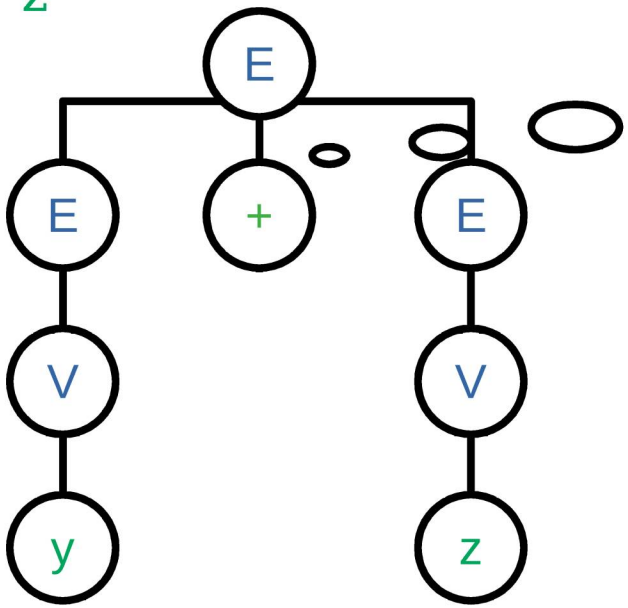
```
y + z
```

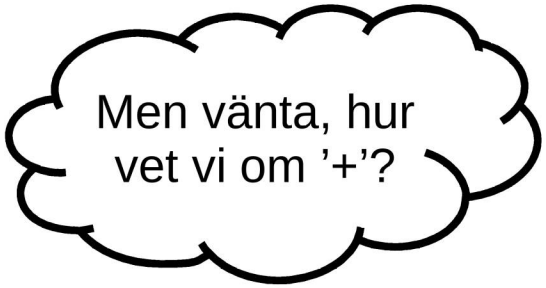


Utforska vänsterledet i sin helhet


```
<E> ::= <E> + <E> | <E> * <E> | <V>
<V> ::= x | y | z
```

y + z





Problem som kan uppstå

LINKÖPINGS UNIVERSITET

Oändlig rekursion


```
<E> ::= <E> + <E> | <E> * <E> | <V>
<V> ::= x | y | z
```

```
y + z
```

Oändlig rekursion


```
<E> ::= <E> + <E> | <E> * <E> | <V>
<V> ::= x | y | z
```

y + z

Oändlig rekursion

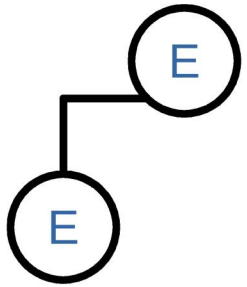

```
<E> ::= <E> + <E> | <E> * <E> | <V>
<V> ::= x | y | z
```


```
y + z
```

Oändlig rekursion


```
<E> ::= <E> + <E> | <E> * <E> | <V>
<V> ::= x | y | z
```

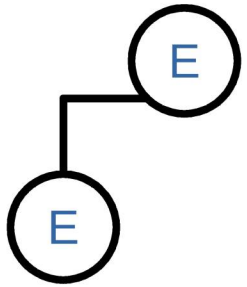
```
y + z
```



Oändlig rekursion


```
<E> ::= <E> + <E> | <E> * <E> | <V>
<V> ::= x | y | z
```

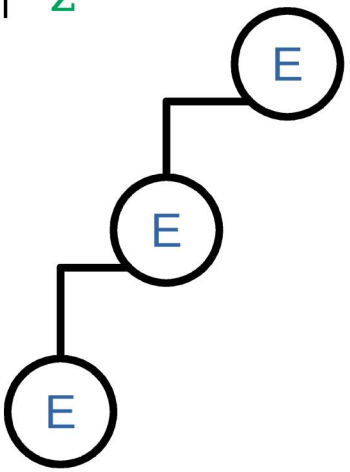
```
y + z
```



Oändlig rekursion

<E> ::= <E> + <E> | <E> * <E> | <V>
<V> ::= x | y | z

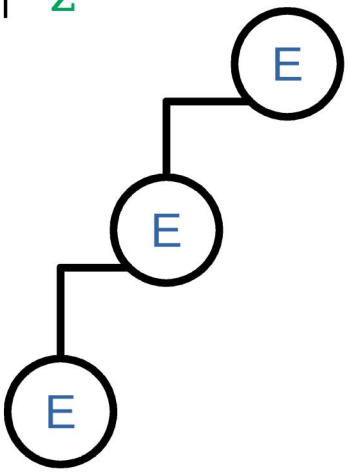
y + z



Oändlig rekursion


```
<E> ::= <E> + <E> | <E> * <E> | <V>
<V> ::= x | y | z
```

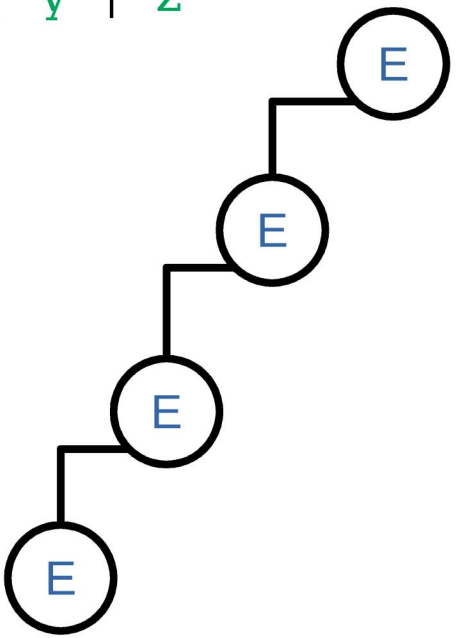
```
y + z
```



Oändlig rekursion


```
<E> ::= <E> + <E> | <E> * <E> | <V>
<V> ::= x | y | z
```

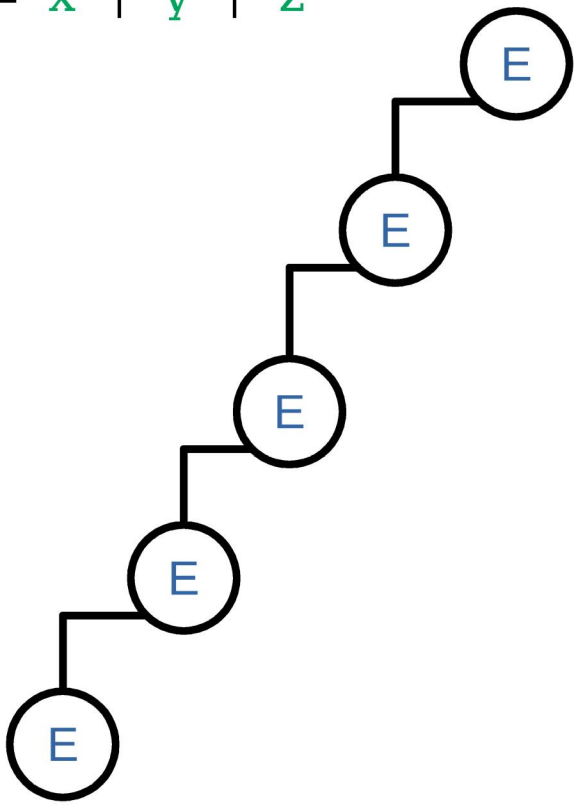
y + z



Oändlig rekursion


```
<E> ::= <E> + <E> | <E> * <E> | <V>
<V> ::= x | y | z
```

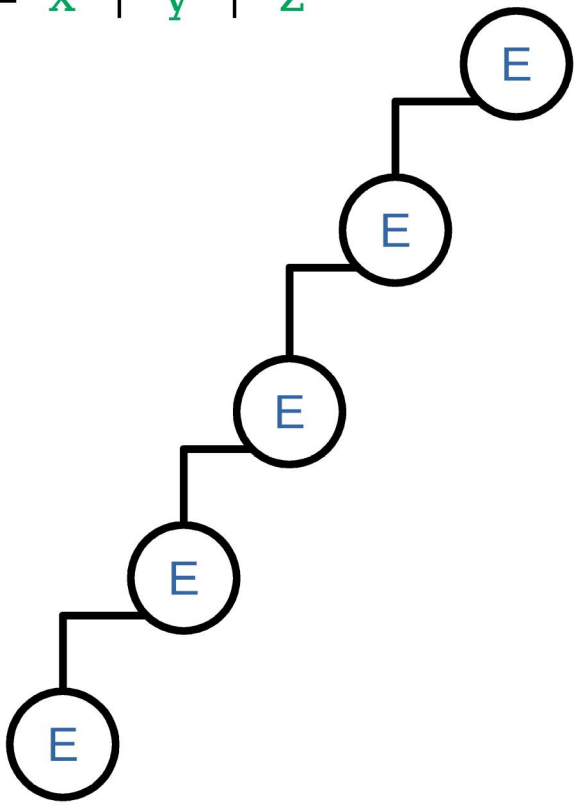
y + z



Oändlig rekursion


```
<E> ::= <E> + <E> | <E> * <E> | <V>
<V> ::= x | y | z
```

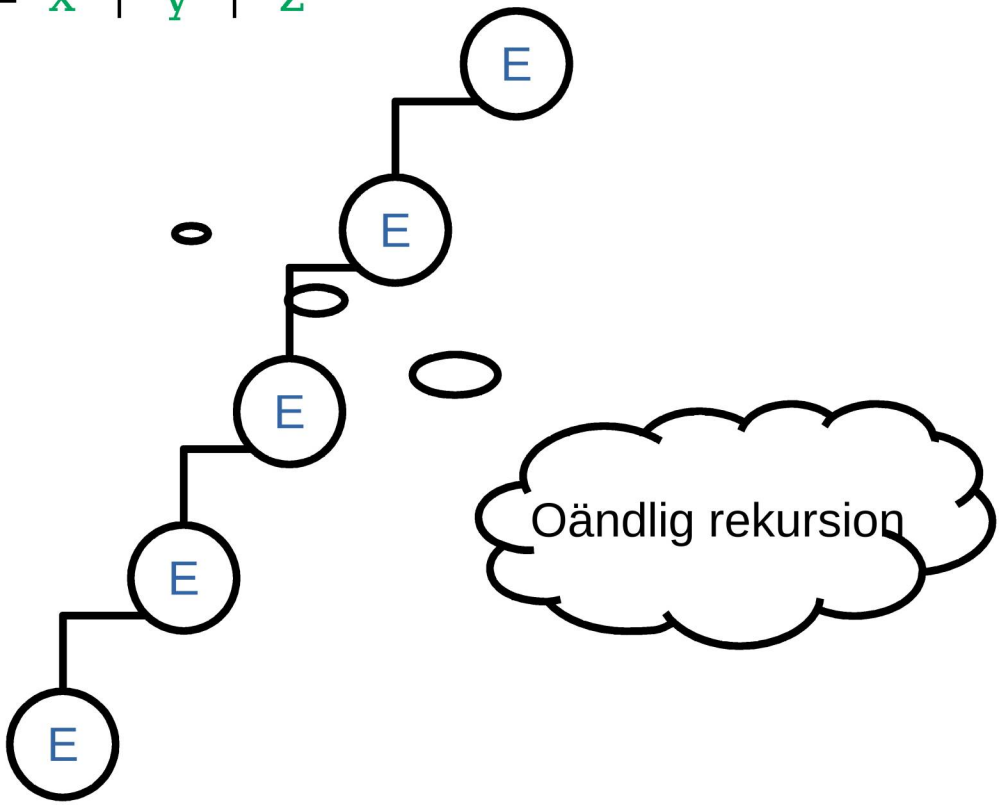
y + z



Oändlig rekursion


```
<E> ::= <E> + <E> | <E> * <E> | <V>
<V> ::= x | y | z
```

y + z

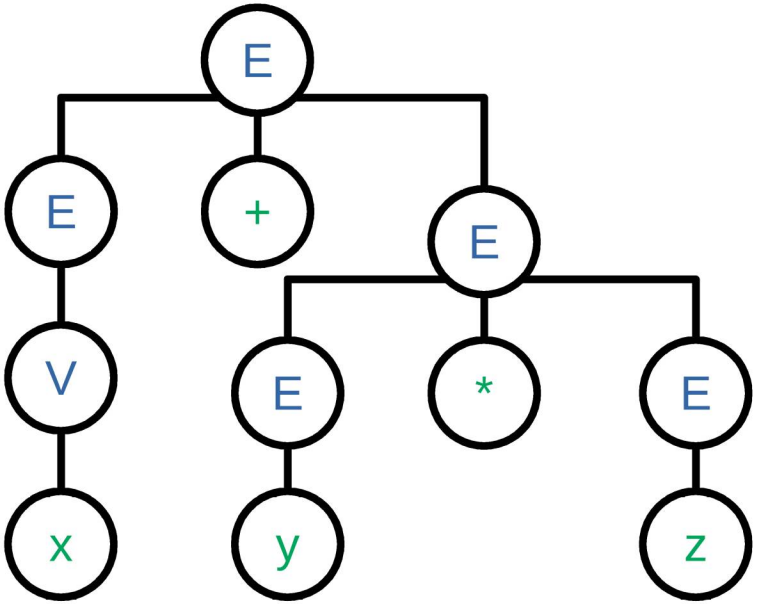


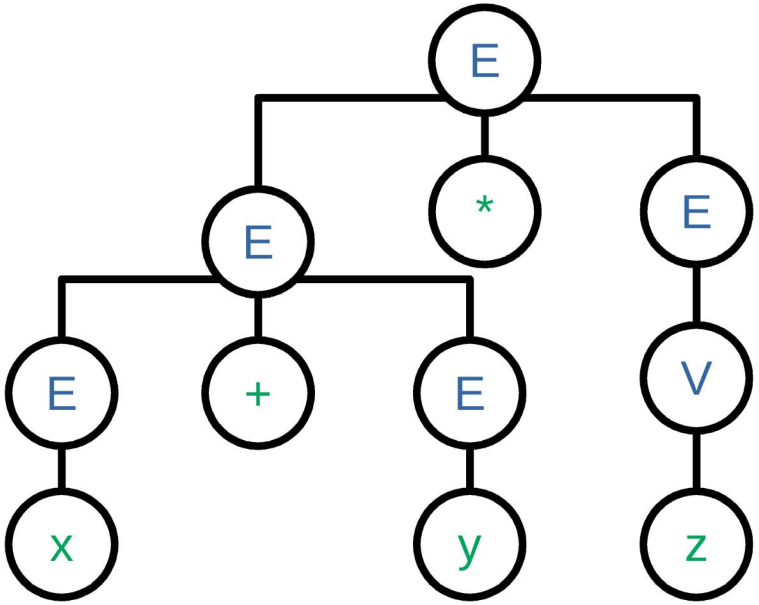
Är detta ett problem?

- Ja och nej
- Rdparse är lite smartare
- Men om vi hade en dummare parser?

Tvetydighet


```
<E> ::= <E> + <E> | <E> * <E> | <V>
<V> ::= x | y | z

x + y * z
```





Är detta ett problem?

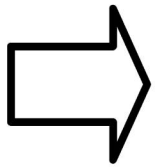
- Ja och nej
- Rdparse kommer vara konsekvent, men…
- Rätt sätt att lösa detta på är att skriva entydig grammatik

```
E → E + T
  | E - T
  | T
```

Är detta ett problem?

- Ja och nej
- Rdparse är lite smartare
- Men om vi hade en dummare parser?

```
E → E + T
  | E - T
  | T
```

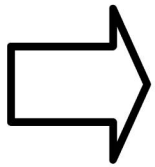


```
E  → T E'

E' → + T E'
   | - T E'
   | <empty>
```

Är detta ett problem?

- Ja och nej
- Rdparse är lite smartare
- Men om vi hade en dummare parser?

```
E → E + T
  | E - T
  | T
```



```
E  → T E'

E' → + T E'
   | - T E'
   | <empty>
```

Insikt: Problemet är att det vi vill skilja på måste vara längst till vänster

Se till att skillnaden är längst till vänster

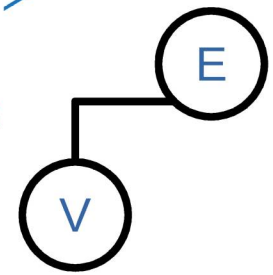

```
<E> ::= <V> <E'>
<E'>::= + <V> <E'>
      | <empty>
<V> ::= x | y | z
```

y + z

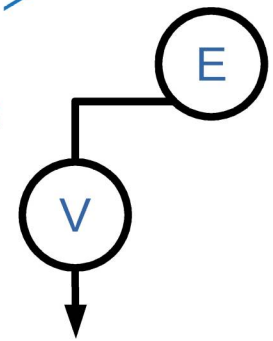
```
<E> ::= <V> <E'>
<E'>::= + <V> <E'>
      | <empty>
<V> ::= x | y | z
```

y + z

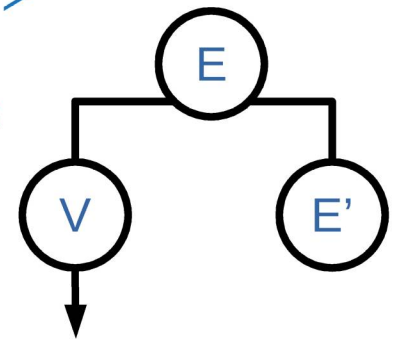
```
<E> ::= <V> <E'>
<E'>::= + <V> <E'>
      | <empty>
<V> ::= x | y | z

y + z
```



```
<E> ::= <V> <E'>
<E'>::= + <V> <E'>
      | <empty>
<V> ::= x | y | z

y + z
```

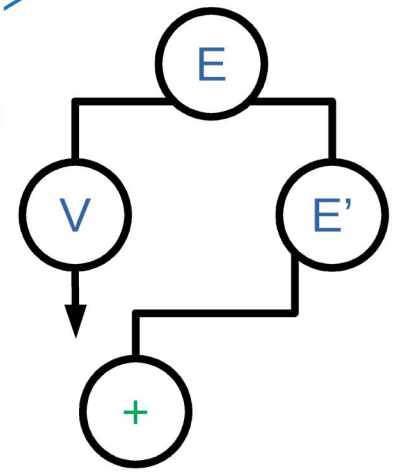


```
<E> ::= <V> <E'>
<E'>::= + <V> <E'>
      | <empty>
<V> ::= x | y | z

y + z
```



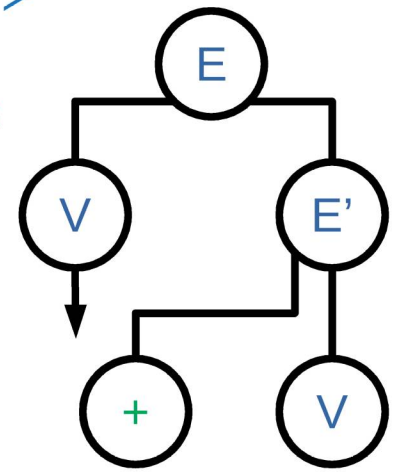
```
<E> ::= <V> <E'>
<E'>::= + <V> <E'>
      | <empty>
<V> ::= x | y | z
```

y + z



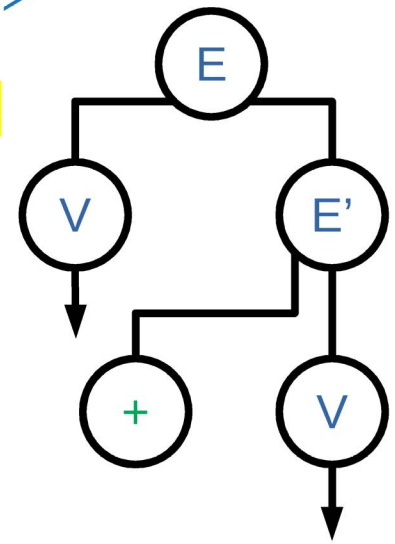
```
<E> ::= <V> <E'>
<E'>::= + <V> <E'>
      | <empty>
<V> ::= x | y | z
```

y + z

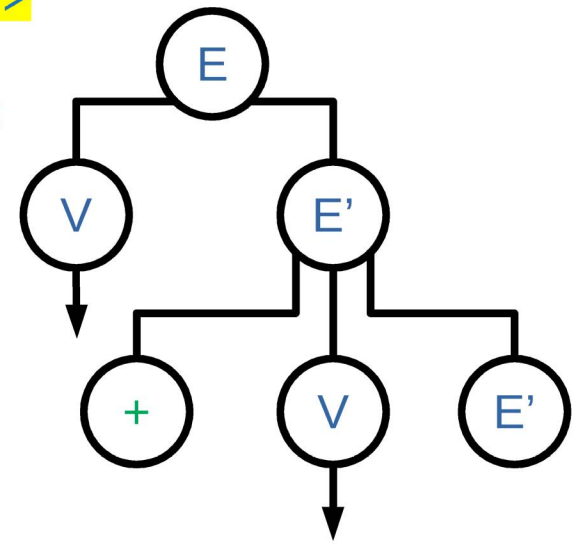


```
<E> ::= <V> <E'>
<E'>::= + <V> <E'>
      | <empty>
<V> ::= x | y | z
```

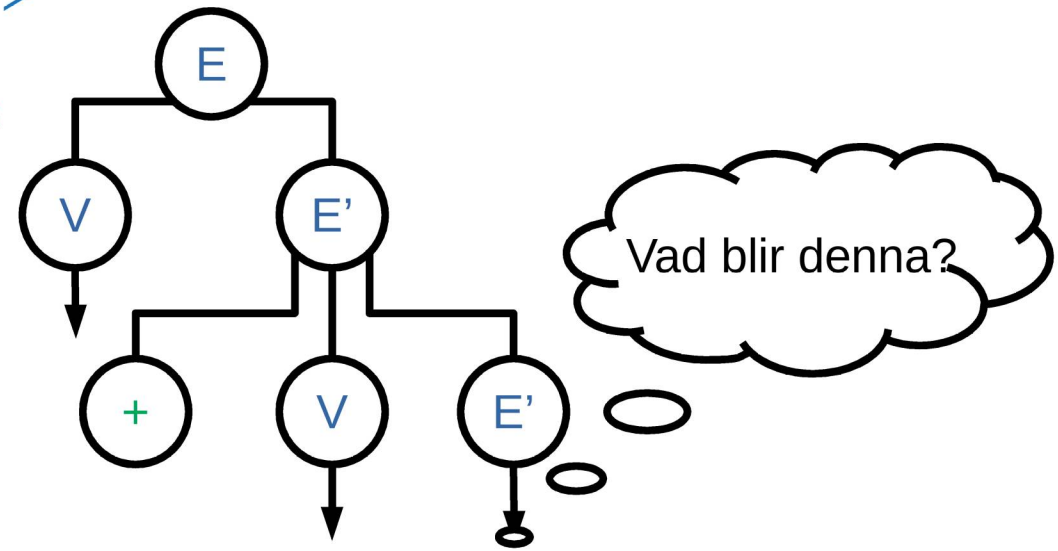
y + z



```
<E> ::= <V> <E'>
<E'>::= + <V> <E'>
      | <empty>
<V> ::= x | y | z

y + z
```

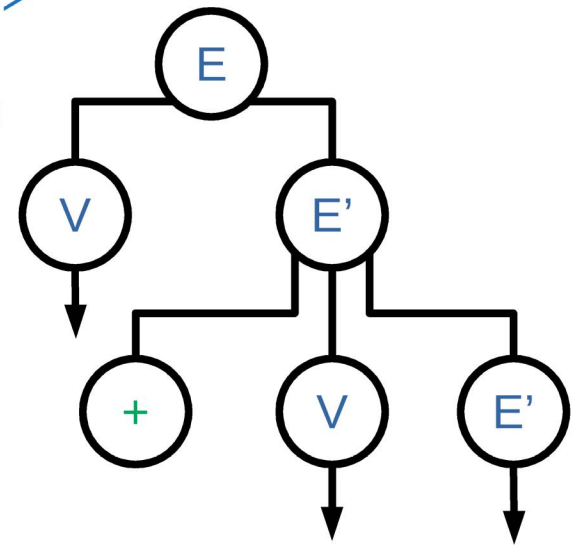


```
<E> ::= <V> <E'>
<E'>::= + <V> <E'>
      | <empty>
<V> ::= x | y | z

y + z
```



```
<E> ::= <V> <E'>
<E'>::= + <V> <E'>
      | <empty>
<V> ::= x | y | z
```

y + z



Om det är komplexare uttryck?

Utbyggbart

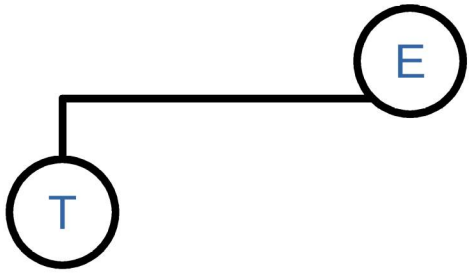

```
<E> ::= <T> <E'>

<E'>::= + <T> <E'>
      | <empty>

<T> ::= <V> <T'>

<T'>::= * <V> <T'>
      | <empty>

<V> ::= x | y | z

x * y + z
```

Utbyggbart


```
<E> ::= <T> <E'>

<E'>::= + <T> <E'>
      | <empty>

<T> ::= <V> <T'>

<T'>::= * <V> <T'>
      | <empty>

<V> ::= x | y | z

x * y + z
```

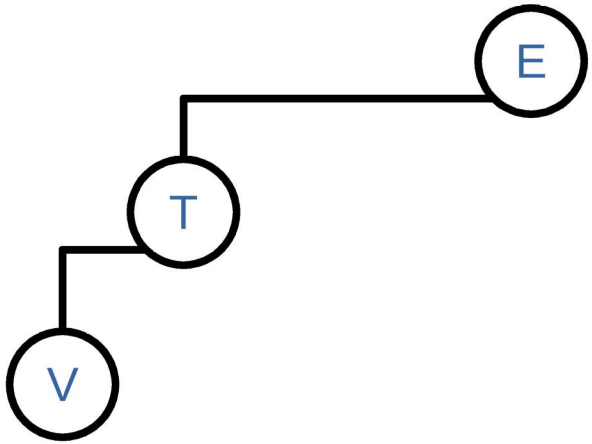
Utbyggbart


```
<E> ::= <T> <E'>

<E'>::= + <T> <E'>
      | <empty>

<T> ::= <V> <T'>

<T'>::= * <V> <T'>
      | <empty>

<V> ::= x | y | z

x * y + z
```



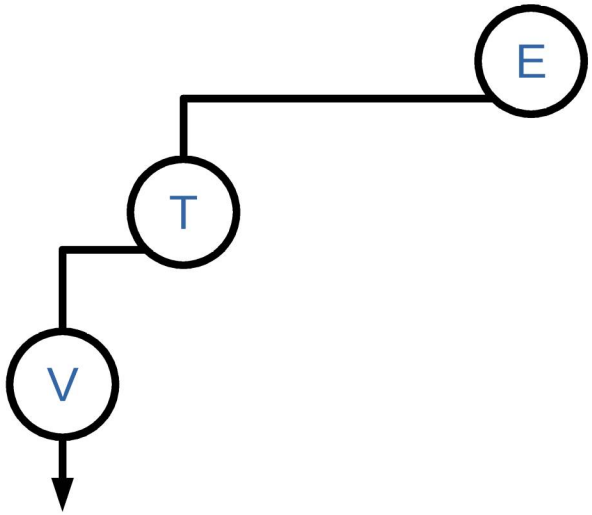
Utbyggbart


```
<E> ::= <T> <E'>

<E'>::= + <T> <E'>
      | <empty>

<T> ::= <V> <T'>

<T'>::= * <V> <T'>
      | <empty>

<V> ::= x | y | z

x * y + z
```



Utbyggbart

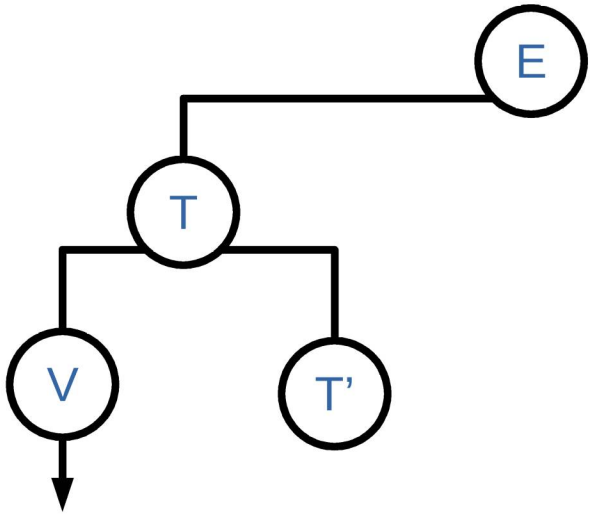

```
<E> ::= <T> <E'>

<E'>::= + <T> <E'>
      | <empty>

<T> ::= <V> <T'>

<T'>::= * <V> <T'>
      | <empty>

<V> ::= x | y | z
```



x * y + z

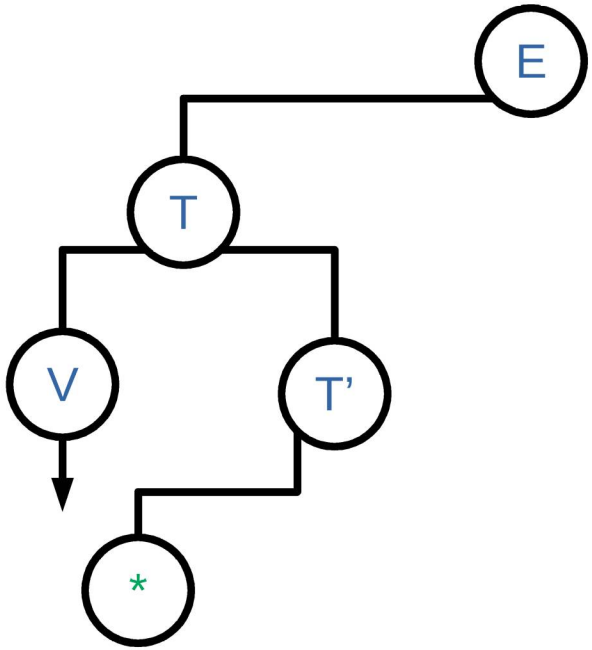
Utbyggbart


```
<E> ::= <T> <E'>

<E'>::= + <T> <E'>
      | <empty>

<T> ::= <V> <T'>

<T'>::= * <V> <T'>
      | <empty>

<V> ::= x | y | z

x * y + z
```



Utbyggbart

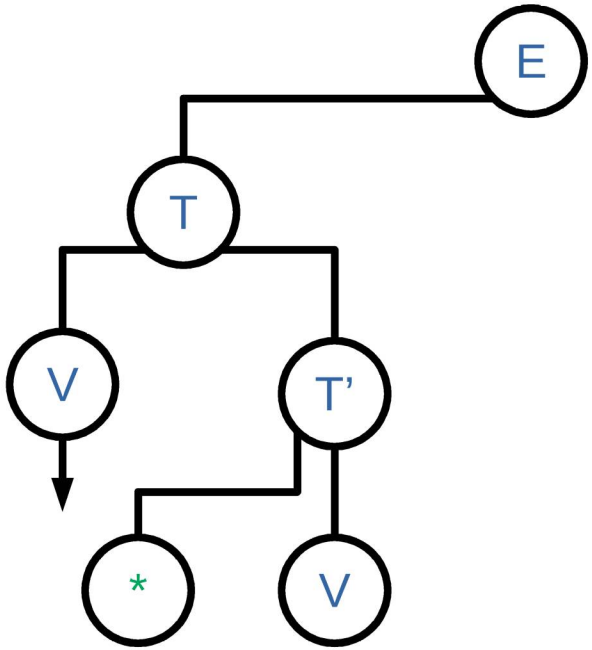

```
<E> ::= <T> <E'>

<E'>::= + <T> <E'>
      | <empty>

<T> ::= <V> <T'>

<T'>::= * <V> <T'>
      | <empty>

<V> ::= x | y | z
```



x * y + z

Utbyggbart

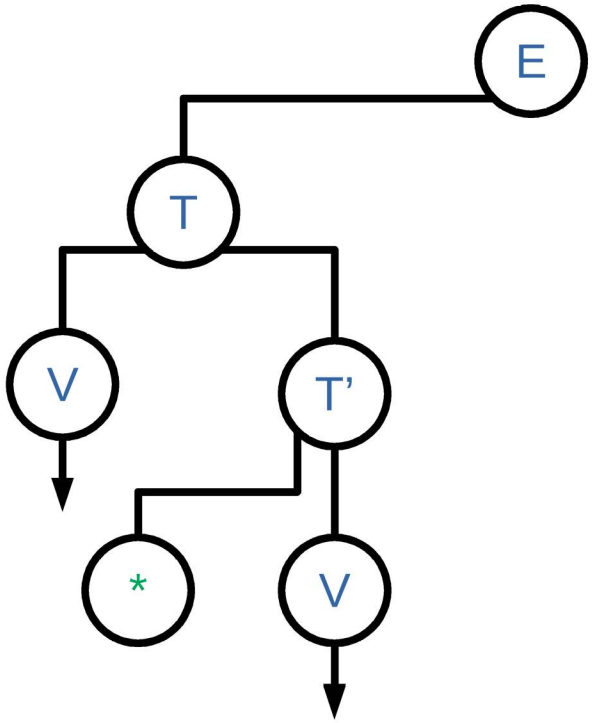

```
<E> ::= <T> <E'>

<E'>::= + <T> <E'>
      | <empty>

<T> ::= <V> <T'>

<T'>::= * <V> <T'>
      | <empty>

<V> ::= x | y | z

x * y + z
```



Utbyggbart

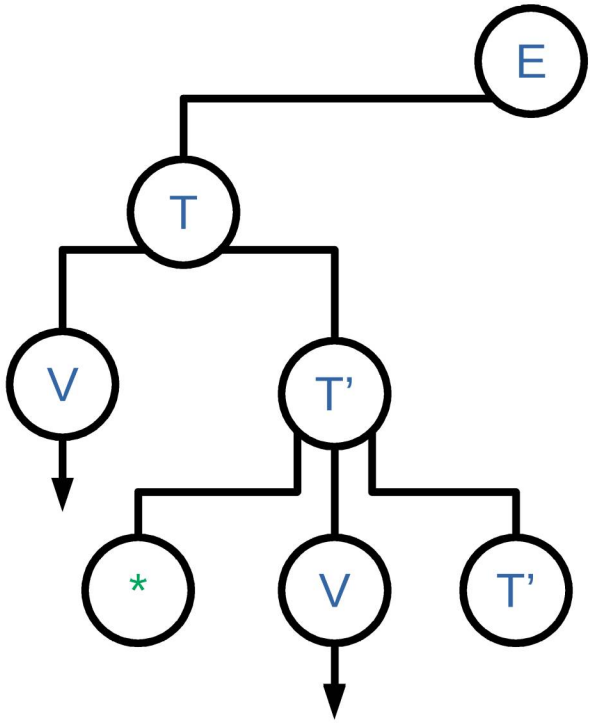

```
<E> ::= <T> <E'>

<E'>::= + <T> <E'>
      | <empty>

<T> ::= <V> <T'>

<T'>::= * <V> <T'>
      | <empty>

<V> ::= x | y | z
```

x * y + z



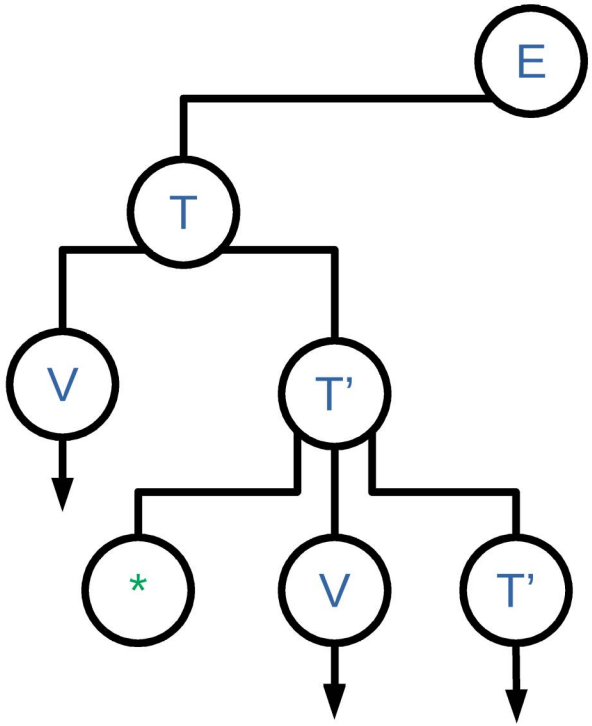
Utbyggbart


```
<E> ::= <T> <E'>

<E'>::= + <T> <E'>
      | <empty>

<T> ::= <V> <T'>

<T'>::= * <V> <T'>
      | <empty>

<V> ::= x | y | z

x * y + z
```



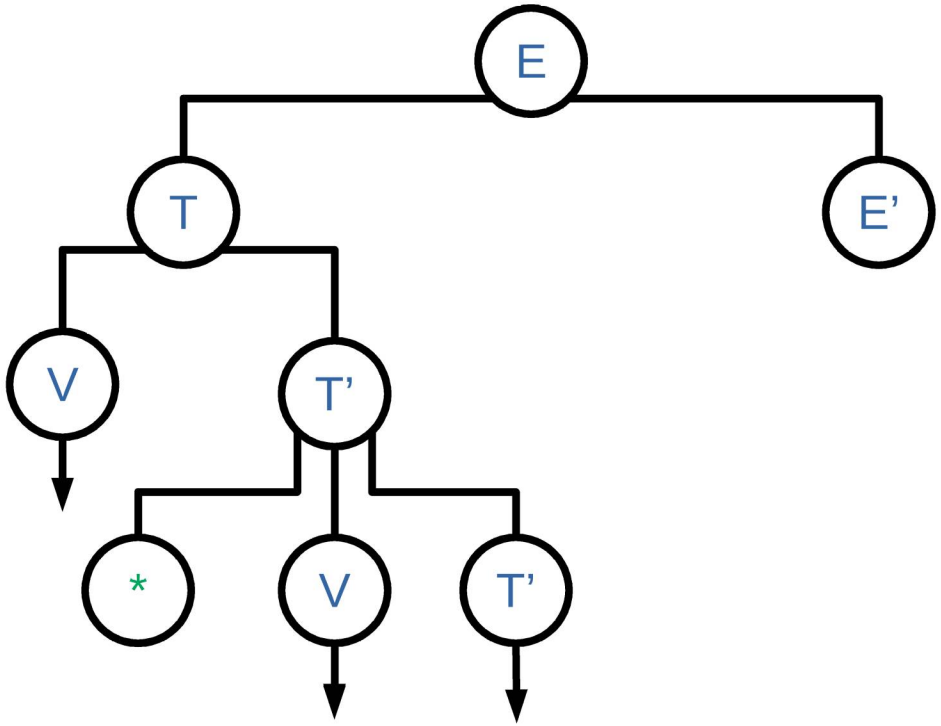
Utbyggbart


```
<E> ::= <T> <E'>

<E'>::= + <T> <E'>
      | <empty>

<T> ::= <V> <T'>

<T'>::= * <V> <T'>
      | <empty>

<V> ::= x | y | z

x * y + z
```



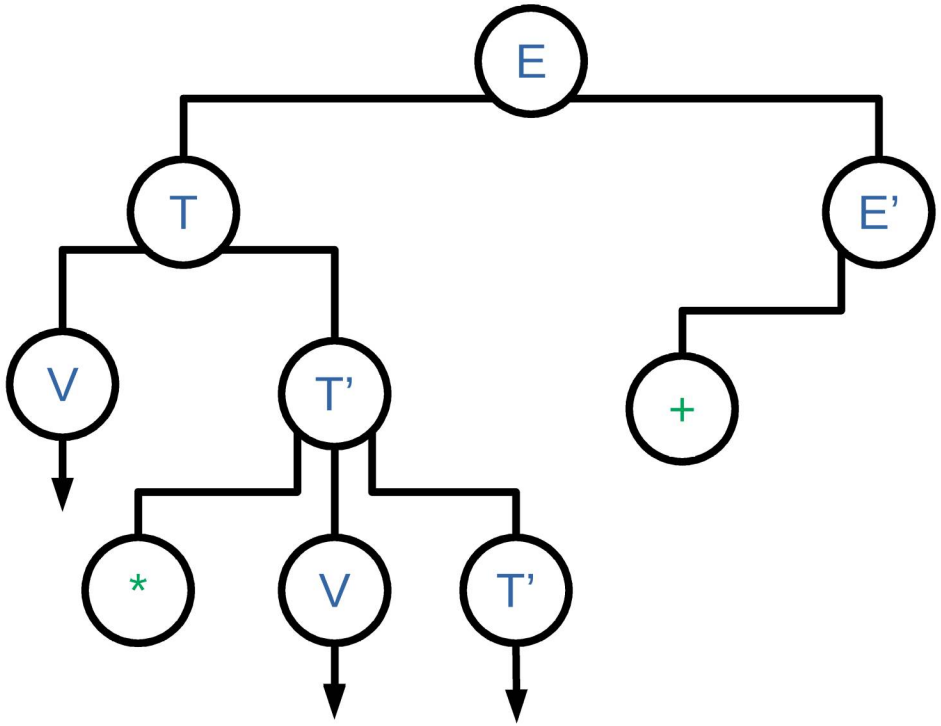
Utbyggbart


```
<E> ::= <T> <E'>

<E'>::= + <T> <E'>
      | <empty>

<T> ::= <V> <T'>

<T'>::= * <V> <T'>
      | <empty>

<V> ::= x | y | z

x * y + z
```



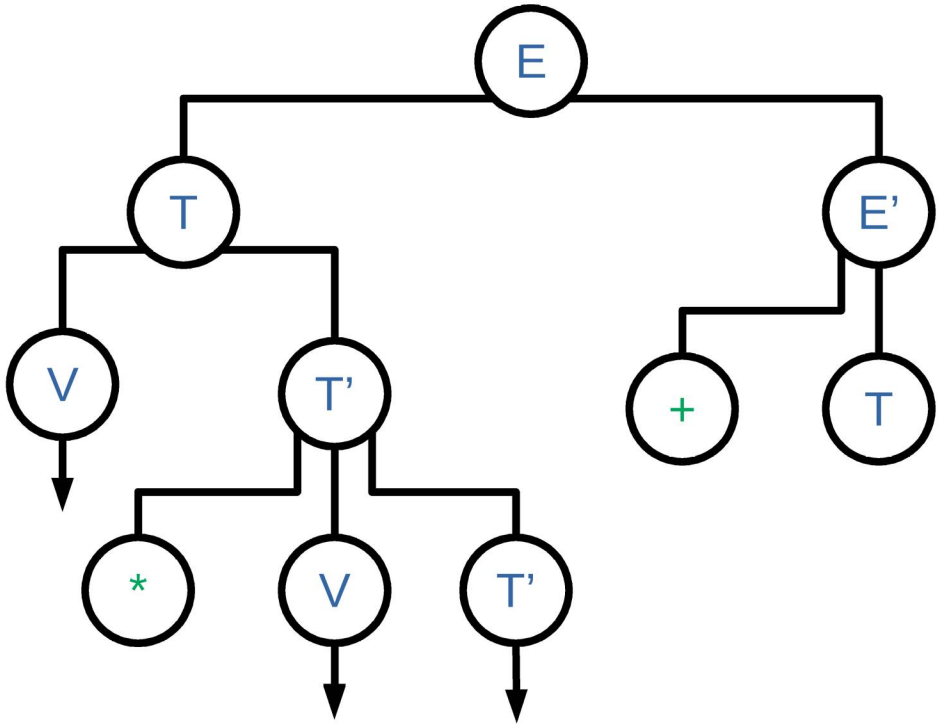
Utbyggbart


```
<E> ::= <T> <E'>

<E'>::= + <T> <E'>
      | <empty>

<T> ::= <V> <T'>

<T'>::= * <V> <T'>
      | <empty>

<V> ::= x | y | z

x * y + z
```



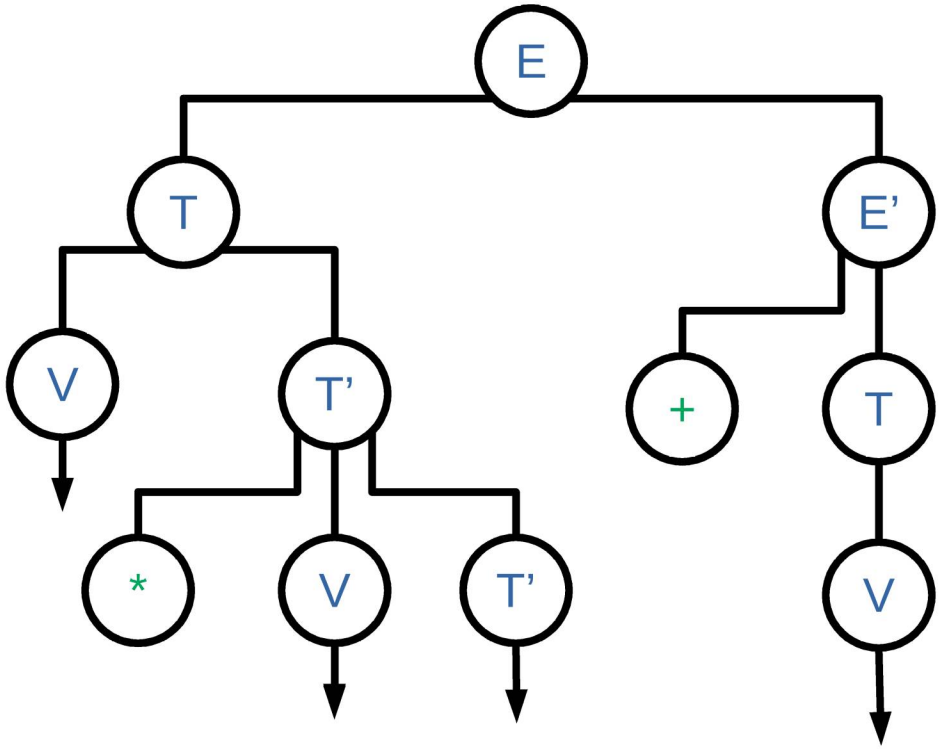
Utbyggbart


```
<E> ::= <T> <E'>

<E'>::= + <T> <E'>
      | <empty>

<T> ::= <V> <T'>

<T'>::= * <V> <T'>
      | <empty>

<V> ::= x | y | z

x * y + z
```



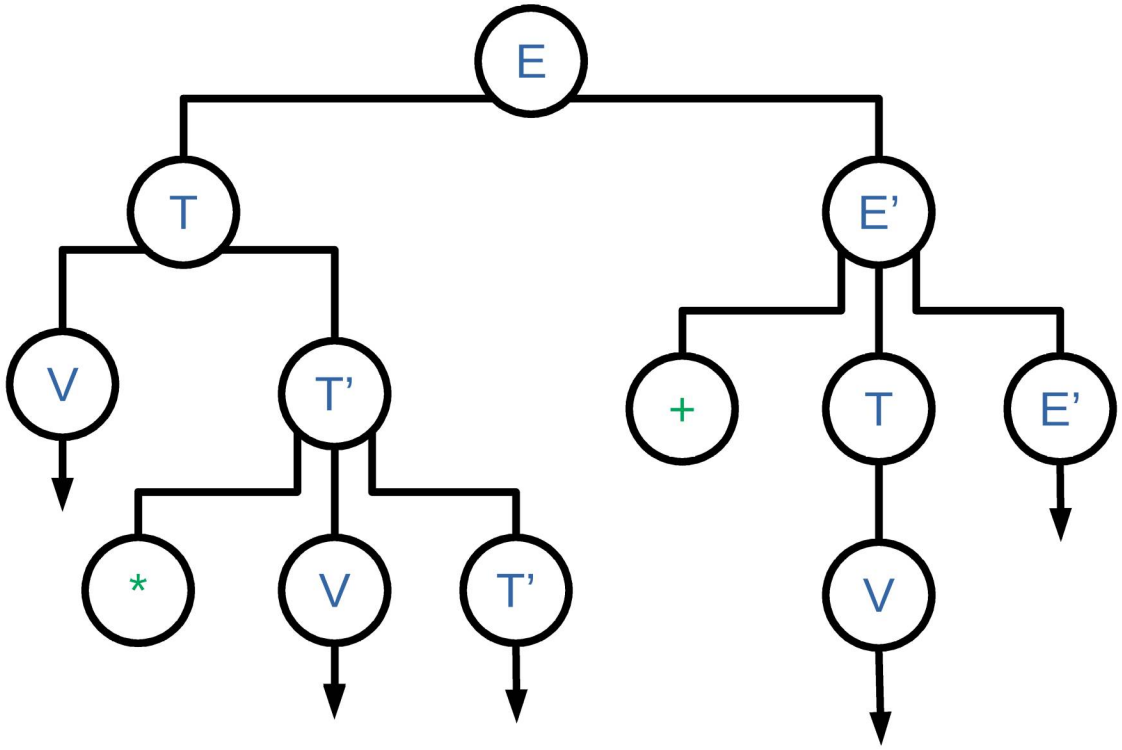
Utbyggbart


```
<E> ::= <T> <E'>

<E'>::= + <T> <E'>
      | <empty>

<T> ::= <V> <T'>

<T'>::= * <V> <T'>
      | <empty>

<V> ::= x | y | z

x * y + z
```



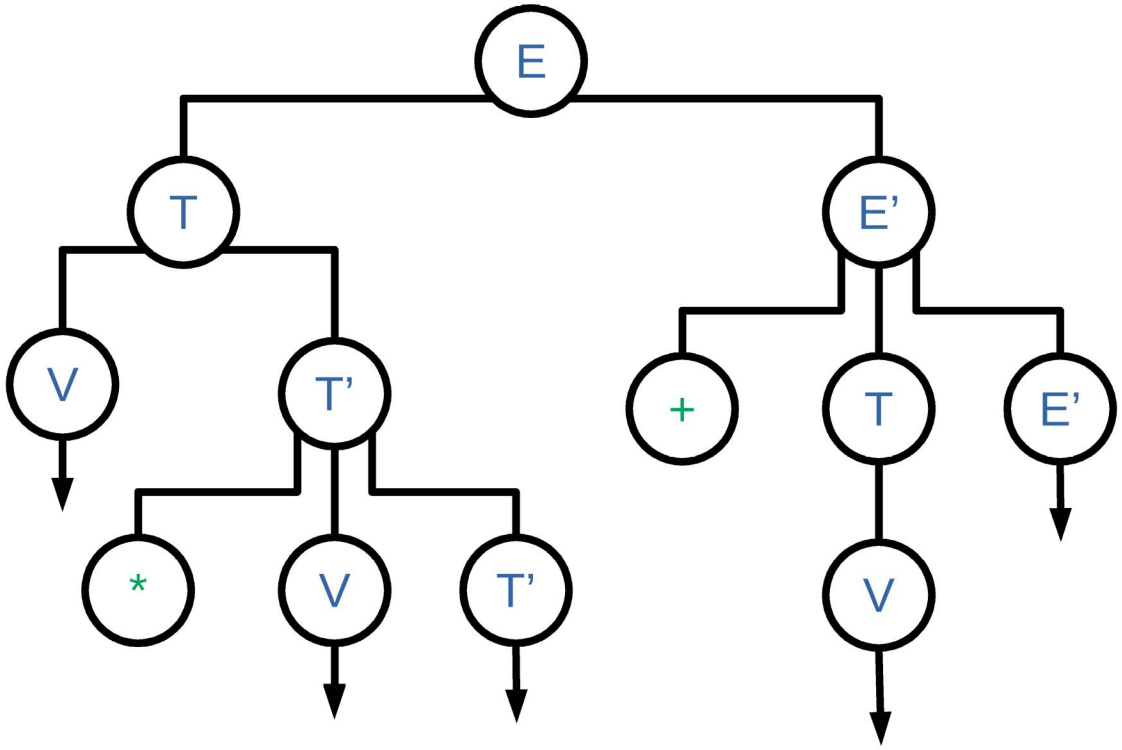
Utbyggbart


```
<E> ::= <T> <E'>

<E'>::= + <T> <E'>
      | <empty>

<T> ::= <V> <T'>

<T'>::= * <V> <T'>
      | <empty>

<V> ::= x | y | z

x * y + z
```



Utbyggbart


```
<E> ::= <T> <E'>

<E'>::= + <T> <E'>
      | <empty>

<T> ::= <V> <T'>

<T'>::= * <V> <T'>
      | <empty>

<V> ::= x | y | z

x * y + z
```



Insikt: Detta löser massor av problem med tvetydighet

Prioritet

LINKÖPINGS UNIVERSITET

Prioritet

- I vilken ordning ska saker utföras?

```
1+2*3
1*2+3
```

Prioritet

- I vilken ordning ska saker utföras?

```
1+2*3   ⟹  1+(2*3)
1*2+3
```

Prioritet

- I vilken ordning ska saker utföras?

```
1+2*3   ⟹  1+(2*3)
1*2+3   ⟹  (1*2)+3
```

Prioritet

- I vilken ordning ska saker utföras?

```
1+2*3   ⟹  1+(2*3)
1*2+3   ⟹  (1*2)+3
-1+3
--1+3
x<y  and  y>z
```

Prioritet

- I vilken ordning ska saker utföras?

```
1+2*3   ⟹  1+(2*3)
1*2+3   ⟹  (1*2)+3
-1+3    ⟹  (-1)+3
--1+3
x<y and y>z
```

Prioritet

- I vilken ordning ska saker utföras?

```
1+2*3    ⟹  1+(2*3)
1*2+3    ⟹  (1*2)+3
-1+3     ⟹  (-1)+3
--1+3    ⟹  (-(-1))+3
x<y  and  y>z
```

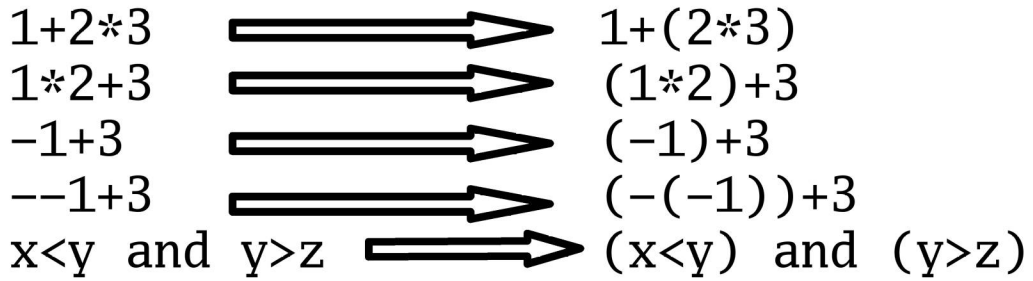
Prioritet

- I vilken ordning ska saker utföras?

```
1+2*3    ⟹  1+(2*3)
1*2+3    ⟹  (1*2)+3
-1+3     ⟹  (-1)+3
--1+3    ⟹  (-(-1))+3
x<y and y>z ⟹ (x<y) and (y>z)
```

Prioritet

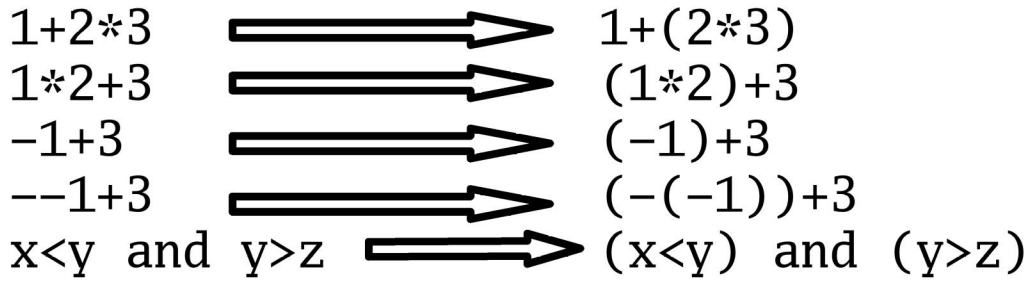
- I vilken ordning ska saker utföras?



- Att välja en bra ordning
- Att styra hur det fungerar

Prioritet

- I vilken ordning ska saker utföras?



- Att välja en bra ordning
- Att styra hur det fungerar

Insikt: Att styra ordningen har vi redan lärt oss, detta är samma sak
Längre ned betyder högre prioritet

Korrekt associativitet

associativitet

- Hur styr jag detta?
- Hur kontrollerar jag detta?
- Vänster eller höger?

associativitet

- Hur styr jag detta?
- Hur kontrollerar jag detta?
- Vänster eller höger?
- Vänster: Höger:

associativitet

- Hur styr jag detta?
- Hur kontrollerar jag detta?
- Vänster eller höger?
- Vänster: Höger:

```
1 + 2 + 3
```

```
1 + 2 + 3
```

associativitet

- Hur styr jag detta?
- Hur kontrollerar jag detta?
- Vänster eller höger?
- Vänster:

```
1 + 2 + 3
(1 + 2) + 3
```

Höger:

```
1 + 2 + 3
1 + (2 + 3)
```

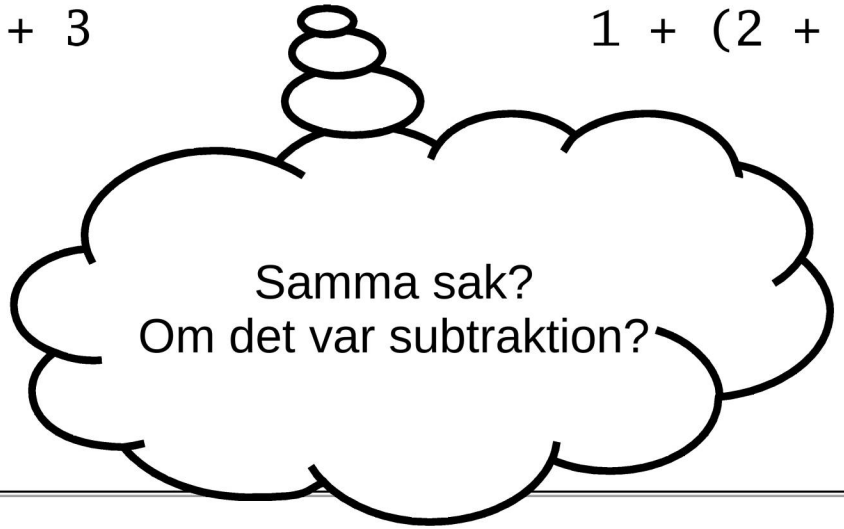
associativitet

- Hur styr jag detta?
- Hur kontrollerar jag detta?
- Vänster eller höger?
- Vänster:

```
1 + 2 + 3
(1 + 2) + 3
```

- Höger:

```
1 + 2 + 3
1 + (2 + 3)
```



associativitet

Exempel som bör vara höger associativa

−1+3	(−1)+3
−−1+3	(−(−1))+3
1+1^2	1+(1^2)
1+1^2^3	1+1^(2^3)

associativitet != Prioritet

- Skillnad mellan associativitet och prioritet
-
  1+2*3
  1*2+3
  1–2–3

Höger associativitet


```
<UE> ::= + <UE>
       | - <UE>
       | <V>

<V> ::= x | y | z

--x
```

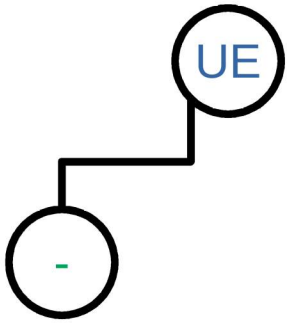
Höger associativitet


```
<UE> ::= + <UE>
       | - <UE>
       | <V>

<V> ::= x | y | z

--x
```

Höger associativitet

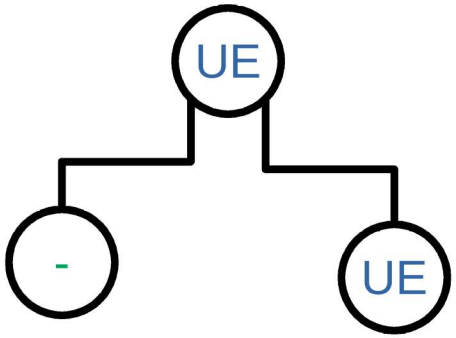

```
<UE> ::= + <UE>
       | - <UE>
       | <V>

<V> ::= x | y | z
```

```
--x
```



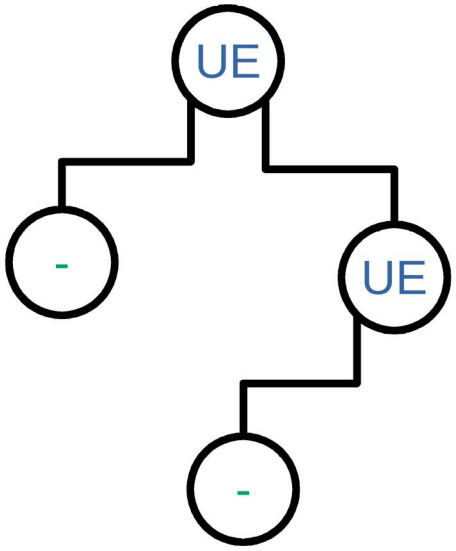
Höger associativitet


```
<UE> ::= + <UE>
       | - <UE>
       | <V>

<V> ::= x | y | z

--x
```



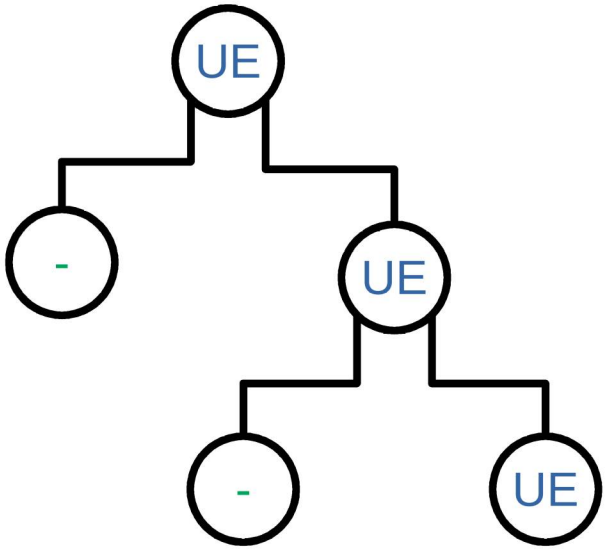
Höger associativitet


```
<UE> ::= + <UE>
       | - <UE>
       | <V>

<V> ::= x | y | z

--x
```



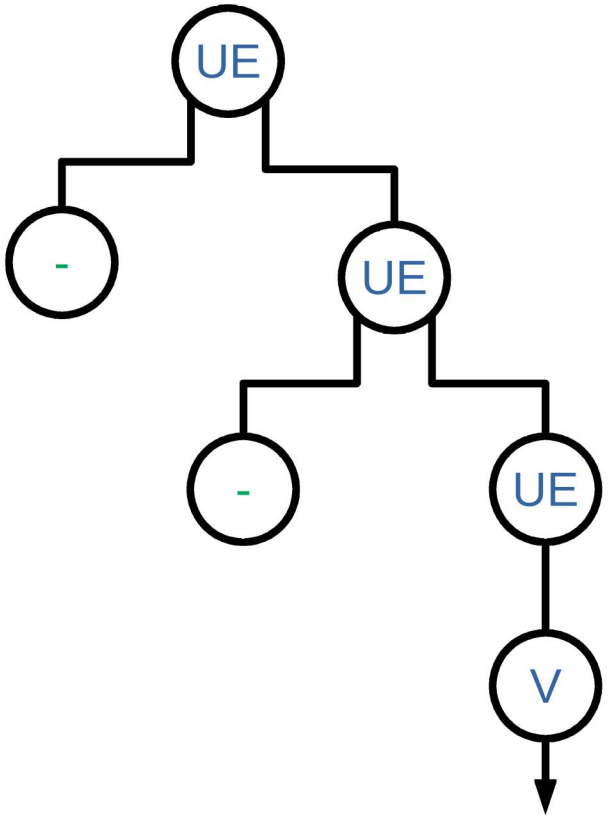
Höger associativitet


```
<UE> ::= + <UE>
       | - <UE>
       | <V>

<V> ::= x | y | z

--x
```



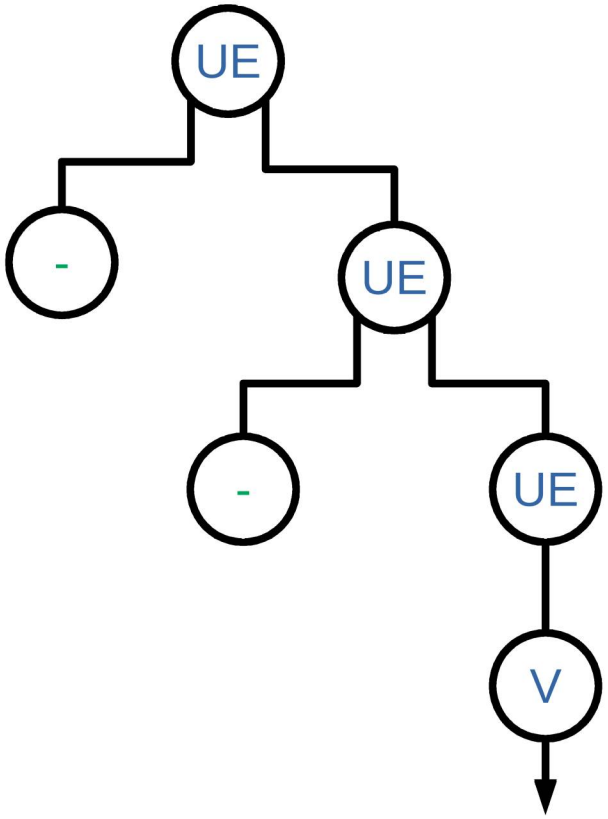
Höger associativitet


```
<UE> ::= + <UE>
       | - <UE>
       | <V>

<V> ::= x | y | z

--x
```



Höger associativitet


```
<UE> ::= + <UE>
       | - <UE>
       | <V>

<V> ::= x | y | z

--x
```



Insikt: Detta verkar fungera… Och är det enda rimliga sättet att skriva det på

Höger associativitet – annat exempel


```
<EX> ::=<EX> ^ <V>
       | <V>

<V> ::= x | y | z
```

x^y^z

Höger associativitet – annat exempel

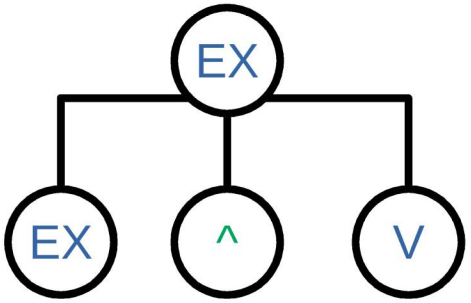

```
<EX> ::=<EX> ^ <V>
       | <V>

<V> ::= x | y | z
```

x^y^z

Höger associativitet – annat exempel


```
<EX> ::=<EX> ^ <V>
       | <V>

<V> ::= x | y | z
```

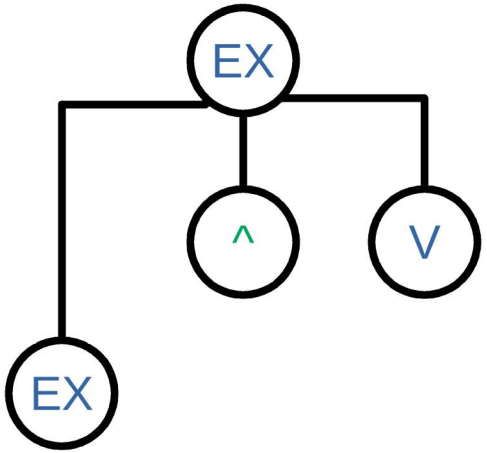
x^y^z



Höger associativitet – annat exempel


```
<EX> ::=<EX> ^ <V>
       | <V>

<V> ::= x | y | z
```

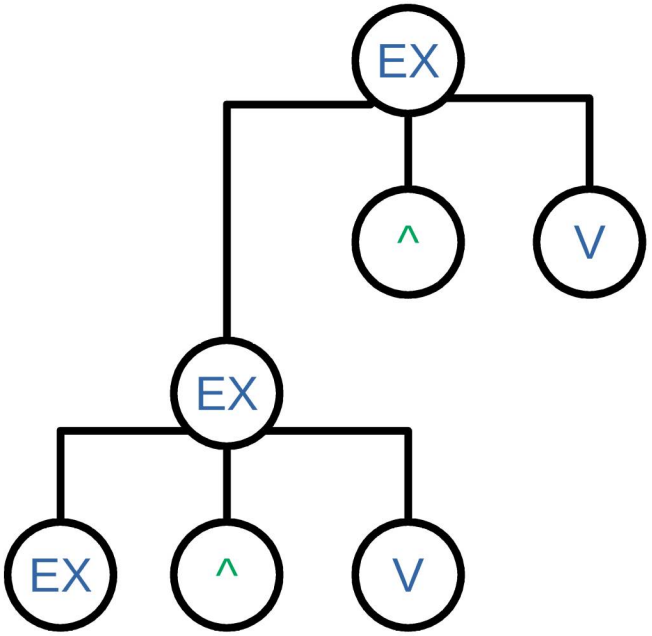
x^y^z



Höger associativitet – annat exempel


```
<EX> ::=<EX> ^ <V>
       | <V>

<V> ::= x | y | z
```

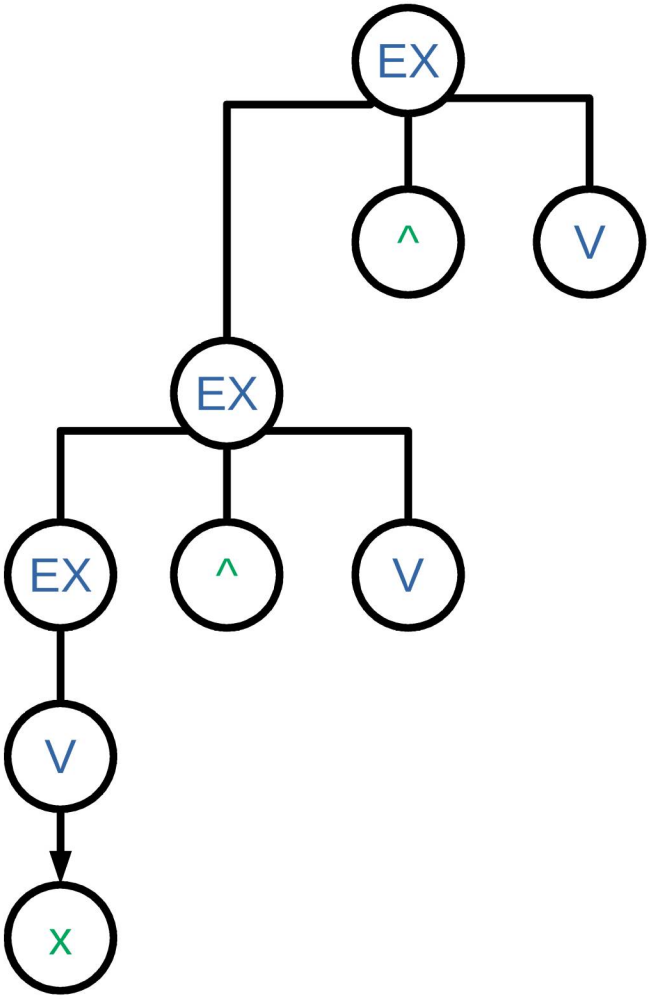
x^y^z



Höger associativitet – annat exempel


```
<EX> ::=<EX> ^ <V>
       | <V>

<V> ::= x | y | z
```

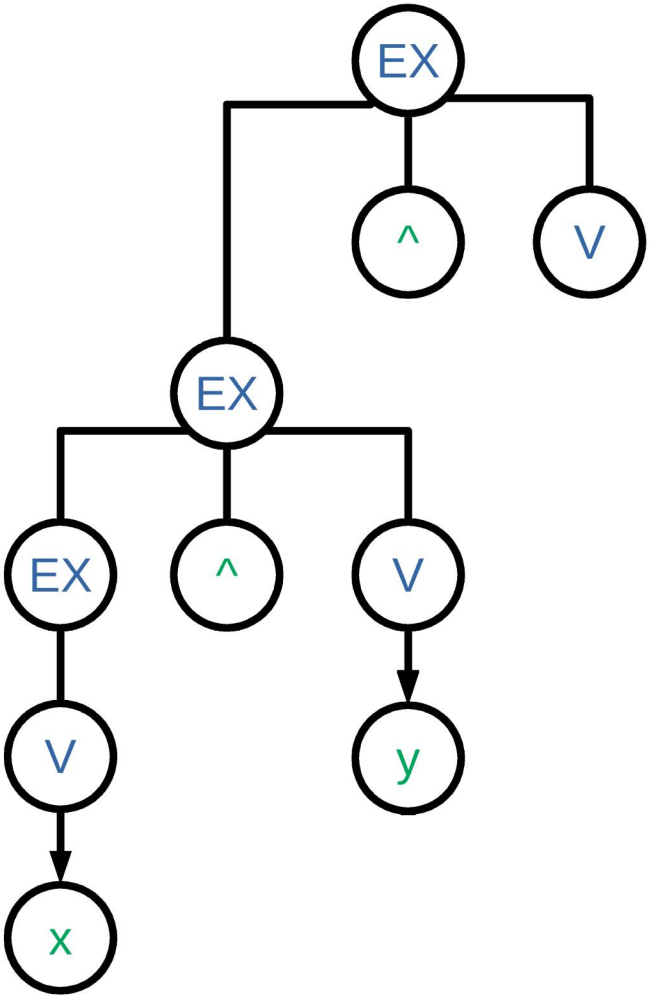
x^y^z



Höger associativitet – annat exempel


```
<EX> ::=<EX> ^ <V>
       | <V>

<V> ::= x | y | z
```

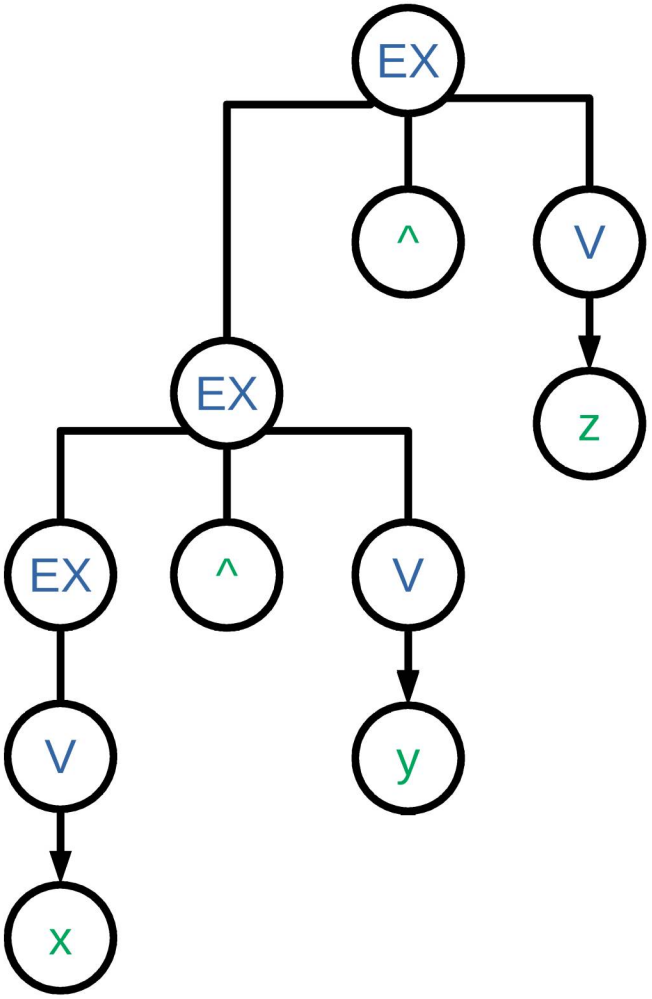
x^y^z



Höger associativitet – annat exempel


```
<EX> ::=<EX> ^ <V>
       | <V>

<V> ::= x | y | z
```

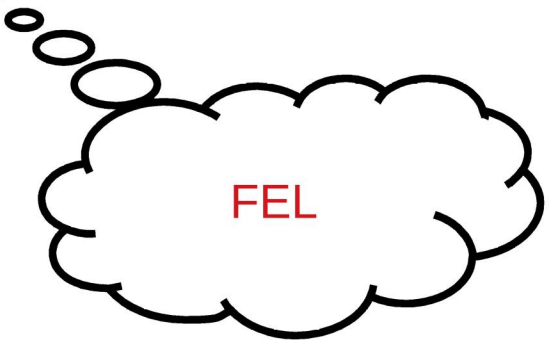
x^y^z

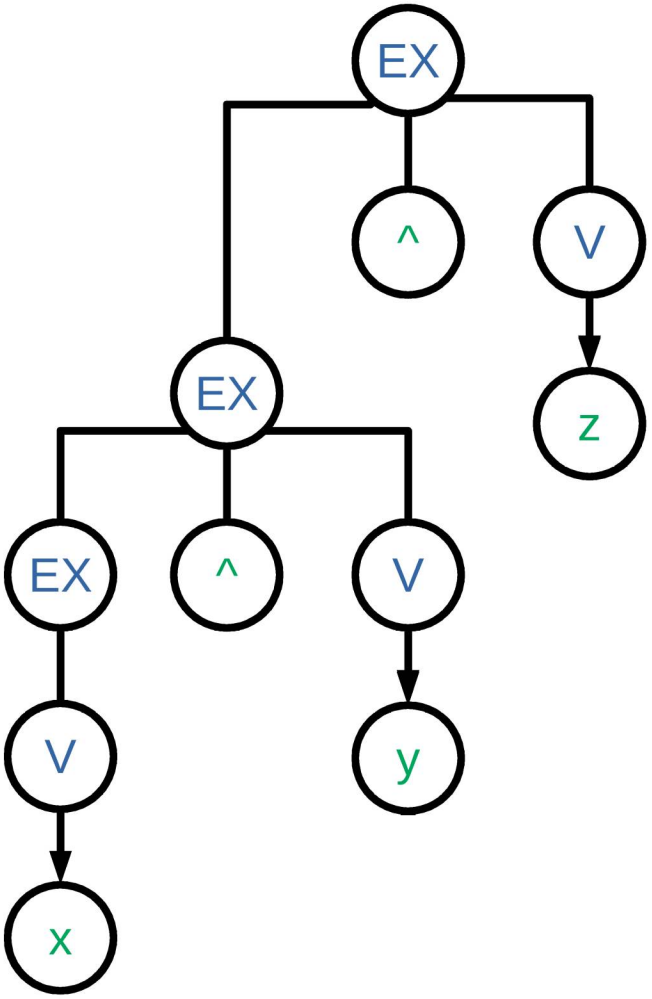


~~Höger~~ associativitet – annat exempel


```
<EX> ::=<EX> ^ <V>
       | <V>

<V> ::= x | y | z
```

(x^y)^z





Höger associativitet – annat exempel


```
<EX> ::=<EX> ^ <V>
       | <V>

<V> ::= x | y | z
```

x^y^z

Höger associativitet – annat exempel


```
<EX> ::=<V> ^ <EX>
       | <V>

<V> ::= x | y | z
```

x^y^z

Höger associativitet – annat exempel

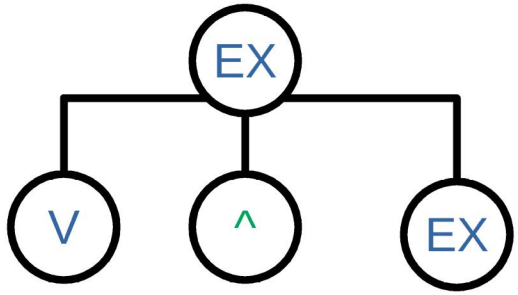

```
<EX> ::=<V> ^ <EX>
       | <V>

<V> ::= x | y | z
```

x^y^z

Höger associativitet – annat exempel


```
<EX> ::=<V> ^ <EX>
       | <V>

<V> ::= x | y | z
```

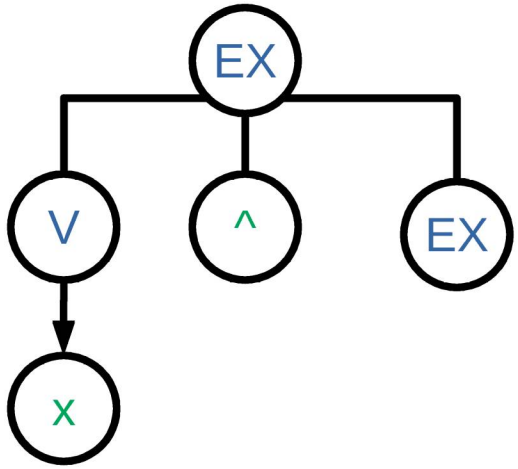
x^y^z



Höger associativitet – annat exempel


```
<EX> ::=<V> ^ <EX>
       | <V>

<V> ::= x | y | z
```

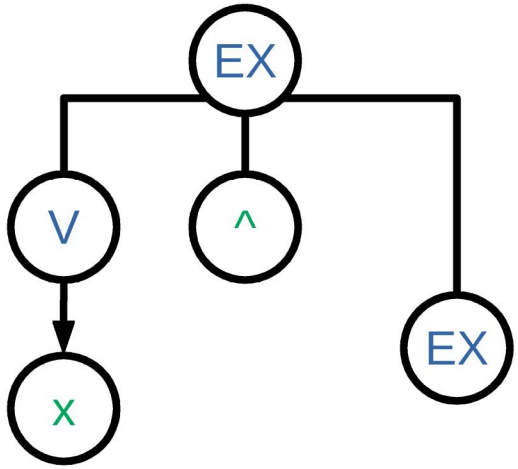
x^y^z



Höger associativitet – annat exempel


```
<EX> ::=<V> ^ <EX>
       | <V>

<V> ::= x | y | z
```

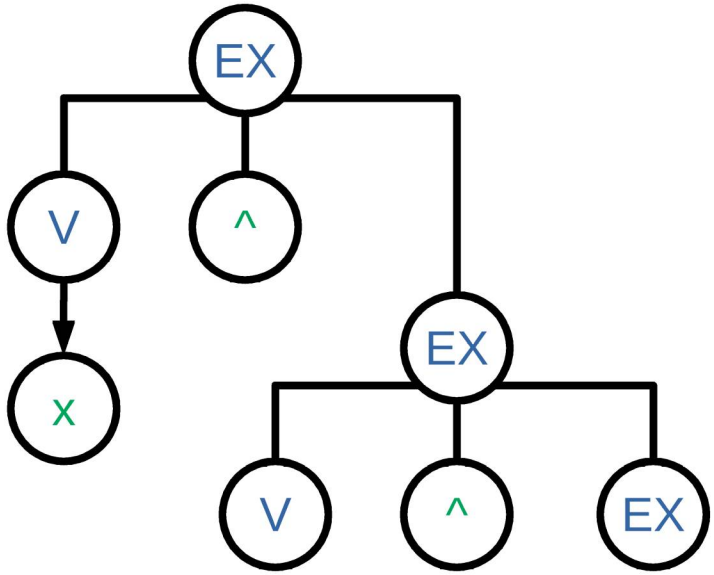
x^y^z



Höger associativitet – annat exempel


```
<EX>  ::=<V>  ^  <EX>
        |  <V>

<V>  ::=  x  |  y  |  z
```

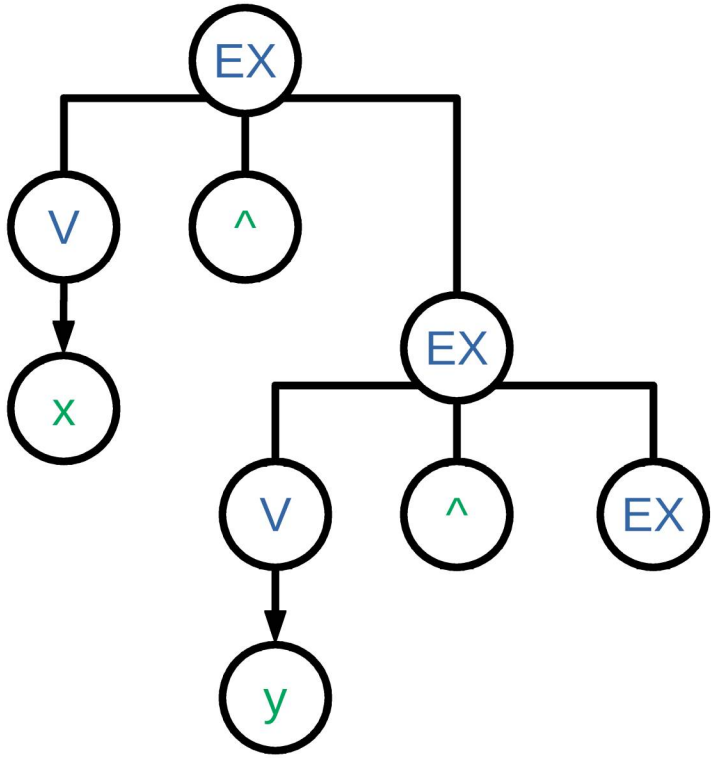
x^y^z



Höger associativitet – annat exempel


```
<EX> ::=<V> ^ <EX>
       | <V>

<V> ::= x | y | z
```

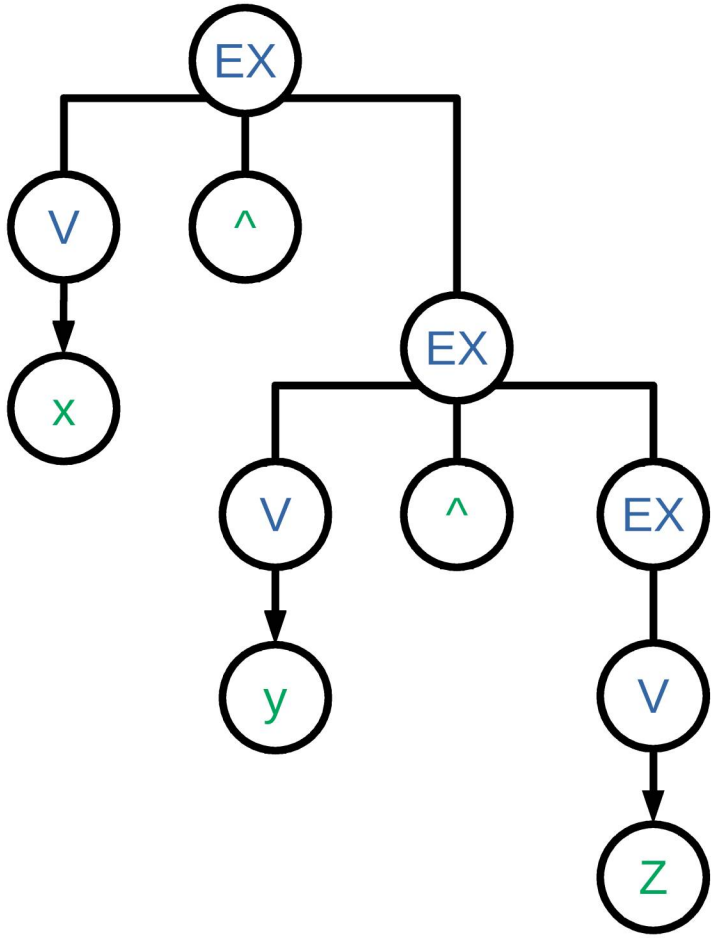
x^y^z



Höger associativitet – annat exempel


```
<EX> ::=<V> ^ <EX>
       | <V>

<V> ::= x | y | z
```

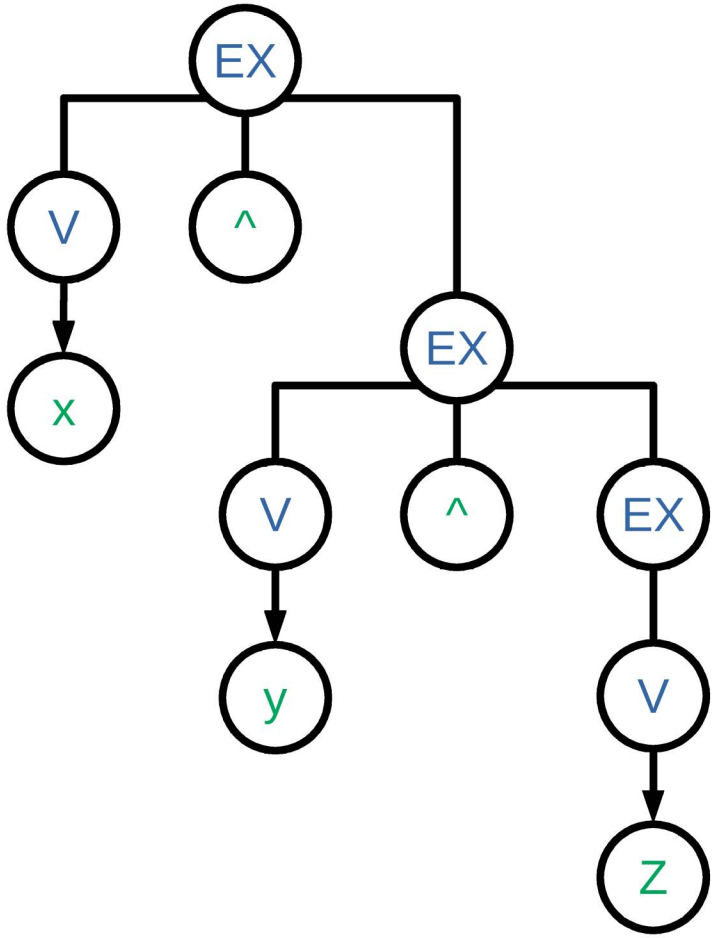
x^y^z



Höger associativitet – annat exempel


```
<EX> ::=<V> ^ <EX>
       | <V>

<V> ::= x | y | z
```

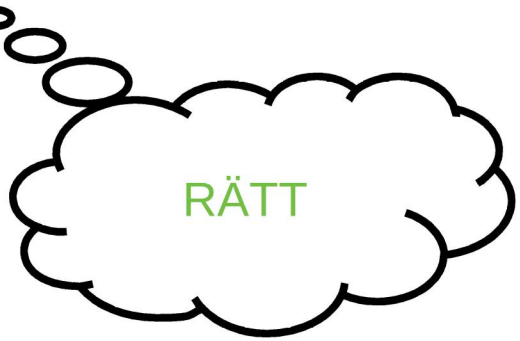
x^(y^z)

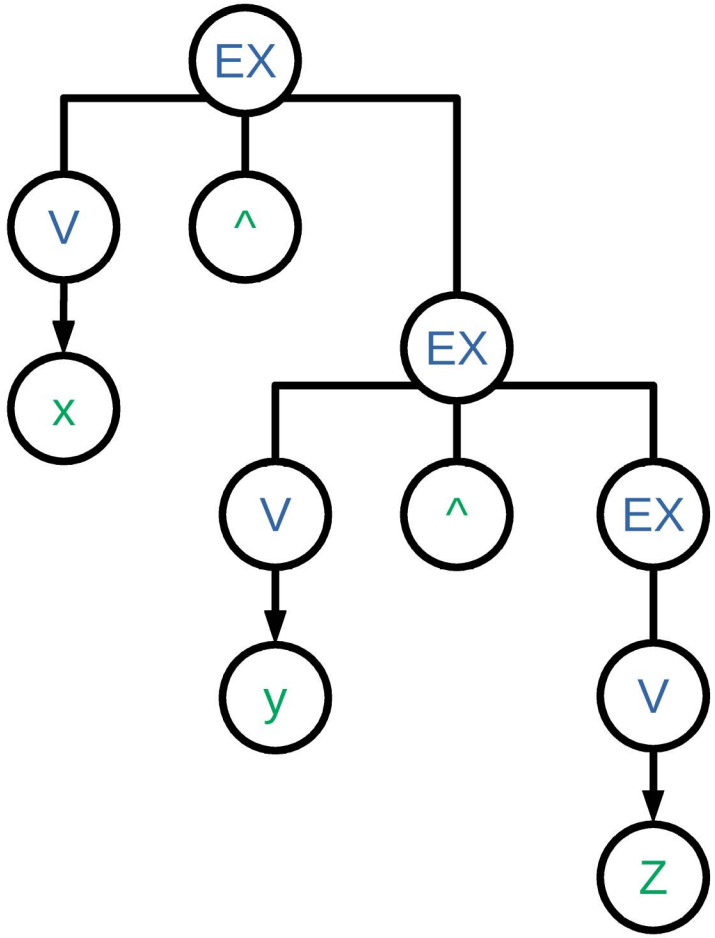


Höger associativitet – annat exempel


```
<EX> ::=<V> ^ <EX>
       | <V>

<V> ::= x | y | z
```

x^(y^z)





associativitet - avslutning

- Bestäm hur det är rimligt att det ska fungera
- Styr associativiteten med grammatiken
- Vänster så är den högsta (grundaste) regeln till vänster i regeln
- Höger så är den högsta (grundaste) regeln till höger i regeln

```
<EX> ::=<EX> ^ <V>
       | <V>

<V> ::= x | y | z

(x^y)^z
```

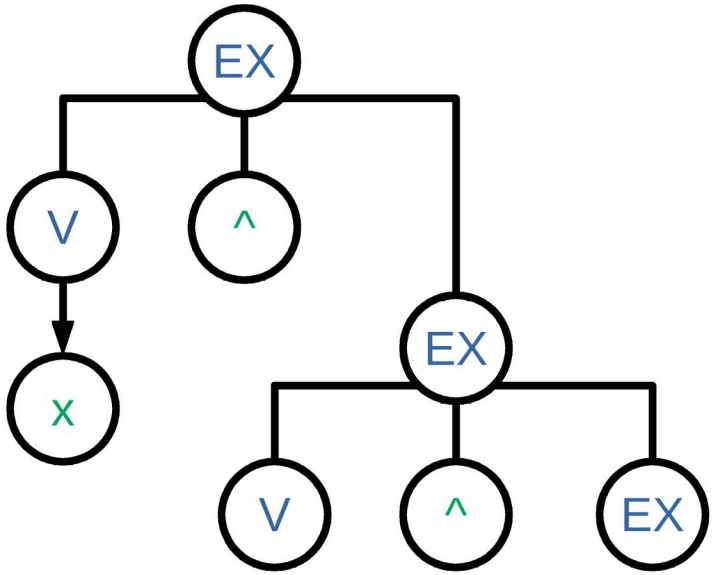
```
<EX> ::=<V> ^ <EX>
       | <V>

<V> ::= x | y | z

x^(y^z)
```

Några tankar om Bottom-Up

LINKÖPINGS UNIVERSITET

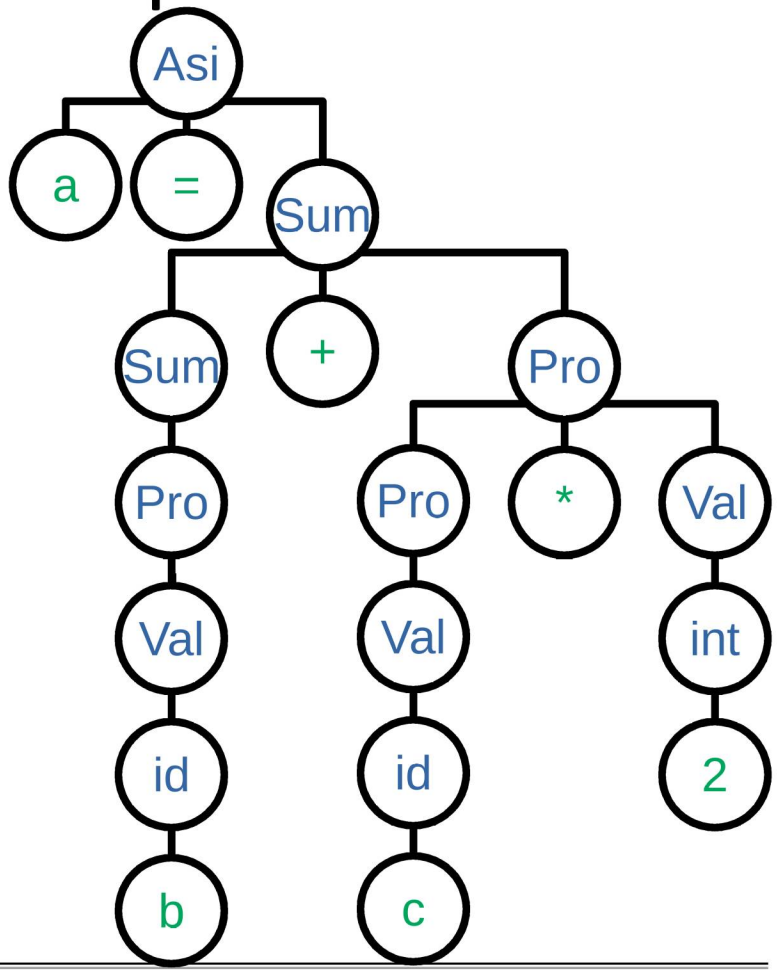
Bottom-Up

- Annat sätt att parsa på.
- Går minsta → största när det kommer till delar i språket.
- Potentiellt kraftfullt.



Bottom-Up | Hur läser Top-Down?

```
Assign ::= id = Sums

Sums ::= Sums + Products
       | Products

Products ::= Products*Value
           | Value

Value ::= int
        | id

a = b + c * 2
```



Exempel från: https://en.wikipedia.org/wiki/Shift-reduce_parser

Bottom-Up | Hur läser Bottom-Up?

Shift-reduce parser

```
a = b + c * 2
```


Exempel från: https://en.wikipedia.org/wiki/Shift-reduce_parser

Bottom-Up | Hur läser Bottom-Up?

Shift-reduce parser

```
a = b + c * 2
```

SHIFT

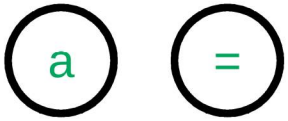
Exempel från: https://en.wikipedia.org/wiki/Shift-reduce_parser

Bottom-Up | Hur läser Bottom-Up?

Shift-reduce parser

```
a = b + c * 2
```

SHIFT



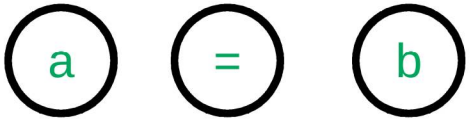
Exempel från: https://en.wikipedia.org/wiki/Shift-reduce_parser

Bottom-Up | Hur läser Bottom-Up?

Shift-reduce parser

```
a = b + c * 2
```

SHIFT



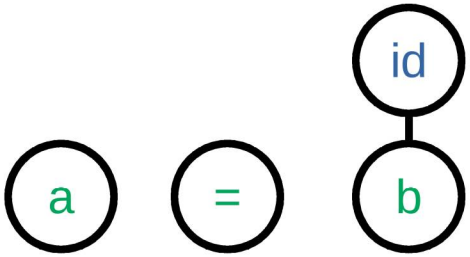
Exempel från: https://en.wikipedia.org/wiki/Shift-reduce_parser

Bottom-Up | Hur läser Bottom-Up?

Shift-reduce parser

a = b + c * 2

REDUCE



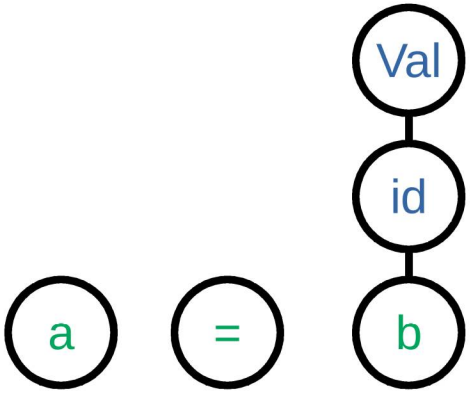
Exempel från: https://en.wikipedia.org/wiki/Shift-reduce_parser

Bottom-Up | Hur läser Bottom-Up?

Shift-reduce parser

a = b + c * 2

REDUCE



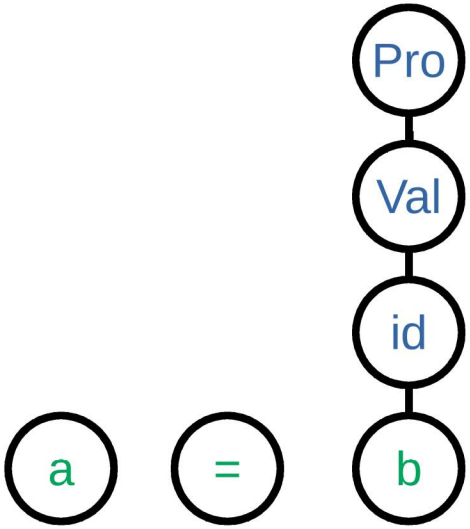
Exempel från: https://en.wikipedia.org/wiki/Shift-reduce_parser

Bottom-Up | Hur läser Bottom-Up?

Shift-reduce parser

a = b + c * 2

REDUCE



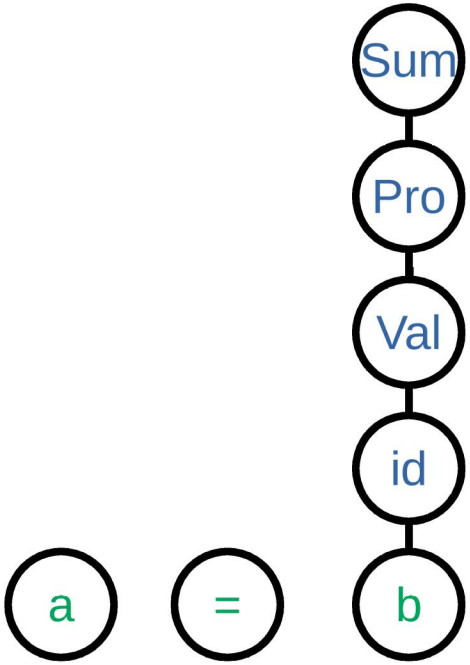
Exempel från: https://en.wikipedia.org/wiki/Shift-reduce_parser

Bottom-Up | Hur läser Bottom-Up?

Shift-reduce parser

```
a = b + c * 2
```

REDUCE



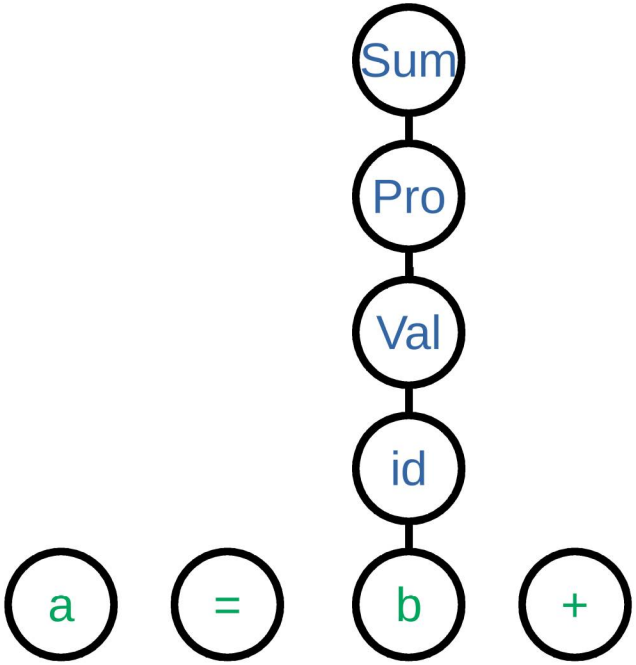
Exempel från: https://en.wikipedia.org/wiki/Shift-reduce_parser

Bottom-Up | Hur läser Bottom-Up?

Shift-reduce parser

```
a = b + c * 2
```

SHIFT



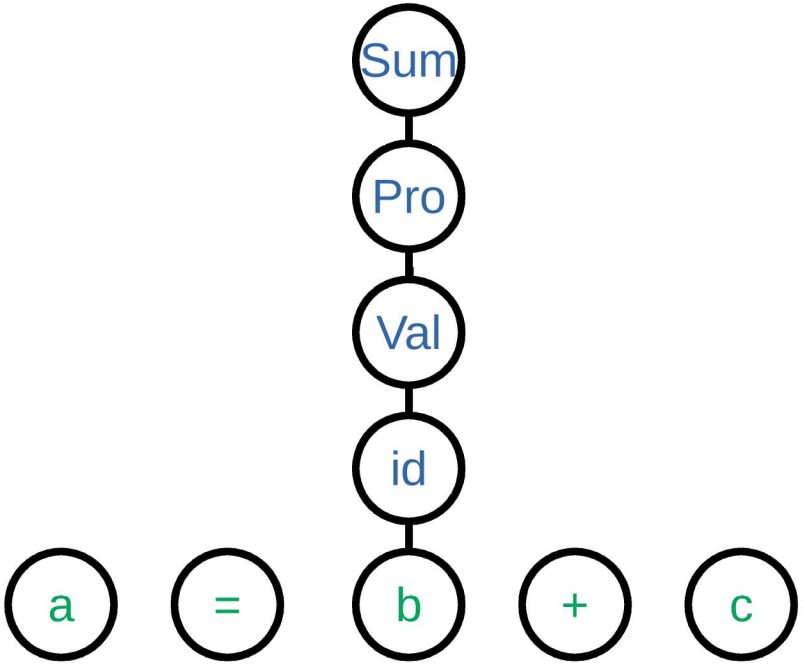
Exempel från: https://en.wikipedia.org/wiki/Shift-reduce_parser

Bottom-Up | Hur läser Bottom-Up?

Shift-reduce parser

```
a = b + c * 2
```

SHIFT



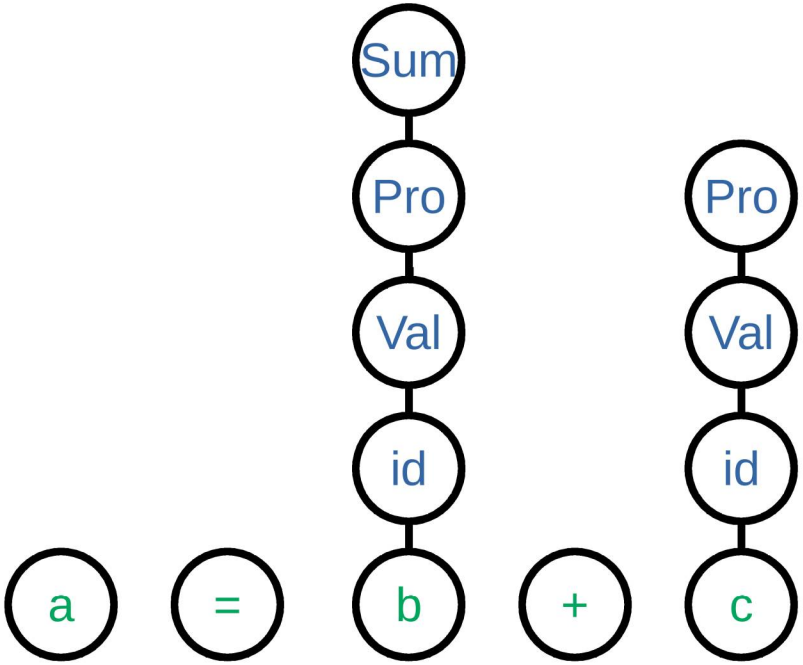
Exempel från: https://en.wikipedia.org/wiki/Shift-reduce_parser

Bottom-Up | Hur läser Bottom-Up?

Shift-reduce parser

a = b + c * 2

REDUCE



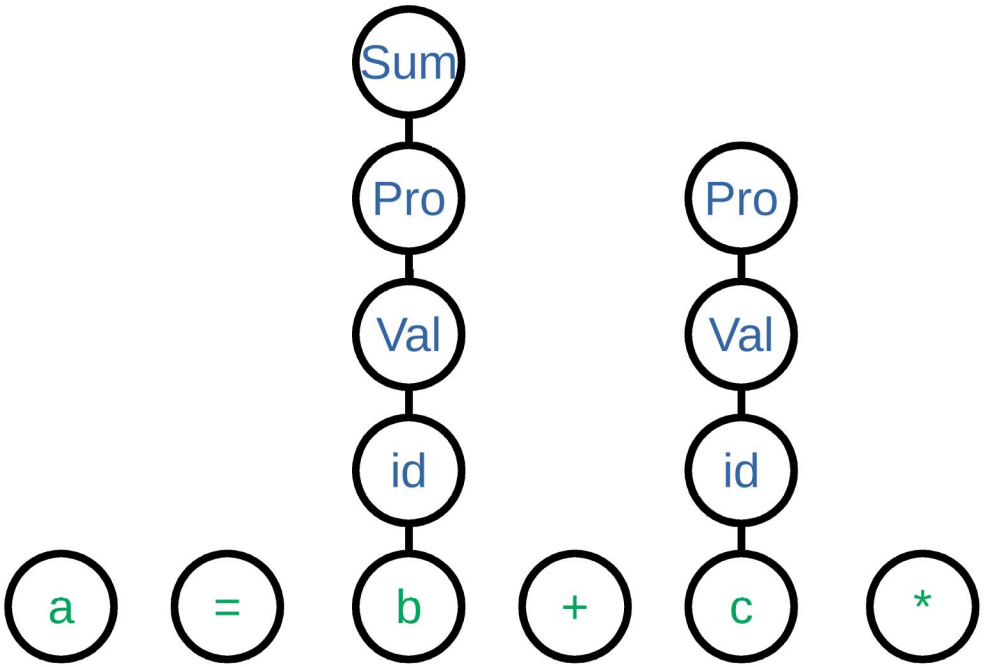
Exempel från: https://en.wikipedia.org/wiki/Shift-reduce_parser

Bottom-Up | Hur läser Bottom-Up?

Shift-reduce parser

a = b + c * 2

SHIFT



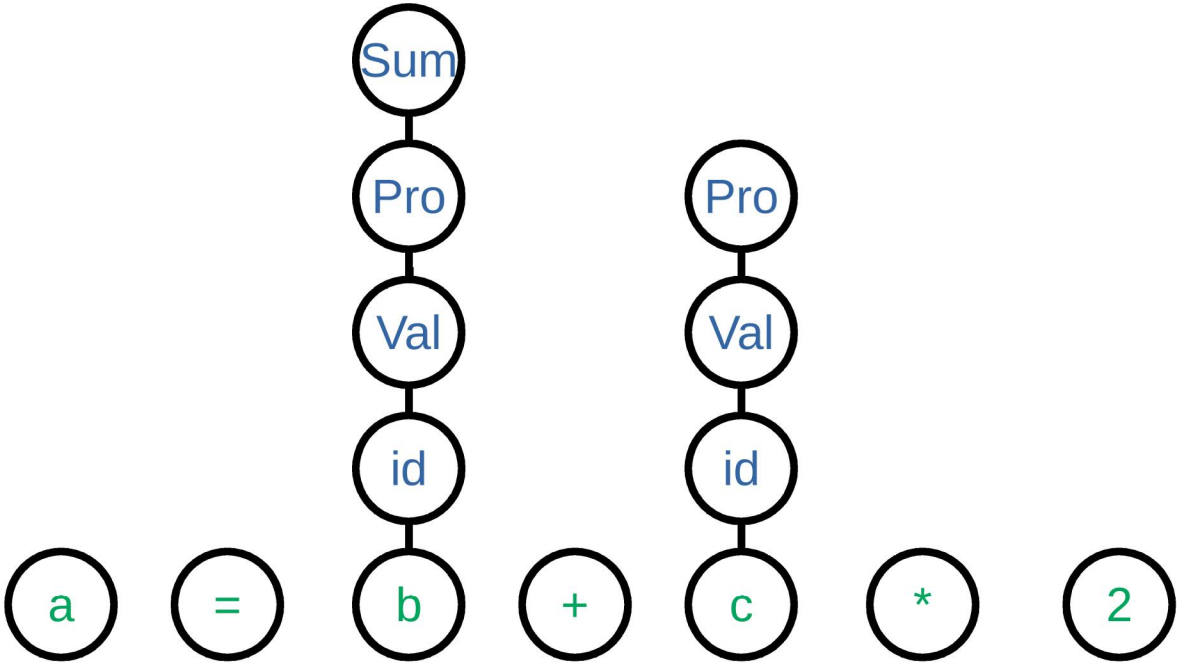
Exempel från: https://en.wikipedia.org/wiki/Shift-reduce_parser

Bottom-Up | Hur läser Bottom-Up?

Shift-reduce parser

a = b + c * 2

SHIFT



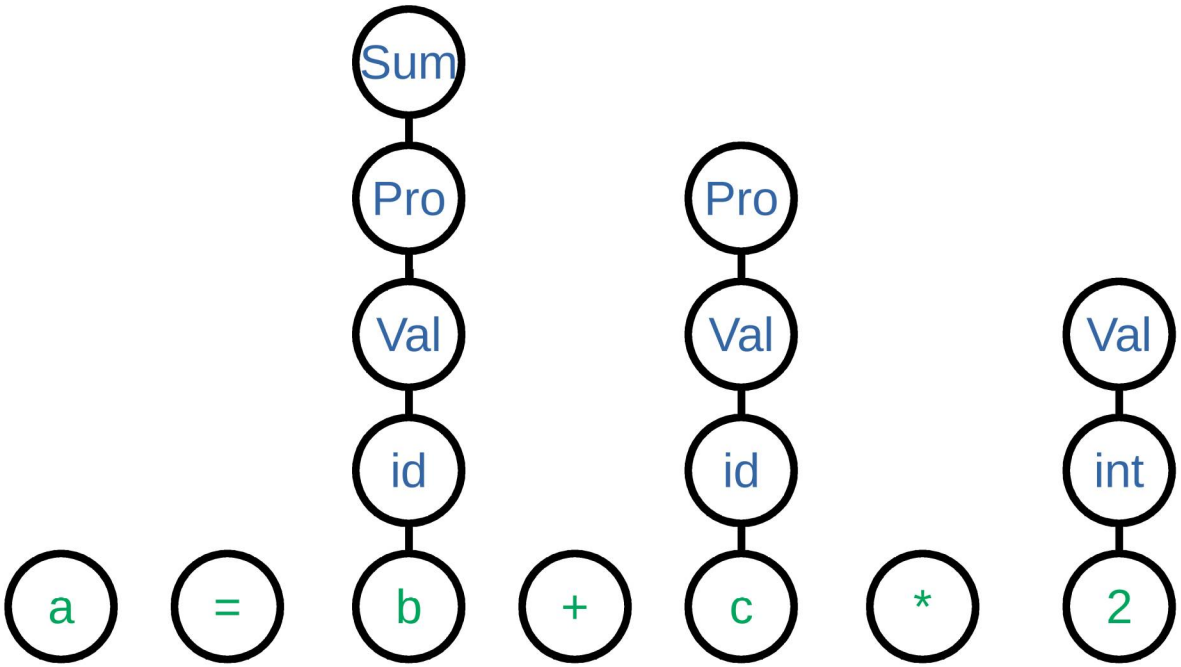
Exempel från: https://en.wikipedia.org/wiki/Shift-reduce_parser

Bottom-Up | Hur läser Bottom-Up?

Shift-reduce parser

a = b + c * 2

REDUCE



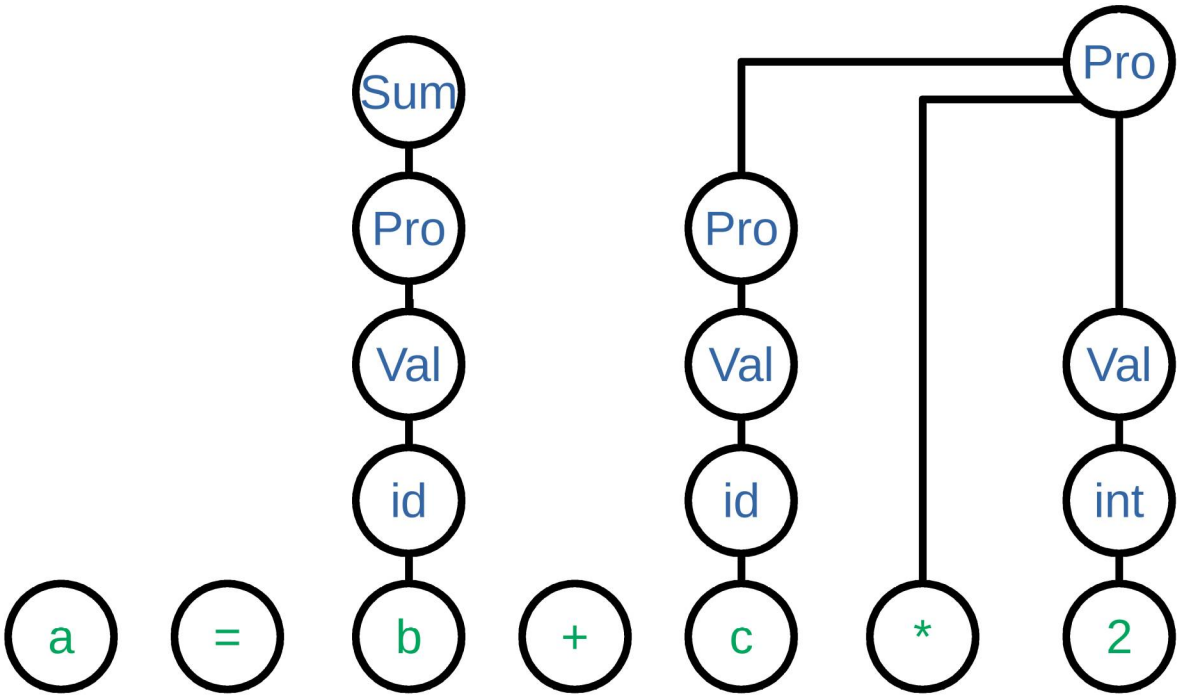
Exempel från: https://en.wikipedia.org/wiki/Shift-reduce_parser

Bottom-Up | Hur läser Bottom-Up?

Shift-reduce parser

a = b + c * 2

REDUCE



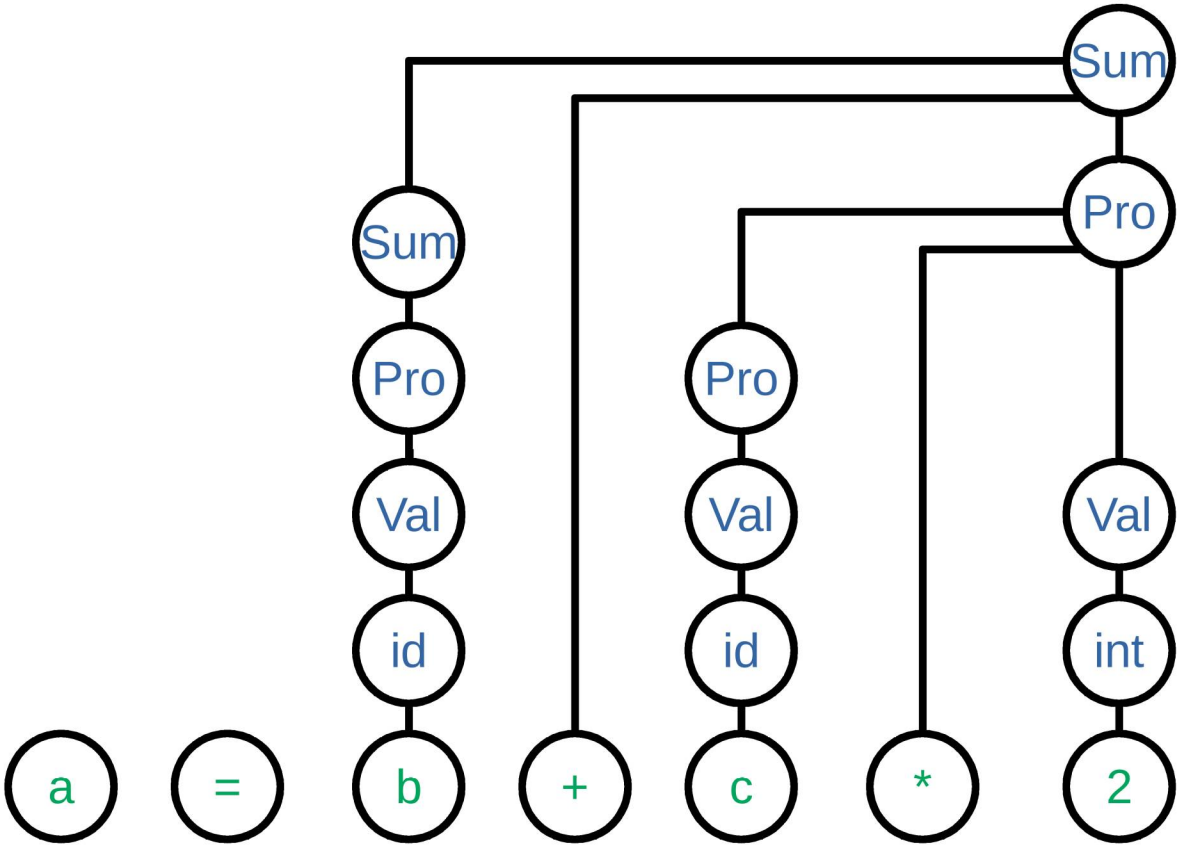
Exempel från: https://en.wikipedia.org/wiki/Shift-reduce_parser

Bottom-Up | Hur läser Bottom-Up?

Shift-reduce parser

a = b + c * 2

REDUCE



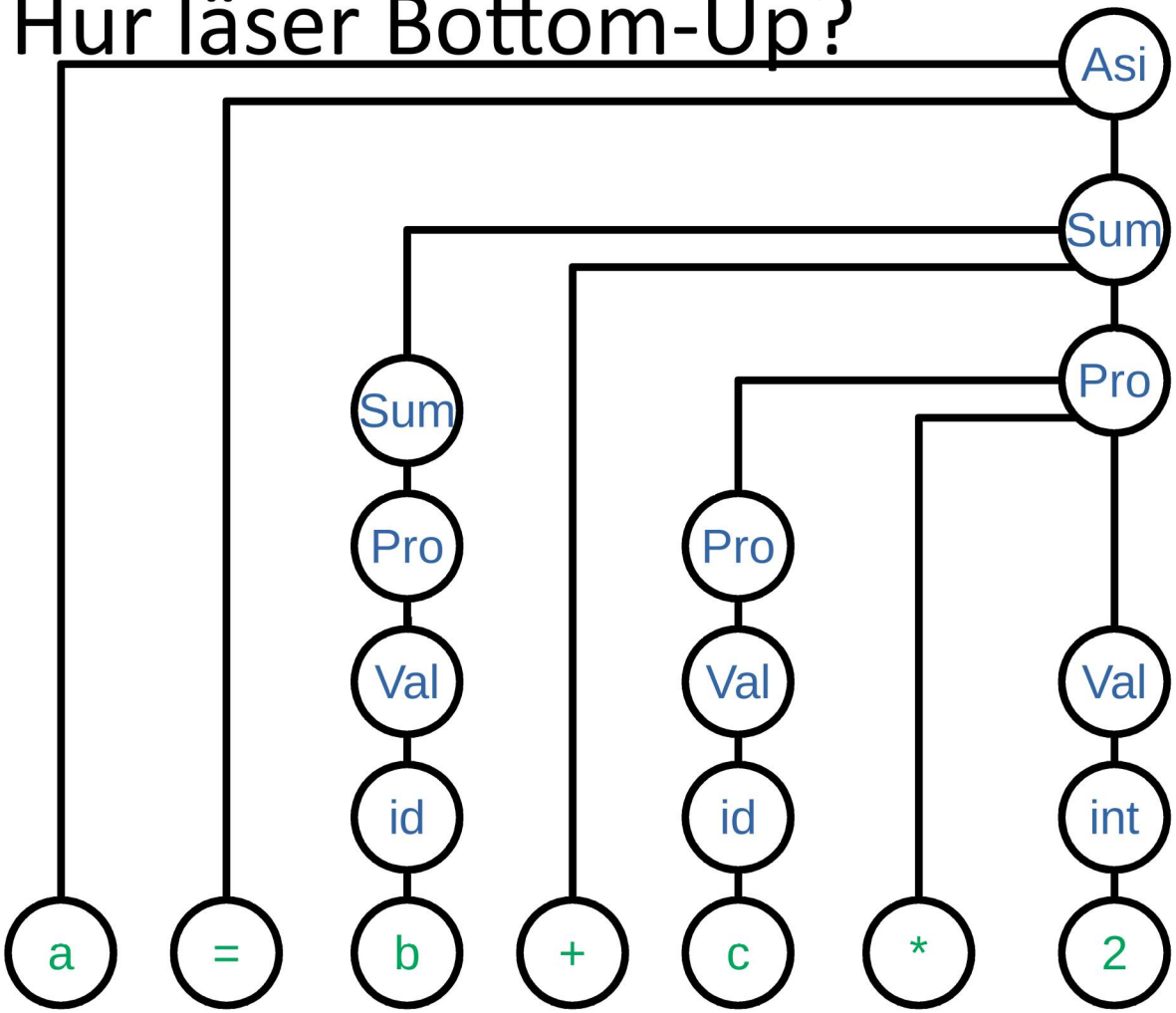
Exempel från: https://en.wikipedia.org/wiki/Shift-reduce_parser

Bottom-Up | Hur läser Bottom-Up?

Shift-reduce parser

a = b + c * 2

REDUCE



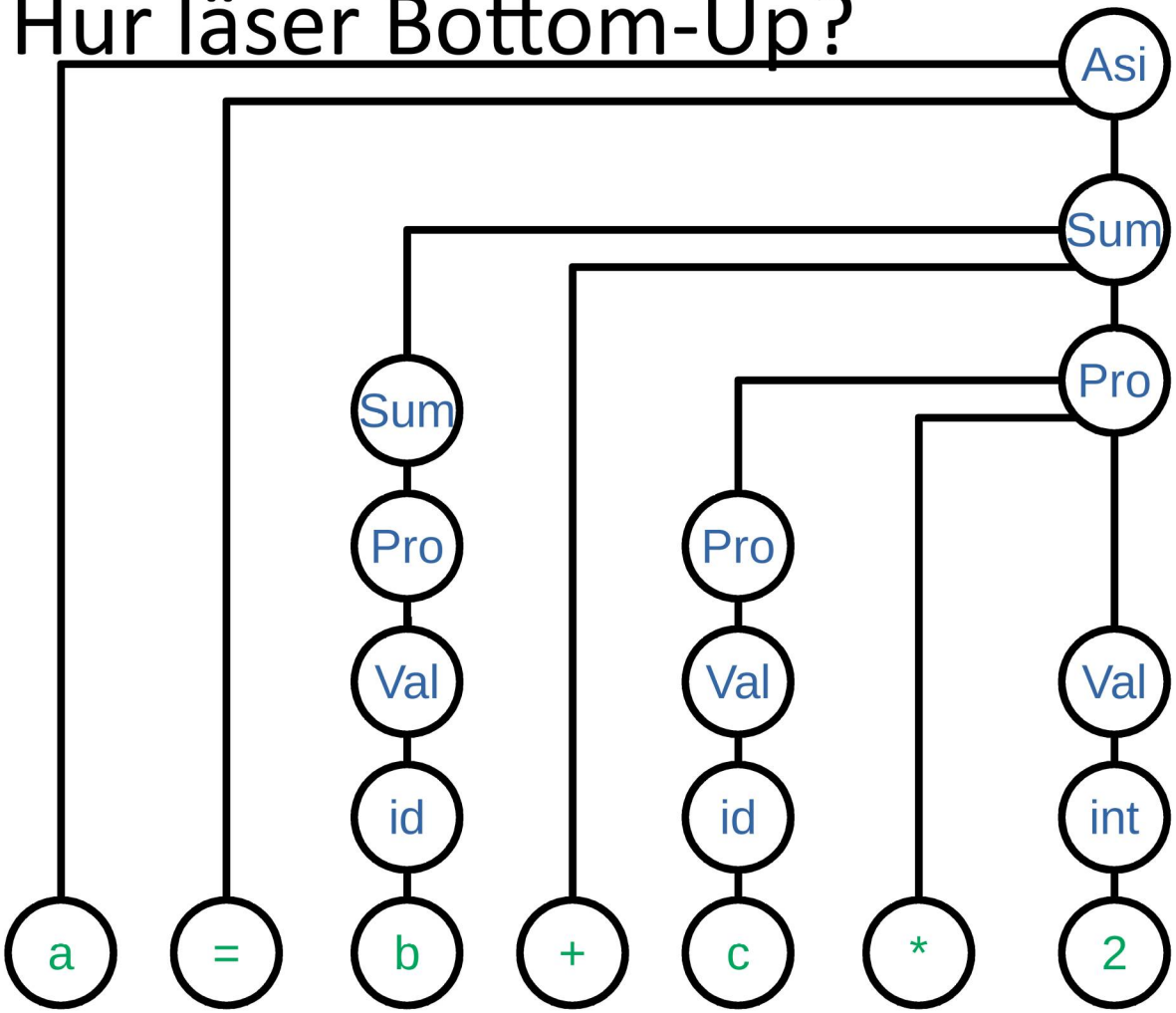
Exempel från: https://en.wikipedia.org/wiki/Shift-reduce_parser

Bottom-Up | Hur läser Bottom-Up?

Shift-reduce parser

a = b + c * 2

DONE

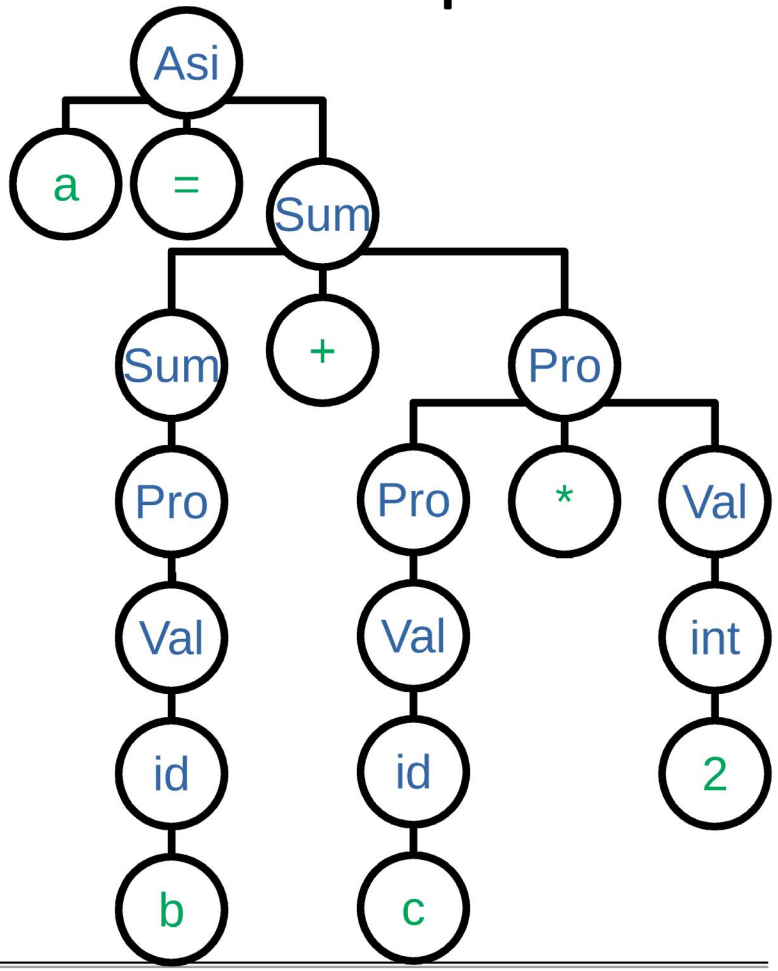


Exempel från: https://en.wikipedia.org/wiki/Shift-reduce_parser

Bottom-Up | Hur läser Bottom-Up?

Shift-reduce parser

```
a = b + c * 2
```



Exempel från: https://en.wikipedia.org/wiki/Shift-reduce_parser

Övningar

LINKÖPINGS UNIVERSITET

Booleans

- Man vill ofta kunna skriva något liknande if-satser, or, and, etc i alla språk
- Exempelvis kan man då behöva parsa strängen:
  a+b > 3 and x > y+z

<C> ::= <AE> Comp <AE>
<Comp> ::= < | <= | >= | > | != | ==

<BE> ::= <BE> or <BT> | <BT>
<BT> ::= <BT> and <BF> | <BF>
<BF> ::= (<BE>) | <V> | true | false

Övning: Hur kopplar man ihop <C> med <BE>/<BT>/<BF> ?

Alltså lägg till eller ändra i de existerande reglerna så att strängen ovan går att köra.

Du kan anta att <AE> fungerar helt

Syntaxträd

1) *Rita syntaxträd för* `a + b > 3 and ( x > y or x==2 )`
2) *med unärt minus* `(-1 - x)`
3) *med negation (not)* `x > y or not(x==2)`
4) *med negation (not)* `x > y or not(not(x==2))`
5) med exponent `x^y^z == x^(y^z)`

Palindrom - strängar

- Hur kan man matcha palindrom med grammatik?
- aabaa abcba abba aaaa
- Fungerar följande? Varför eller varför inte?

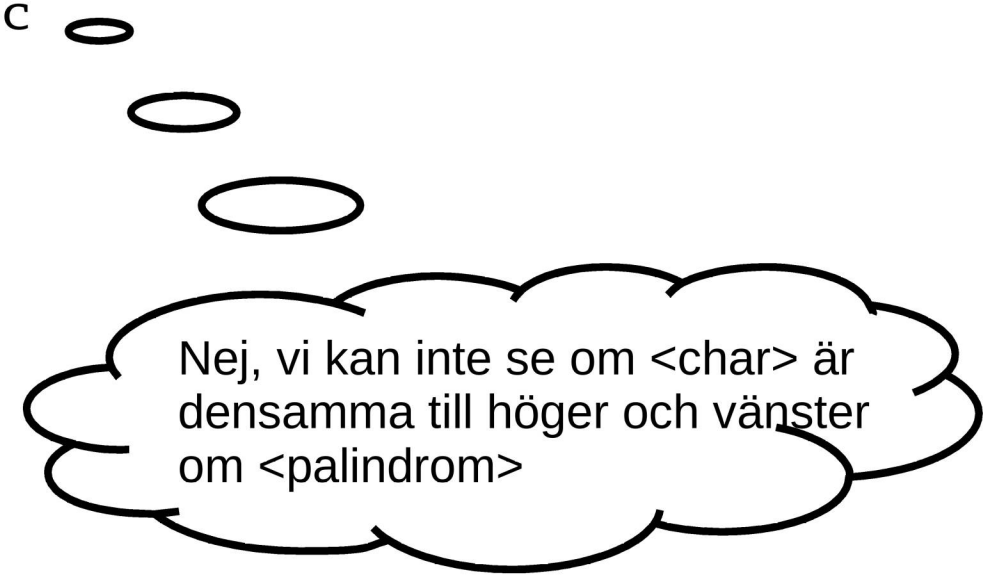
```
<palindrom> ::= <char> <palindrom> <char>
<char>          ::= a|b|c
```

Lösning på nästa sida

Palindrom - strängar

- Hur kan man matcha palindrom med grammatik?
- aabaa abcba abba aaaa
- Fungerar följande? Varför eller varför inte?

```
<palindrom> ::= <char> <palindrom> <char>
<char>          ::= a|b|c
```



Palindrom – strängar Lösning1

- Hur kan man matcha palindrom med grammatik?
- aabaa abcba abba aaaa
- Fungerar följande? Varför eller varför inte?

```
<palindrom> ::= <char> <palindrom> <char>
<char>          ::= a|b|c
```

Lösning1

Utöka parserna för att tillåta en speciell notation som försäkrar likheten

Palindrom – strängar Lösning1

- Hur kan man matcha palindrom med grammatik?
- aabaa abcba abba aaaa
- Fungerar följande? Varför eller varför inte?

```
<palindrom> ::= <char> <palindrom> <char>
<char>          ::= a|b|c
```

Lösning2

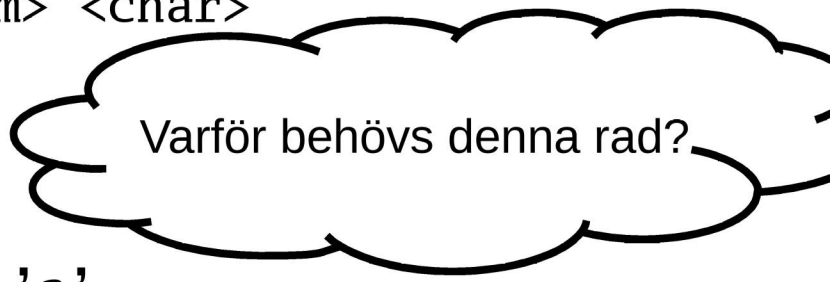
Skriv ut fallen

```
<palindrom> ::= 'a' <palindrom> 'a'
              | 'b' <palindrom> 'b'
              | 'b' <palindrom> 'b'
              | 'a' | 'b' | 'c' | <empty>
```

Palindrom – strängar Lösning1

- Hur kan man matcha palindrom med grammatik?
- aabaa abcba abba aaaa
- Fungerar följande? Varför eller varför inte?

```
<palindrom> ::= <char> <palindrom> <char>
<char>          ::= a|b|c
```



Lösning2

Skriv ut fallen

```
<palindrom> ::= 'a' <palindrom> 'a'
              | 'b' <palindrom> 'b'
              | 'b' <palindrom> 'b'
              | 'a' | 'b' | 'c' | <empty>
```

www.liu.se

LINKÖPINGS UNIVERSITET

Lista

LINKÖPINGS UNIVERSITET

Lista (samma problem som parametrar)

- <list> ::= slut
- | <element> <lista>
- | <lista>, <lista>
-
- <element> ::= 1|2
-
- Ett underprogram för varje icketerminal, i någon form av pseudokod
- lista() = if "slut" then true;
- if element() then lista();
- lista(); ','; lista();
-
- element() = '1' or '2';