Formella språk

Ett **alfabet** är en ändlig mängd av symboler
t ex bokstäver
{0,1}

En **sträng** över ett alfabet är en ändlig följd av symboler från alfabetet .
t ex apabepa
100101
ε epsilon = den tomma strängen

Ett **språk** är en mängd strängar.
t ex { a, aa, aaa, aaaa, ... }
{ 0, 1, 10, 11, 100, ...}
∅ Det tomma språket
{ ε }
{ ε , 10 }

Grammatiker

En kontext-fri grammatik består av fyra delar:

- En mängd terminala symboler
- En mängd icketerminala symboler
- En mängd produktionsregler
- En startsymbol, en av icketerminalerna

En grammatik, del 1

Produktionsregler

prog ::= **{** *satser* **}**

satser ::= *sats*
| *sats satser*

sats ::= **variabel =** *uttryck* **;**
| **while (** *uttryck* **)** *sats*
| **if (** *uttryck* **)** *sats* **else** *sats*
| **{** *satser* **}**

uttryck ::= *uttryck* **==** *uttryck*
| *uttryck* **+** *uttryck*
| *uttryck* ***** *uttryck*
| **variabel**
| **heltal**

En grammatik, del 2

Terminaler

if else while ( ) { } = == + * ;
variabel heltal

Icketerminaler

prog satser sats uttryck

Startsymbol

prog

Övning

Vilket är den kortaste sträng som grammatiken genererar?

prog

{ *satser* **}**

{ variabel = *uttryck* **; }**

{ variabel = heltal ; }

Härledning

Vänsterhärledning

I varje steg byts icketerminalen längs till vänster ut.

Högerhärledning

I varje steg byts icketerminalen längs till höger ut.

Vänsterhärledning

prog

{ *satser* **}**

{ *sats satser* **}**

{ variabel = *uttryck* **;** *satser* **}**

{ variabel = heltal ; *satser* **}**

{ variabel = heltal ; *sats* **}**

{ variabel = heltal ; while (*uttryck* **)** *sats* **}**

{ variabel = heltal ; while (variabel) *sats* **}**

.....

Högerhärledning

prog

{ *satser* **}**

{ *sats satser* **}**

{ *sats sats* **}**

{ *sats* **while (** *uttryck* **)** *sats* **}**

{ *sats* **while (** *uttryck* **) variabel =** *uttryck* **;}**

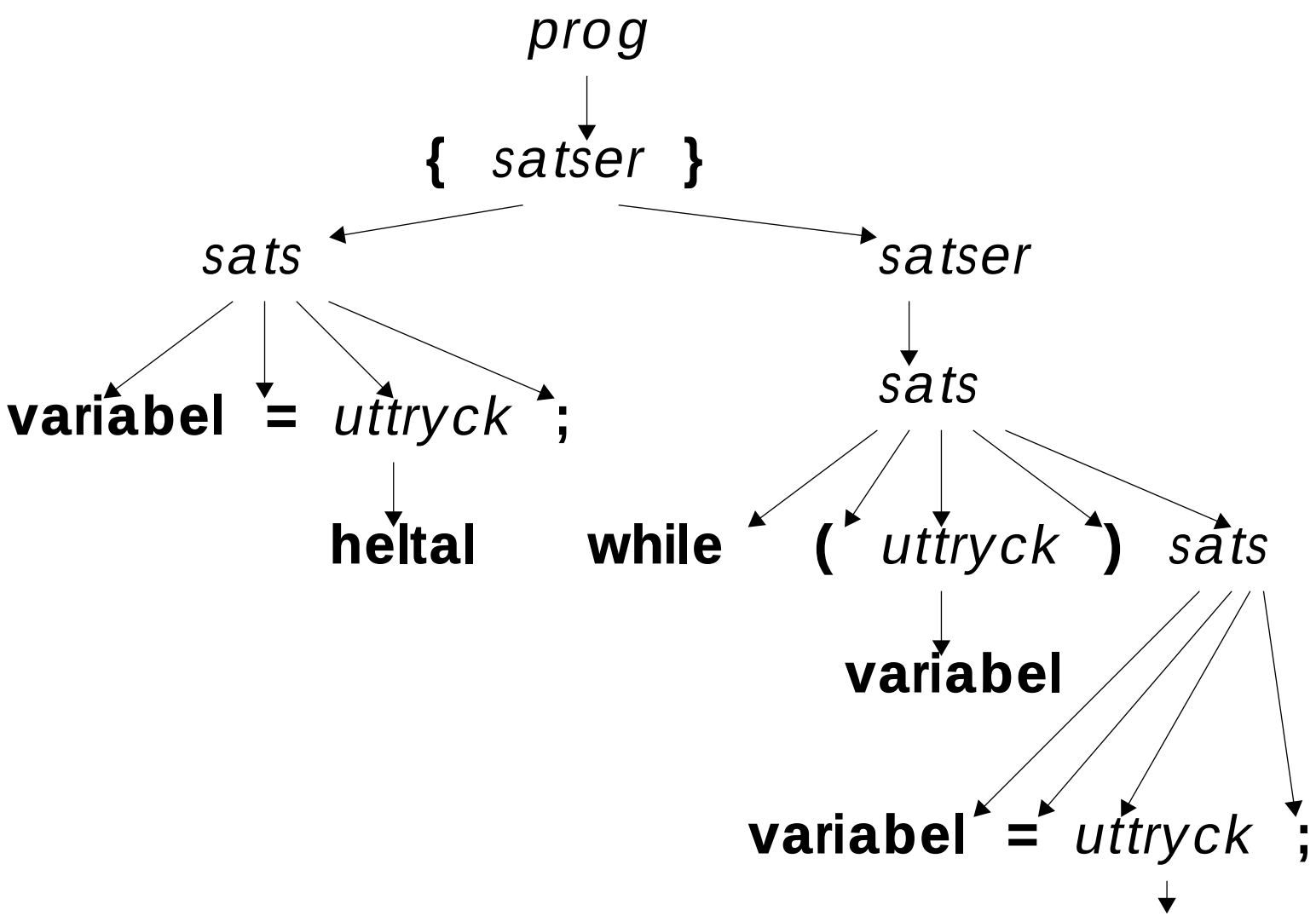
{ *sats* **while (** *uttryck* **) variabel =** *uttryck* **+** *uttryck* **;}**

{ *sats* **while (** *uttryck* **) variabel =** *uttryck* **+ heltal ;}**

{ *sats* **while (** *uttryck* **) variabel = variabel + heltal ;}**

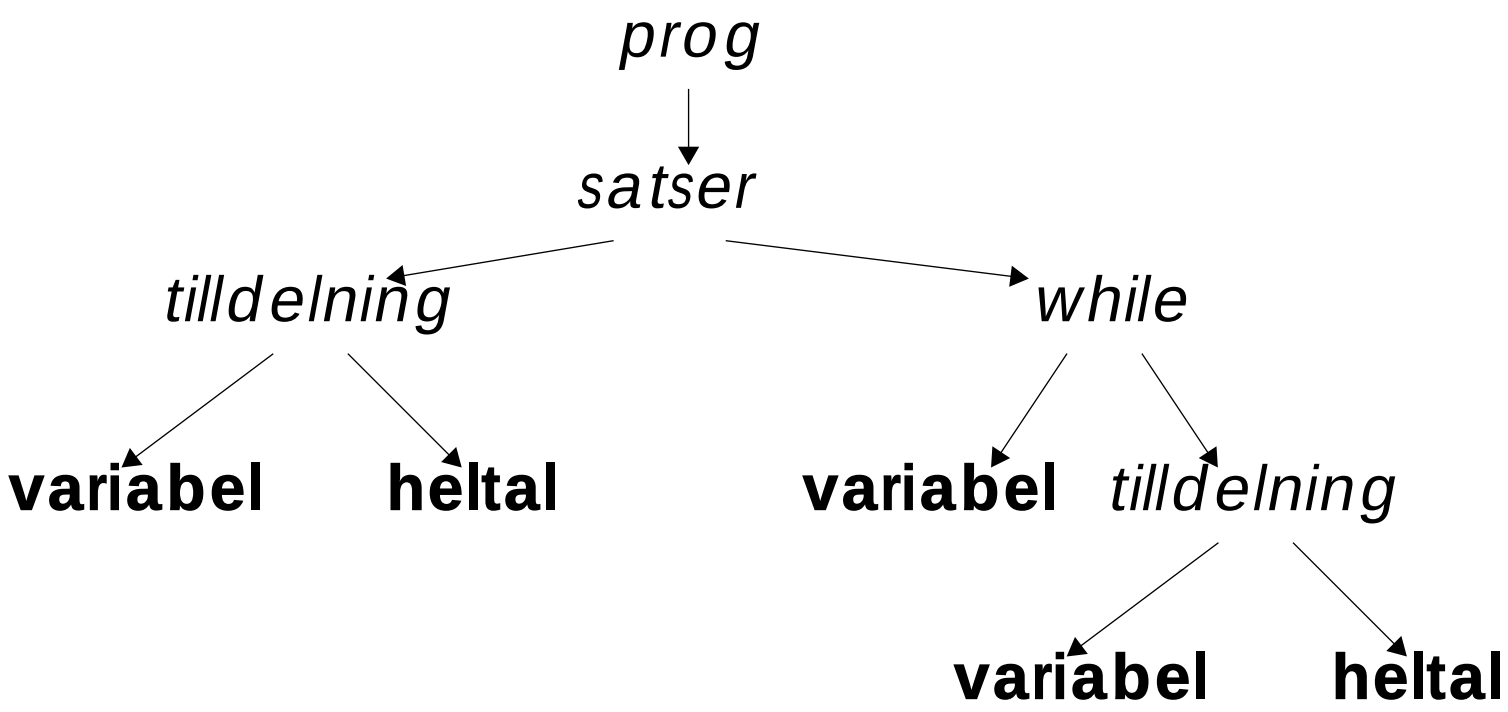
Parsträd - Härledningsträd

Representation av härledningen.



Abstrakt syntaxsträd

Endast strukturen kvar.



Abstrakt syntaxträd

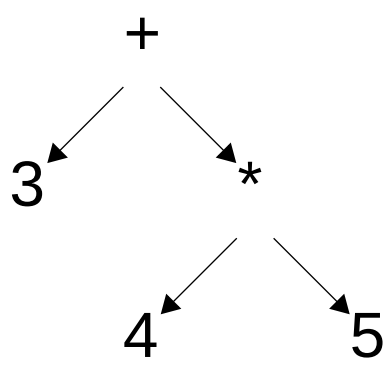
Abstrakta syntaxträd passer utmärkt som datatyp i Haskell.

```
data Prog = Prog Satser

data Satser = Satser [ Sats ]

data Sats = Tilldelning    Variabel Uttryck
          | While          Uttryck Sats
          | If             Uttryck Sats Sats
          | Block          Satser

data Uttryck =  Likhet     Uttryck Uttryck
             | Plus        Uttryck Uttryck
             | Mult        Uttryck Uttryck
             ...
```

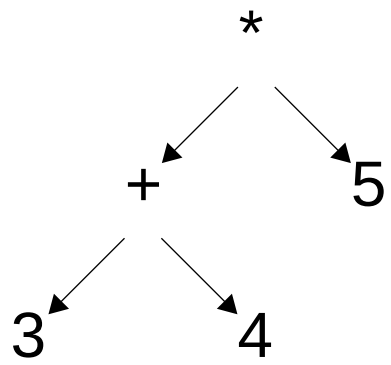
Flertydighet, “Ambiguity”

Vad är parseträdet för “3+4*5”?

uttryck
uttryck+uttryck
uttryck+uttryck*uttryck
...
3 + 4 * 5



uttryck
uttryck*uttryck
uttryck+uttryck*uttryck
...
3 + 4 * 5



En grammatik som kan generera minst en sträng med mer än ett parseträd är **flertydig**.

Flertydighet

Flertydighet kan tas bort genom att:

- Ändra språket

- Lägga till extra regler
  * har högre precedence än +
  + har högre precedence än ==
  *,+ är vänsterassociativa
  == är ickeassociativ (ändrar språket)

- Skriva om grammatiken

  uttryck ::= *uttryck2* **==** *uttryck2* | *uttryck2*
  uttryck2 ::= *uttryck2* **+** *uttryck3* | *uttryck3*
  uttryck3 ::= *uttryck3* ***** *uttryck4* | *uttryck4*
  uttryck4 ::= **variabel**
  | **heltal**

Flertydighet

"Dangling else"

sats ::= **if (** *uttryck* **)** *sats* **else** *sats*
| **if (** *uttryck* **)** *sats*

Strängen "if (*u1*) if(*u2*) *s1* else *s2*" är flertydig.

Tolkning 1:

```
if (u1)
    if (u2)
        s1
    else
        s2
```

Tolkning 2:

```
if (u1)
    if (u2)
        s1
else
    s2
```

Teori

Det är oavgörbart om en grammatik är flertydig.

Vissa mönster ger alltid flertydiga grammatiker:

- Om det finns en icketerminal som är både vänster och högerrekursiv.

- En grammatik med följande två produktioner

  $C := \alpha A \beta B$

  $\quad | \ \alpha A$

Ofta går det skriva om grammatiken, men inte alltid.

$\{a^i b^j c^k \mid i = j$ eller $j = k$ och $i,j,k \quad 0\}$

Chomskyhierarkin

typ 0 Inga restriktioner

$\alpha \rightarrow \beta$

typ 1 kontextberoende

$\alpha A \beta \rightarrow \alpha \gamma \beta$

typ 2 kontext-fria

$A \rightarrow \alpha$

typ 3 reguljära

A -> Bc | c eller

A -> cB | c

A är en icketerminal,

c en terminal,

α, γ, β är sekvenser av terminaler och icketerminaler.

Exempel

S = (S) S
  | ε

{wcw | w tillhör (a | b)*}

{wcw^R | w tillhör (a | b)*} (^{R} betyder “reverse”)

{$a^n b^n c^n$ | n 0}

{$a^n b^m c^n d^m$ | m,n 1}

{$a^n b^n c^m d^m$ | m,n 1}

{$a^n b^m c^m d^n$ | m,n 1}

Reguljära uttryck

- Symboler

  a $\qquad$ L(a) = { a }

- Alternativ

  a | b $\qquad$ L(a | b) = { a , b }

- Tomma strängen

  ε $\qquad$ L(ε) = { ε }

  alternativt

  L ( ) = { “ ” }

- Konkatenering

  a b $\qquad$ L(a b) = { ab }

- Repetering

  a* $\qquad$ L (a*) = { ε,a,aa,aaa ...}

Reguljära uttryck, utökningar

- Repetering en eller flera gånger

  a+ a+ = a a*

- Val (“optional”)

  a? a? = (a | ε)

- En av många

  [axz] [axz] = (a | x | z)
  [a-z] [a-z] = (a | b | ... | z)
  [a-zA-Z] [a-zA-Z] = (a | b | ... | z
  | A | B | ... | Z)

- Strängar, alternativt escapetecken

  “a.*+” L(“a.*+”) = { a.*+ }
  \a\.*\+ L(\a\.*\+) = { a.*+ }