INF5110

Oppgaver som gjennomgås 7/2 – 2014

Med svarforslag

Spørsmålene på de tre oppgave-foilene:

- Grammatikk med eksponensiering, lag entydig grammatikk
- Spørsmål om Tiny-grammatikken
- Lag klasser til et objektorientert AST

Samt følgende:

- Finn *First-* og *Follow* -mengdene til grammatikken i eksempelet på side 160 *etter at* venstre-rekursjon er fjernet.
- Beskriv en algoritme som bare finner de utnullbare ikke-terminaler, uten å beregne First og Follow.

Oppgave: Lag entydig grammatikk for uttrykk med høyreassosiativ eksponering

- Regler for presedens og assosiativitet for hver operasjon, slik at alle setninger får bare *ett* lovlig syntakstre:

+, –	lav,	venstre-ass.
*, /	høyere ,	venstre-ass.
↑ ,	høyest,	høyre-ass.

Altså: 3 + 5 / 3 * 2 + 4 ↑ 2 ↑ 3

Betyr: (3 + ((5 / 3) * 2)) + (4 ↑ (2 ↑ 3))

Lag entydig grammatikk for uttrykk med høyreassosiativ eksponensiering

- Regler for presedens og assosiativitet

+, –	lav,	venstre-ass.
*, /	høyere ,	venstre-ass.
↑ ,	høyest,	høyre-ass.

Altså: 3 + 5 / 3 * 2 + 4 ↑ 2 ↑ 3

Betyr: (3 + ((5 / 3) * 2)) + (4 ↑ (2 ↑ 3))

Altså, hvordan utvide denne??:

$$exp \rightarrow exp\ addop\ term \mid term$$
$$addop \rightarrow \mathbf{+} \mid \mathbf{-}$$
$$term \rightarrow term\ mulop\ factor \mid factor$$
$$mulop \rightarrow \mathbf{*}$$
$$factor \rightarrow \mathbf{(}\ exp\ \mathbf{)} \mid \mathbf{number}$$

Svar:
Entydig grammatikk for uttrykk med eksponensiering

Uten eksponensiering:

$$exp \rightarrow exp\ addop\ term \mid term$$
$$addop \rightarrow \mathbf{+} \mid \mathbf{-}$$
$$term \rightarrow term\ mulop\ factor \mid factor$$
$$mulop \rightarrow \mathbf{*}$$
$$factor \rightarrow \mathbf{(}\ exp\ \mathbf{)} \mid \mathbf{number}$$

Med eksponensiering. Vi treneger en ny ikke-terminal ”expon”:

exp -> *exp addop term* | *term*

addop -> **+** | **-**

term -> *term mulop factor* | *factor*

mulop -> *****

factor -> *expon eop factor* | *expon* Høyre-assosiativ!

eop -> ↑

expon -> **(** *exp* **)** | ***number***

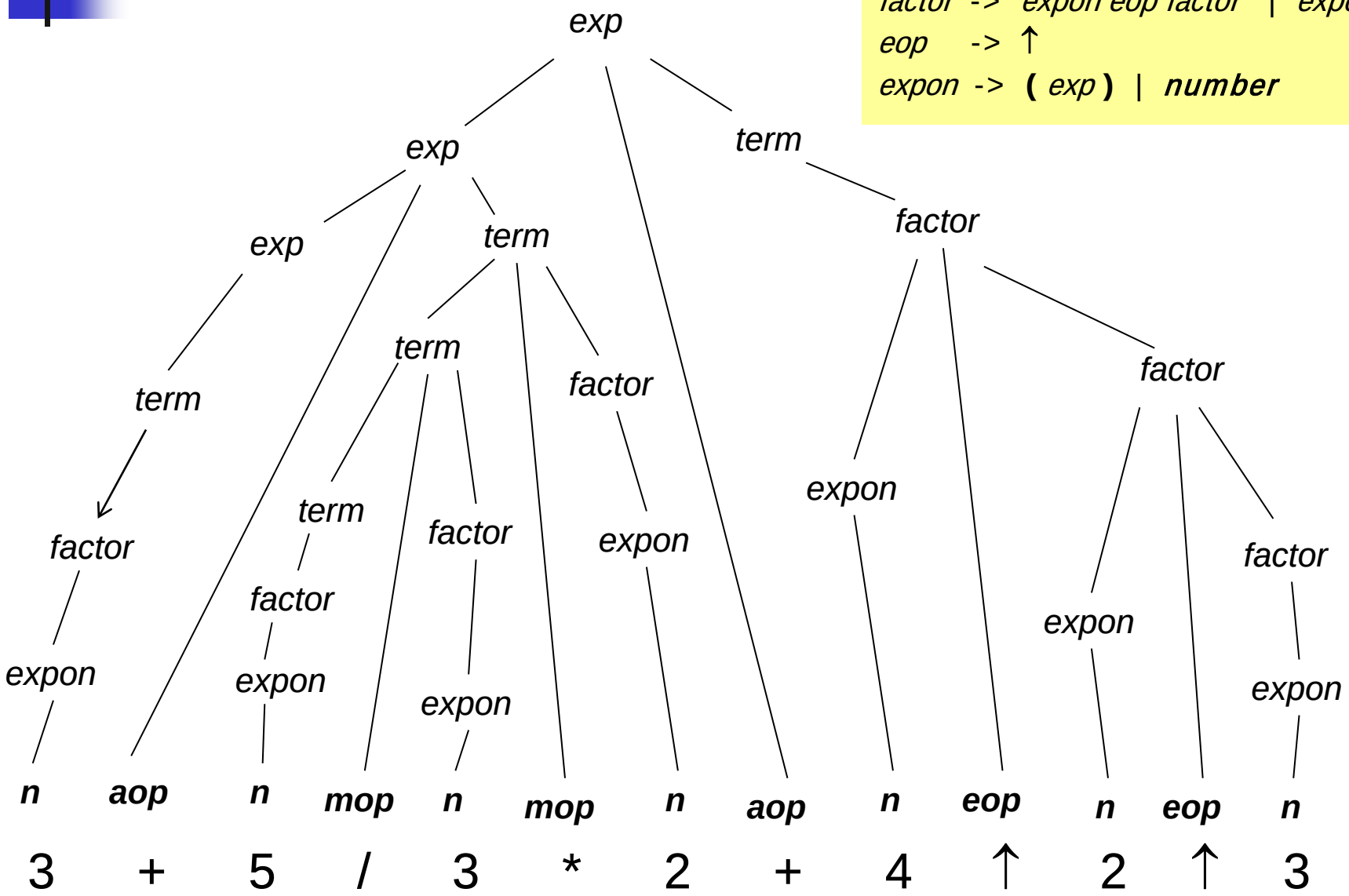
Tegn det konkrete syntaks-treet for avledning til:

3 + 5 / 3 * 2 + 4 ↑ 2 ↑ 3

exp -> *exp addop term* | *term*
addop -> **+** | **-**
term -> *term mulop factor* | *factor*
mulop -> ***
factor -> *expon eop factor* | *expon*
eop -> ↑
expon -> **(** *exp* **)** | ***number***

Det konkrete syntaks-treet

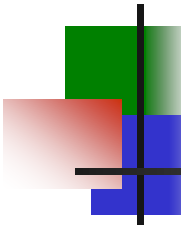
exp -> *exp addop term* | *term*
addop -> **+** | **-**
term -> *term mulop factor* | *factor*
mulop -> *****
factor -> *expon eop factor* | *expon*
eop -> ↑
expon -> **(** *exp* **)** | ***number***



Noen spørsmål om Tiny-grammatikken

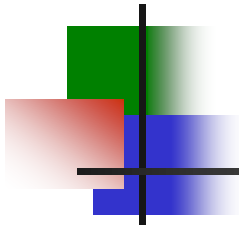
program → *stmt-sequence*
stmt-sequence → *stmt-sequence* **;** *statement* | *statement*
statement → *if-stmt* | *repeat-stmt* | *assign-stmt* | *read-stmt* | *write-stmt*
if-stmt → **if** *exp* **then** *stmt-sequence* **end**
| **if** *exp* **then** *stmt-sequence* **else** *stmt-sequence* **end**
repeat-stmt → **repeat** *stmt-sequence* **until** *exp*
assign-stmt → **identifier** **:=** *exp*
read-stmt → **read** **identifier**
write-stmt → **write** *exp*
exp → *simple-exp comparison-op simple-exp* | *simple-exp*
comparison-op → **<** | **=**
simple-exp → *simple-exp addop term* | *term*
addop → **+** | **-**
term → *term mulop factor* | *factor*
mulop → ***** | **/**
factor → **(** *exp* **)** | **number** | **identifier**

- *Er grammatikken entydig?*
- *Hva om vi vil tillate tomme setninger*
- *Hva om vi vil ha semikolon etter og ikke mellom setningene?*
- *Hva slags assossiativitet og presedens er det for operatorene?*



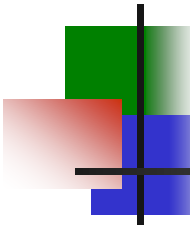
program → *stmt-sequence*
stmt-sequence → *stmt-sequence* **;** *statement* | *statement*
statement → *if-stmt* | *repeat-stmt* | *assign-stmt* | *read-stmt* | *write-stmt*
if-stmt → **if** *exp* **then** *stmt-sequence* **end**
| **if** *exp* **then** *stmt-sequence* **else** *stmt-sequence* **end**
repeat-stmt → **repeat** *stmt-sequence* **until** *exp*
assign-stmt → **identifier** **:=** *exp*
read-stmt → **read** **identifier**
write-stmt → **write** *exp*
exp → *simple-exp comparison-op simple-exp* | *simple-exp*
comparison-op → **<** | **=**
simple-exp → *simple-exp addop term* | *term*
addop → **+** | **-**
term → *term mulop factor* | *factor*
mulop → ***** | **/**
factor → **(** *exp* **)** | **number** | **identifier**

- Er grammatikken entydig?
  - Dette er generelt *uavgjørbart* for generelle BNF-grammatikker
  - Vi skal se på metode for å avgjøre det for mange praktiske grammatikker
  - Denne er ihvertfall delt opp i presedens-nivåer, og har assossiativites-angivelse, og er nok derved entydig. If-setninger med både **else** og **end** er ikke noe problem for entydigheten (men derimot for top-down-parsering!)



program → *stmt-sequence*
stmt-sequence → *stmt-sequence* **;** *statement* | *statement*
statement → *if-stmt* | *repeat-stmt* | *assign-stmt* | *read-stmt* | *write-stmt*
if-stmt → **if** *exp* **then** *stmt-sequence* **end**
| **if** *exp* **then** *stmt-sequence* **else** *stmt-sequence* **end**
repeat-stmt → **repeat** *stmt-sequence* **until** *exp*
assign-stmt → **identifier** **:=** *exp*
read-stmt → **read** **identifier**
write-stmt → **write** *exp*
exp → *simple-exp comparison-op simple-exp* | *simple-exp*
comparison-op → **<** | **=**
simple-exp → *simple-exp addop term* | *term*
addop → **+** | **-**
term → *term mulop factor* | *factor*
mulop → ***** | **/**
factor → **(** *exp* **)** | **number** | **identifier**

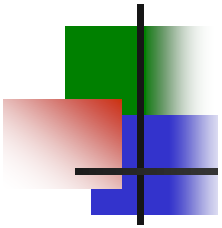
- Hva om vi vil tillate tomme setninger
  - Det er bare å sette til et tomt alternativ for *statement*
  - Den ser ut til fremdeles å være entydig



program → *stmt-sequence*
stmt-sequence → *stmt-sequence* **;** *statement* | *statement*
statement → *if-stmt* | *repeat-stmt* | *assign-stmt* | *read-stmt* | *write-stmt*
if-stmt → **if** *exp* **then** *stmt-sequence* **end**
| **if** *exp* **then** *stmt-sequence* **else** *stmt-sequence* **end**
repeat-stmt → **repeat** *stmt-sequence* **until** *exp*
assign-stmt → **identifier** **:=** *exp*
read-stmt → **read identifier**
write-stmt → **write** *exp*
exp → *simple-exp comparison-op simple-exp* | *simple-exp*
comparison-op → **<** | **=**
simple-exp → *simple-exp addop term* | *term*
addop → **+** | **-**
term → *term mulop factor* | *factor*
mulop → ***** | **/**
factor → **(** *exp* **)** | **number** | **identifier**

- Hva om vi vil ha semikolon etter og ikke mellom setningene?
  - Bytt ut reglen for stmt-sequence med:

    stmt-sequence - > stmt-sequence statement ; | statement ;
  - Også tomme setninger løser dette, og kanskje mer fleksibelt!



program → *stmt-sequence*
stmt-sequence → *stmt-sequence* **;** *statement* | *statement*
statement → *if-stmt* | *repeat-stmt* | *assign-stmt* | *read-stmt* | *write-stmt*
if-stmt → **if** *exp* **then** *stmt-sequence* **end**
| **if** *exp* **then** *stmt-sequence* **else** *stmt-sequence* **end**
repeat-stmt → **repeat** *stmt-sequence* **until** *exp*
assign-stmt → **identifier** **:=** *exp*
read-stmt → **read** **identifier**
write-stmt → **write** *exp*
exp → *simple-exp comparison-op simple-exp* | *simple-exp*
comparison-op → **<** | **=**
simple-exp → *simple-exp addop term* | *term*
addop → **+** | **-**
term → *term mulop factor* | *factor*
mulop → ***** | **/**
factor → **(** *exp* **)** | **number** | **identifier**

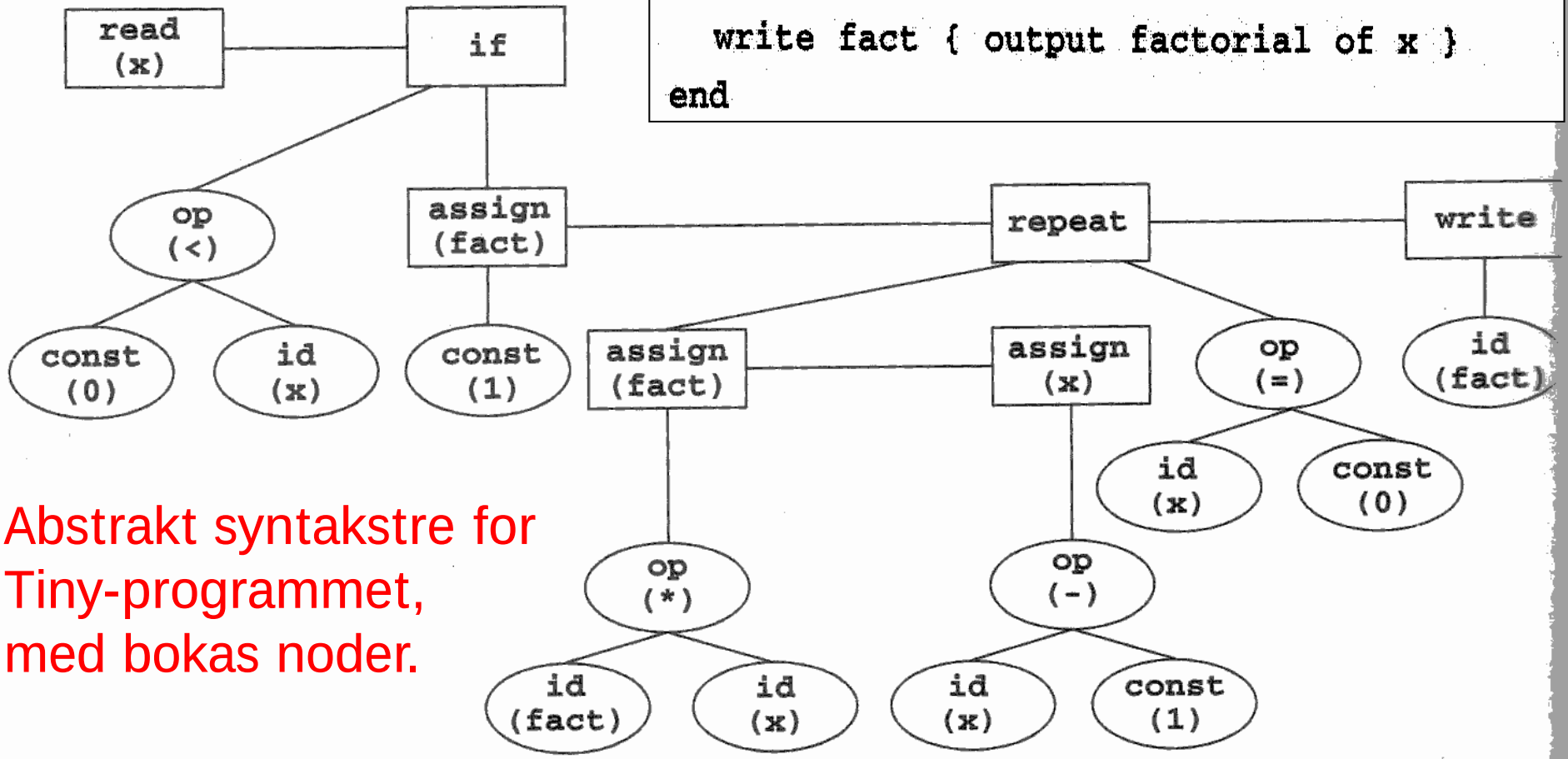
- Hva slags assosiativitet og presedens er det for operatorene?
  - Høyest presedens * / Venstre-assossiativ
  - Midlere presedens + - Venstre-assossiativ
  - Lavest presedens < = Ikke-assossiativ (akkurat to operander)

Kunne altså brukt en flertydig grammatikk, med disse tilleggs-reglene

Spørsmål:

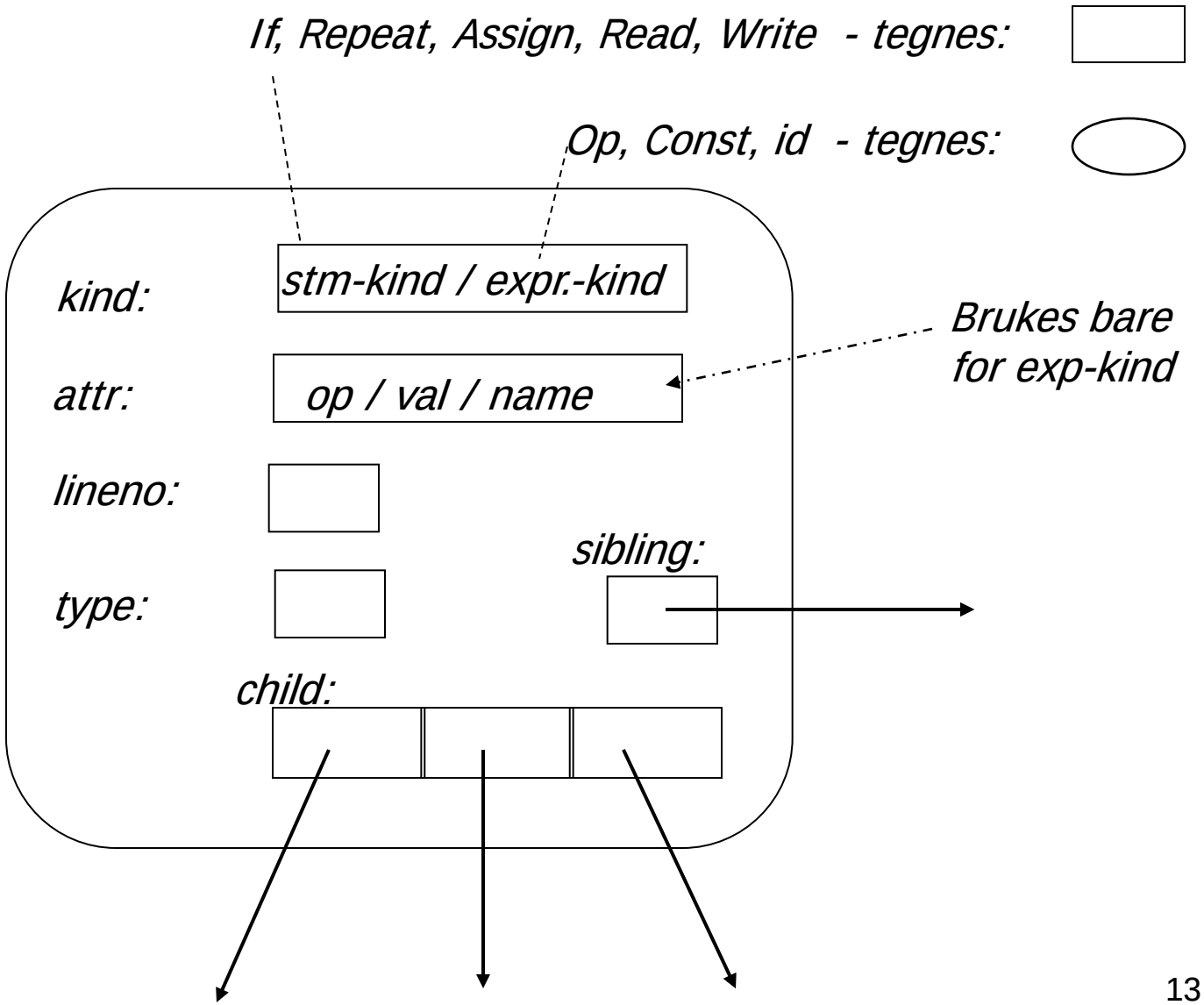
Finn et klassehierarki for nodeklassene til en objekt-orientert utgave av dette treet!

```
read x; { input an integer }
if 0 < x then { don't compute if x <= 0 }
  fact := 1;
  repeat
    fact := fact * x;
    x := x - 1
  until x = 0;
  write fact { output factorial of x }
end
```



Abstrakt syntakstre for Tiny-programmet, med bokas noder.

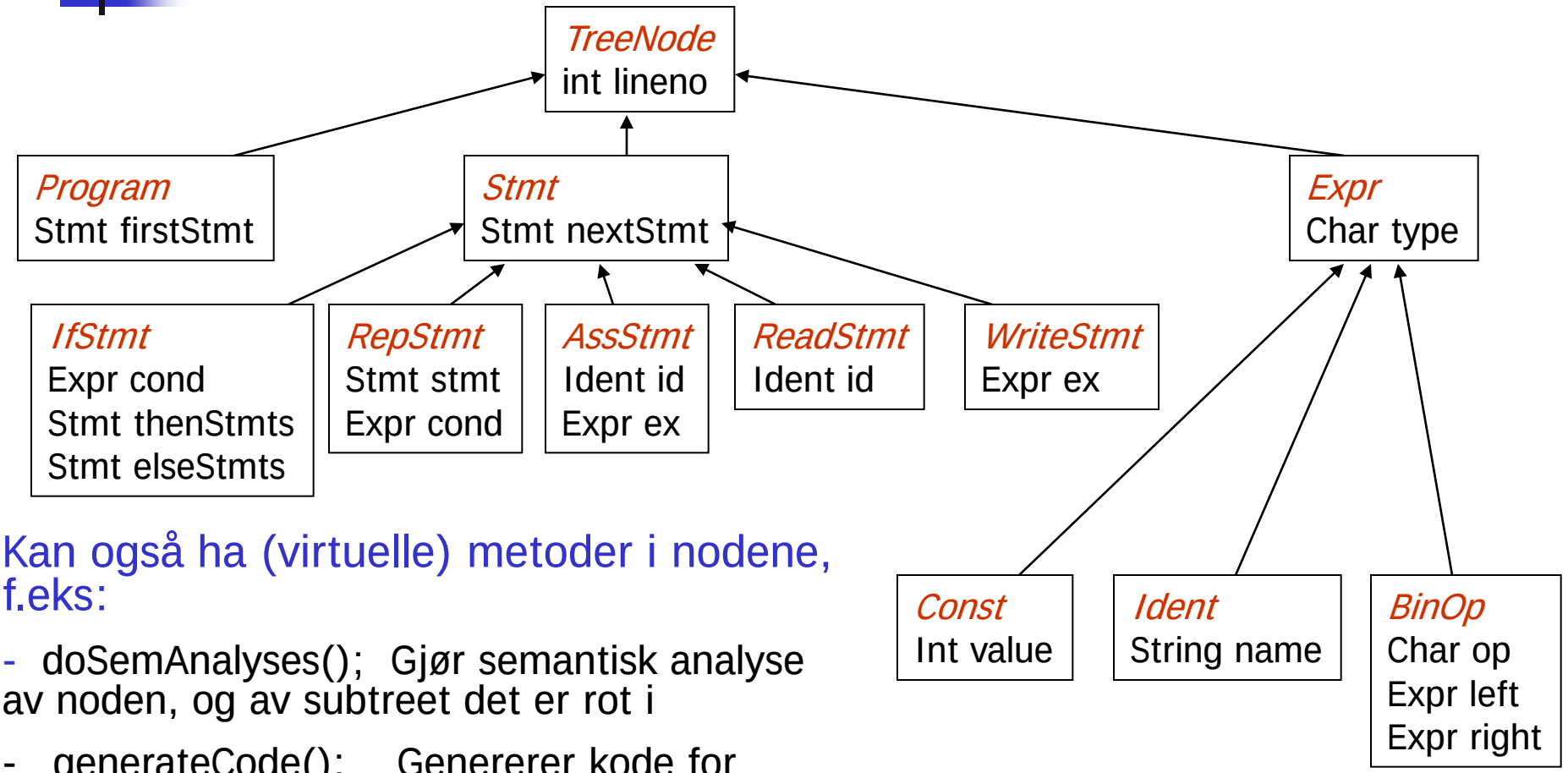
Nodestruktur i C for Tiny



Denne nodestrukturen passer enda bedre med et OO-språk med klasser /sublklasser **som implementasjons-språk.**

Nodeklasser for OO-utgave av abst.-synt.-tre for Tiny-språket.

Merk: Dette er altså en fast subklasse-struktur i kompilatoren, og må ikke forveksles med det abstrakte syntaks-treet for et gitt program!

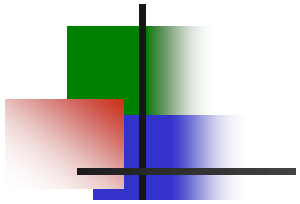


Kan også ha (virtuelle) metoder i nodene, f.eks:

- doSemAnalyses(); Gjør semantisk analyse av noden, og av subtreet det er rot i
- generateCode(); Genererer kode for noden, og for subtreet det er rot i

Merk at språket ikke har deklarasjoner, ellers ville vi trengt egne klasser for slike også

```
read x; { input an integer }
if 0 < x then { don't compute if x <= 0 }
  fact := 1;
  repeat
    fact := fact * x;
    x := x - 1
  until x = 0;
  write fact { output factorial of x }
end
```



Merk at ***typen*** vanligvis ikke blir satt før under semantikkanalysen, så den vil stå tom etter syntaksanalysen (= parseringen!).

Spørsmål: Finn First og Follow for transformert uttr.gram (Eks. 4.15 i boka)

Opprinnelig utgave:

$$exp \rightarrow exp\ addop\ term \mid term$$
$$addop \rightarrow \texttt{+} \mid \texttt{-}$$
$$term \rightarrow term\ mulop\ factor \mid factor$$
$$mulop \rightarrow \texttt{*}$$
$$factor \rightarrow \texttt{(}\ exp\ \texttt{)} \mid \textbf{\texttt{number}}$$

Har fjernet venstre-rekursjon:

$$exp \rightarrow term\ exp'$$
$$exp' \rightarrow addop\ term\ exp' \mid \varepsilon$$
$$addop \rightarrow \texttt{+} \mid \texttt{-}$$
$$term \rightarrow factor\ term'$$
$$term' \rightarrow mulop\ factor\ term' \mid \varepsilon$$
$$mulop \rightarrow \texttt{*}$$
$$factor \rightarrow \texttt{(}\ exp\ \texttt{)} \mid \textbf{\texttt{number}}$$

First- og Follow-mengder blir etter fjerningen:

First(exp) = {

First(exp') =

First($addop$) =

First($term$) =

First($term'$) =

First($mulop$) =

First($factor$) =

Spørsmål: Finn First og Follow for transformert uttr.gram (Eks. 4.15 i boka)

Opprinnelig utgave:

$exp \rightarrow exp\ addop\ term \mid term$
$addop \rightarrow \texttt{+} \mid \texttt{-}$
$term \rightarrow term\ mulop\ factor \mid factor$
$mulop \rightarrow \texttt{*}$
$factor \rightarrow \texttt{(}\ exp\ \texttt{)} \mid \textbf{\texttt{number}}$

Har fjernet venstre-rekursjon:

$exp \rightarrow term\ exp'$
$exp' \rightarrow addop\ term\ exp' \mid \varepsilon$
$addop \rightarrow \texttt{+} \mid \texttt{-}$
$term \rightarrow factor\ term'$
$term' \rightarrow mulop\ factor\ term' \mid \varepsilon$
$mulop \rightarrow \texttt{*}$
$factor \rightarrow \texttt{(}\ exp\ \texttt{)} \mid \textbf{\texttt{number}}$

First- og Follow-mengder blir etter fjerningen:

First(exp) = { (, **number**}
First(exp') = {+, -, ε}
First($addop$) = {+, -}
First($term$) = { (, **number**}
First($term'$) = {*, ε}
First($mulop$) = {*}
First($factor$) = { (, **number**}

Follow(exp) = {
Follow(exp') =
Follow($addop$) =
Follow($term$) =
Follow($term'$) =
Follow($mulop$) =
Follow($factor$) =

Spørsmål: Finn First og Follow for transformert uttr.gram (Eks. 4.15 i boka)

Opprinnelig utgave:

$$exp \rightarrow exp\ addop\ term \mid term$$
$$addop \rightarrow \texttt{+} \mid \texttt{-}$$
$$term \rightarrow term\ mulop\ factor \mid factor$$
$$mulop \rightarrow \texttt{*}$$
$$factor \rightarrow \texttt{(}\ exp\ \texttt{)} \mid \textbf{\texttt{number}}$$

Har fjernet venstre-rekursjon:

$$exp \rightarrow term\ exp'$$
$$exp' \rightarrow addop\ term\ exp' \mid \varepsilon$$
$$addop \rightarrow \texttt{+} \mid \texttt{-}$$
$$term \rightarrow factor\ term'$$
$$term' \rightarrow mulop\ factor\ term' \mid \varepsilon$$
$$mulop \rightarrow \texttt{*}$$
$$factor \rightarrow \texttt{(}\ exp\ \texttt{)} \mid \textbf{\texttt{number}}$$

First- og Follow-mengder blir etter fjerningen:

First(exp) = { (, **number** }	Follow(exp) = {$,) }
First(exp') = {+, -, ε}	Follow(exp') = {$,) }
First($addop$) = {+, -}	Follow($addop$) = { (, **number** }
First($term$) = { (, **number** }	Follow($term$) = {$,), +, - }
First($term'$) = {*, ε}	Follow($term'$) = {$,), +, - }
First($mulop$) = {*}	Follow($mulop$) = { (, **number** }
First($factor$) = { (, **number** }	Follow($factor$) = {$,), +, -, * }

Algoritme for å beregne mengden av utnullbare ikke-teminaler (UNB) uten å beregne First

Vi bruker en variabel ”UNB” som er en mengde av ikke-terminaler:

- Er fra starten tom
- Skal få nye elementer etter regel 1 og 2 under
- Når den ikke øker mer er vi ferdig

1. Først legges alle ikke-terminaler A som har en produksjon A -> ε, inn i UNB
2. (Gjentas til det ikke blir mer forandring):

   Om en ikke-terminal A har et alternativ:

   A -> ... | B C ... G | ...

   der alle B C ... G er ikke-terminaler som allerede er med i UNB, så legg også A inn i UNB.