

INF5110 – V2014

Stoff som i boka står i kap 4, men som er generelt stoff om grammatikker



5. februar 2014

Stein Krogdahl, Ifi, UiO



Dagens temaer

- Noe avsluttende fra sist
- First og Follow-mengder
 - Boka ser på en av parseringsmetodene først, uten å se på First/Follow-mengder.
 - Vi tar dette først (hører naturlig til i kap 3)
- To greie transformasjoner på grammatikker
 - Også naturlig i kap 3
 - Fjerning av venstre-rekursjon
 - Venstre-faktorisering
- Det blir altså i dag en del "detaljert pirk"
 - ... uten at vi nå får se anvendelser av det. Det kommer i de neste forelesningene
- Kanskje start på kapittel 4??

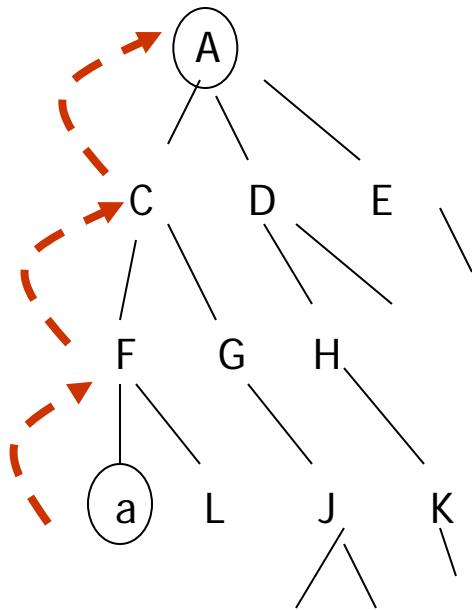
First og Follow-mengder, hva og hvorfor?

- For visse typer syntaksanalyse er det nødvendig å vite hvilke terminalsymboler som kan komme først i strenger som er avledet fra en ikketerminal A . Denne mengden kalles $First(A)$.
 - F.eks. vil $First(\textit{if-setning})$ i de fleste språk være bare nøkkelordet *"if"*, mens $First(\textit{tilordning})$ ofte vil være mengden $\{\textit{navn}, \textit{"("}\}$
- Tilsvarende vil vi få behov for å vite hvilke terminaler som kan følge etter en gitt ikketerminal A i *en eller annen* setningsform (= "halvutviklet setning") i språket. Denne mengden kalles $Follow(A)$
 - F.eks vil $Follow(\textit{statement})$ i Tiny-språket (eller snarere i den gitte grammatikken for Tiny-språket) være mengden $\{\textit{";"}, \textit{"end"}, \textit{"else"}, \textit{"until"}\}$
- Vi skal se på generelle algoritmer for å finne First- og Follow-mengdene for alle ikke-terminaler i en gitt grammatikk
 - Det som kompliserer disse algoritmene noe er produksjoner av typen $A \rightarrow \varepsilon$, som også gjør at vi kan få $B \Rightarrow^* \varepsilon$, selv om ikke $B \rightarrow \varepsilon$ direkte. Slike ikketerminaler A eller B sies å være *utnullbare* ("nullable")

Merk altså: First og Follow avhenger av grammatikken, ikke av språket!

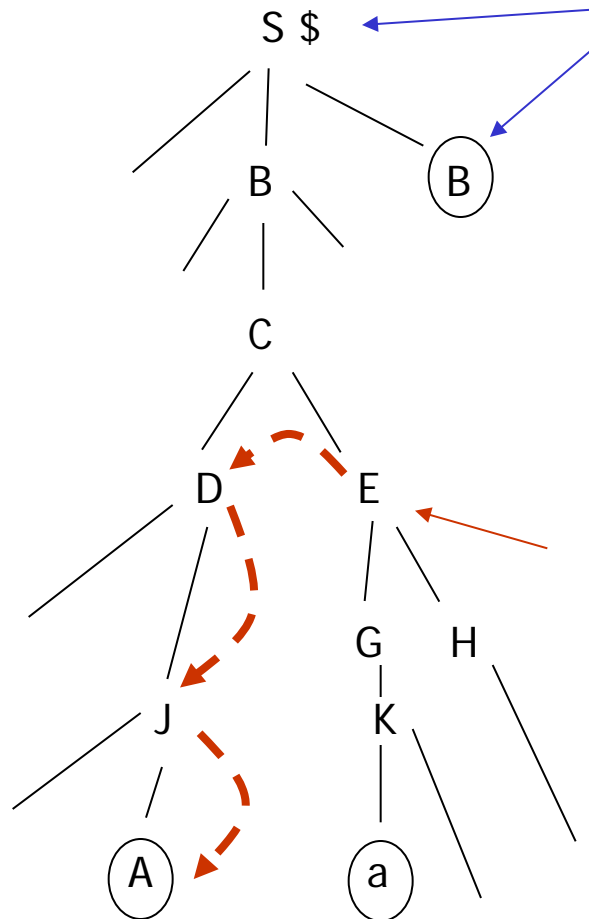
Typiske situasjoner: First- og Follow-mengder

$a \in \text{First}(A)$



Angir informasjonsflyt under algoritmene

$a \in \text{Follow}(A)$



$\$ \in \text{Follow}(B)$

Det at B kan stå på slutten av en setnings-form markeres ved å la Follow-mengden til B inneholde \$.

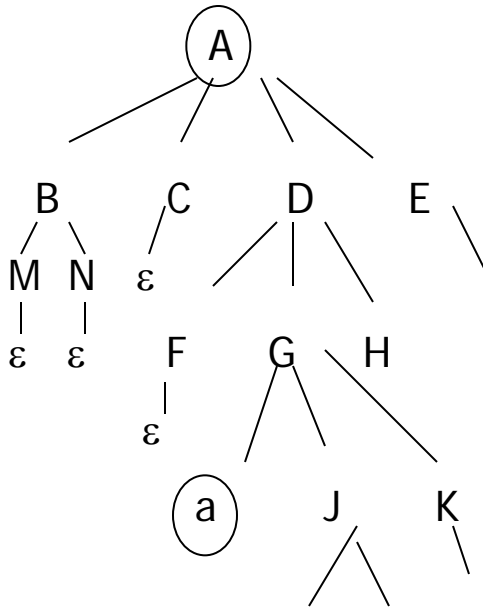
Terminal-symbolet \$ tenkes altså å stå på slutten av enhver setning.

Vet fra før at
 $a \in \text{First}(E)$

Merk: \$ er pr. def. med i Follow-mengden til startsymbolet

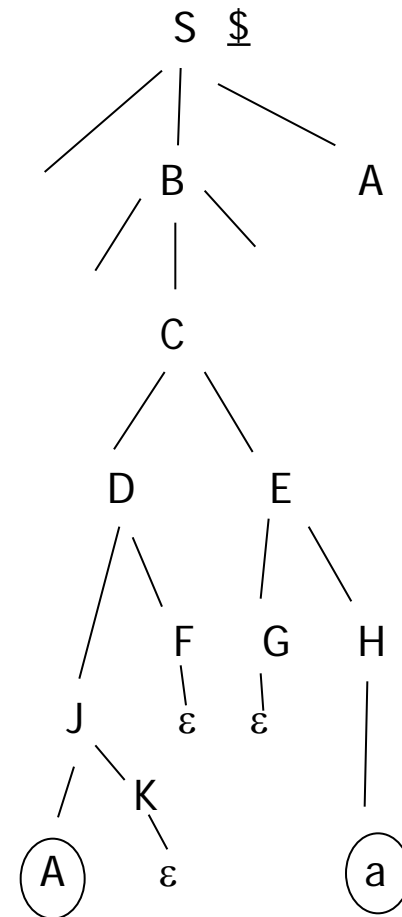
Kompliserte situasjoner: First- og Follow-mengder

$a \in \text{First}(A)$



Her er både B, M, N, C og F utnullbare. Vi markerer det ved å la deres First-mengde inneholde ϵ (men se en oppgave om dette).

$a \in \text{Follow}(A)$



First-mengder

terminal

Def.: $\left\{ \begin{array}{l} \text{First}(A) = \{ a \mid \text{finnes avledning } A \Rightarrow^* a \alpha \} \\ \text{Dessuten: Om } A \text{ "er utnullbar", s\aa er } \varepsilon \in \text{First}(A) \\ \text{Pr. def. For terminal-symboler: } \text{First}(a) = \{ a \} \end{array} \right.$

Def.: "A er utnullbar" hvis og bare hvis $A \Rightarrow^* \varepsilon$

Let X be a grammar symbol (a terminal or nonterminal) or ε . Then the set **First**(X) consisting of terminals, and possibly ε , is defined as follows.

1. If X is a terminal or ε , then $\text{First}(X) = \{X\}$.
2. If X is a nonterminal, then for each production choice $X \rightarrow X_1X_2 \dots X_n$, $\text{First}(X)$ contains $\text{First}(X_1) - \{\varepsilon\}$. If also for some $i < n$, all the sets $\text{First}(X_1), \dots, \text{First}(X_i)$ contain ε , then $\text{First}(X)$ contains $\text{First}(X_{i+1}) - \{\varepsilon\}$. If all the sets $\text{First}(X_1), \dots, \text{First}(X_n)$ contain ε , then $\text{First}(X)$ also contains ε .

Now define **First**(α) for any string $\alpha = X_1X_2 \dots X_n$ (a string of terminals and non-terminals), as follows. $\text{First}(\alpha)$ contains $\text{First}(X_1) - \{\varepsilon\}$. For each $i = 2, \dots, n$, if $\text{First}(X_k)$ contains ε for all $k = 1, \dots, i - 1$, then $\text{First}(\alpha)$ contains $\text{First}(X_i) - \{\varepsilon\}$. Finally, if for all $i = 1, \dots, n$, $\text{First}(X_i)$ contains ε , then $\text{First}(\alpha)$ contains ε .

← Tar vi som *algoritme* for \aa finne *First*(), se nederst.

← Vi snakker alts\aa ogs\aa om *First* av en hel streng α av terminaler og ikke-terminaler

```
for all nonterminals A do First(A) := {};  
while there are changes to any First(A) do  
  for each production choice A → X1X2...Xn do  
    k := 1 ; Continue := true ;  
    while Continue = true and k <= n do  
      add First(Xk) - {ε} to First(A);  
      if ε is not in First(Xk) then Continue := false ;  
      k := k + 1 ;  
    if Continue = true then add ε to First(A);
```

Algoritme for \aa finne First(X):

- Gjør steg 1.
- Gjenta steg 2 til alle Firstmengdene har stabilisert seg.

Eks. 4.9 Beregning av First-mengde

Merk: "First" av et teminalsymbol er symbolet selv! Dette er underforstått i tabellen under.

- (1) $exp \rightarrow exp \text{ addop } term$
- (2) $exp \rightarrow term$
- (3) $addop \rightarrow +$
- (4) $addop \rightarrow -$
- (5) $term \rightarrow term \text{ mulop } factor$
- (6) $term \rightarrow factor$
- (7) $mulop \rightarrow *$
- (8) $factor \rightarrow (exp)$
- (9) $factor \rightarrow \mathbf{number}$

Grammar rule	Pass 1	Pass 2	Pass 3
$exp \rightarrow exp$ $addop \text{ } term$			
$exp \rightarrow term$			First(exp) = { (, number }
$addop \rightarrow +$	First($addop$) = { + }		
$addop \rightarrow -$	First($addop$) = { +, - }		
$term \rightarrow term$ $mulop \text{ } factor$			
$term \rightarrow factor$		First($term$) = { (, number }	
$mulop \rightarrow *$	First($mulop$) = { * }		
$factor \rightarrow (exp)$	First($factor$) = { (}		
$factor \rightarrow \mathbf{number}$	First($factor$) = { (, number }		

Kan fylle ut en tabell.

NB: Må ta alle rader samtidig:

	First		
exp			(n
addop	+ -		
term		(n	
mulop	*		
factor	(n		

NB: Boka gjør altså dette i en meget systematisk rekkefølge, men man kan godt ta reglene i vilkårlig rekkefølge, frem og tilbake til det "roer seg".

Beregning av Follow-mengder

Def: Follow (A) = { a | finnes avledning $S \Rightarrow^* \beta A a \gamma$ }
For stratsymbolet S er \$ med i followmenden

β og γ er virkårlige strenger

« $\beta A a \gamma$ » er en setningsform i denne grammatikken

Dvs.: Det finnes en avledning fra "S \$" til en setningsform hvor A kommer rett før a (a er en terminal)

Given a nonterminal A, the set **Follow(A)**, consisting of terminals, and possibly \$, is defined as follows.

1. If A is the start symbol, then \$ is in Follow(A).
2. If there is a production $B \rightarrow \alpha A \gamma$, then First(γ) - { ϵ } is in Follow(A).
3. If there is a production $B \rightarrow \alpha A \gamma$ such that ϵ is in First(γ), then Follow(A) contains Follow(B).

γ er utnullbar

```
Follow(start-symbol) := {$} ;  
for all nonterminals A  $\neq$  start-symbol do Follow(A) := {} ;  
while there are changes to any Follow sets do  
  for each production  $A \rightarrow X_1 X_2 \dots X_n$  do  
    for each  $X_i$  that is a nonterminal do  
      add First( $X_{i+1} X_{i+2} \dots X_n$ ) - { $\epsilon$ } to Follow( $X_i$ )  
      (* Note: if  $i=n$ , then  $X_{i+1} X_{i+2} \dots X_n = \epsilon$  *)  
      if  $\epsilon$  is in First( $X_{i+1} X_{i+2} \dots X_n$ ) then  
        add Follow(A) to Follow( $X_i$ )
```

Algoritme:

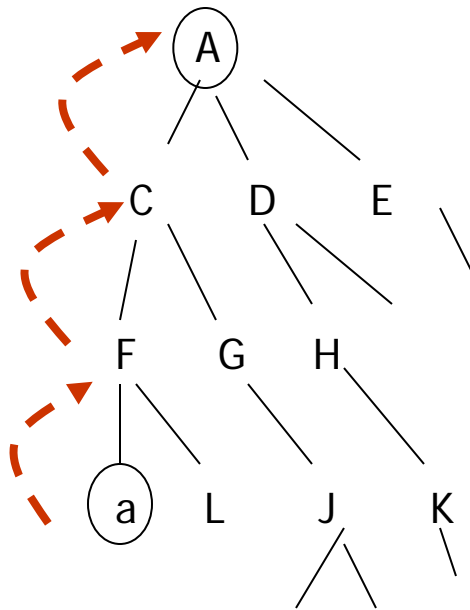
- Gjør steg 1.
- Gjenta steg 2 og 3 til alle Follow-mengdene har stabilisert seg.

NB: Man må altså gjøre dette samlet for alle ikke-terminaler

Samme som tidligere:

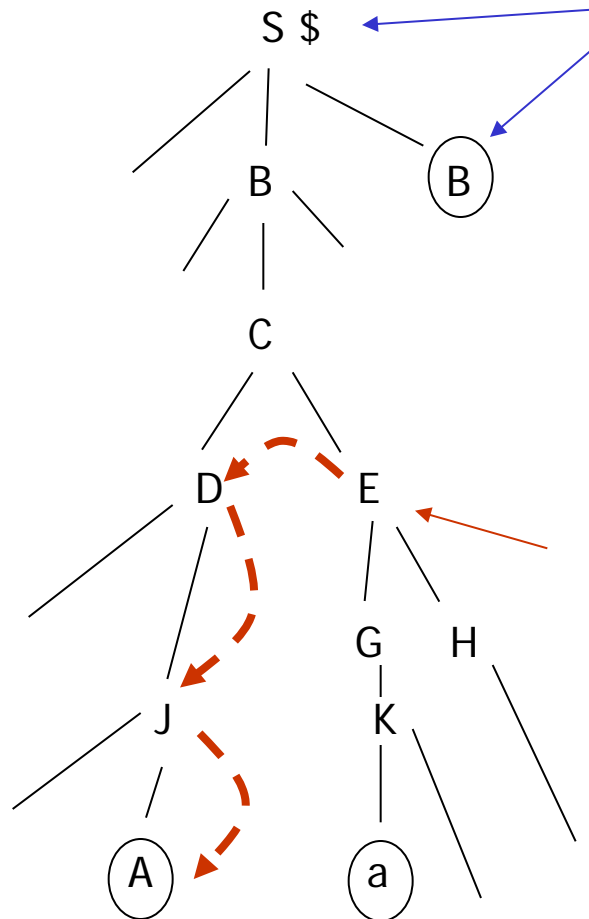
Beregning av First- og Follow-mengder

$a \in \text{First}(A)$



Angir informasjonsflyt under algoritmene

$a \in \text{Follow}(A)$



$\$ \in \text{Follow}(B)$

Det at B kan stå på slutten av en setnings-form markeres ved å la Follow-mengden til B inneholde \$.

Terminal-symbolet \$ tenkes altså å stå på slutten av enhver setning.

Vet fra før at
 $a \in \text{First}(E)$

Merk: \$ er pr. def. med i Follow-mengden til startsymbolet

Beregning av Follow-mengder: 4.12

- (1) $exp \rightarrow exp \text{ addop } term$
- (2) $exp \rightarrow term$
- (3) $addop \rightarrow +$
- (4) $addop \rightarrow -$
- (5) $term \rightarrow term \text{ mulop } factor$
- (6) $term \rightarrow factor$
- (7) $mulop \rightarrow *$
- (8) $factor \rightarrow (exp)$
- (9) $factor \rightarrow \mathbf{number}$

- $$\text{First}(exp) = \{ (, \mathbf{number} \}$$
- $$\text{First}(term) = \{ (, \mathbf{number} \}$$
- $$\text{First}(factor) = \{ (, \mathbf{number} \}$$
- $$\text{First}(addop) = \{ +, - \}$$
- $$\text{First}(mulop) = \{ * \}$$

- $$\text{Follow}(exp) = \{ \$, +, -,) \}$$
- $$\text{Follow}(addop) = \{ (, \mathbf{number} \}$$
- $$\text{Follow}(term) = \{ \$, +, -, *,) \}$$
- $$\text{Follow}(mulop) = \{ (, \mathbf{number} \}$$
- $$\text{Follow}(factor) = \{ \$, +, -, *,) \}$$

Grammar rule	Pass 1	Pass 2
$exp \rightarrow exp \text{ addop } term$	$\text{Follow}(exp) = \{ \$, +, - \}$ $\text{Follow}(addop) = \{ (, \mathbf{number} \}$ $\text{Follow}(term) = \{ \$, +, - \}$	$\text{Follow}(term) = \{ \$, +, -, *,) \}$
$exp \rightarrow term$		
$term \rightarrow term \text{ mulop } factor$	$\text{Follow}(term) = \{ \$, +, -, * \}$ $\text{Follow}(mulop) = \{ (, \mathbf{number} \}$ $\text{Follow}(factor) = \{ \$, +, -, * \}$	$\text{Follow}(factor) = \{ \$, +, -, *,) \}$
$term \rightarrow factor$		
$factor \rightarrow (exp)$	$\text{Follow}(exp) = \{ \$, +, -,) \}$	

Kan fylle ut en tabell ved å gå fram og tilbake i grammatikken til det stopper:

	Follow
exp	\$ + -)
addop	(n
term	\$ + - *)
mulop	(n
factor	\$ + - *)



Et par greie transformasjoner

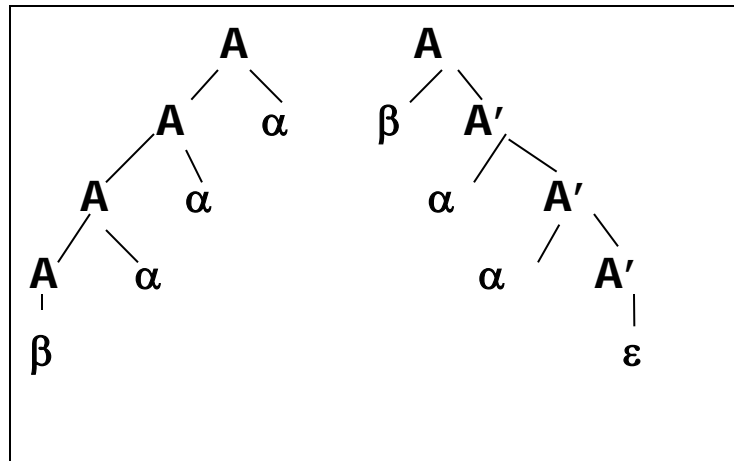
- For noen typer syntaksanalyse er det visse krav til grammatikken for at metoden skal fungere. To vanlige krav er:
 - Grammatikken må ikke ha venstrerekursjon
 - Altså, det må ikke finnes produksjoner av typen: $A \rightarrow A \alpha \mid \dots$
 - F.eks. går da ikke: $\text{expr} \rightarrow \text{expr} + \text{term} \mid \text{term}$
 - Grammatikken må ikke ha noen B (ikke-terminal), slik at to alternativer for B starter på samme måte.
 - Altså, det må ikke finnes slike: $B \rightarrow \alpha \beta \mid \dots \mid \alpha \gamma \mid \dots \quad \alpha \neq \varepsilon$
 - F.eks. går da ikke:
 $\text{if-setn} \rightarrow \mathbf{if} \text{ (bet) setn-sekv } \mathbf{end} \mid \mathbf{if} \text{ (bet) setn-sekv } \mathbf{else} \text{ setn-sekv } \mathbf{end}$

Fjerning av venstre-rekursjon - case 1

Kan skrives på EBNF

$$A \rightarrow A\alpha \mid \beta$$

$$A \rightarrow \beta \{\alpha\}$$



$$A \rightarrow \beta A'$$

$$A' \rightarrow \alpha A' \mid \epsilon$$

NB: Litt dumt, siden den opprinnelige venstreassosiative strukturen forsvinner. Må ev. korrigeres for.

Eksempel:

$$exp \rightarrow exp \text{ addop term} \mid \text{term}$$

This is of the form $A \rightarrow A \alpha \mid \beta$, with $A = exp$, $\alpha = \text{addop term}$, and $\beta = \text{term}$.
 Rewriting this rule to remove left recursion, we obtain

$$exp \rightarrow \text{term } exp'$$

$$exp' \rightarrow \text{addop term } exp' \mid \epsilon$$

Her er:

$$\alpha = \text{addop tem}$$

$$\beta = \text{term}$$

Fjerning av venstre-rekursjon – case 2

Case 2:

$$A \rightarrow A \alpha_1 \mid A \alpha_2 \mid \dots \mid A \alpha_n \mid \beta_1 \mid \beta_2 \mid \dots \mid \beta_m$$

Kan skrives i EBNF:

$$A \rightarrow (\beta_1 \mid \beta_2 \mid \dots \mid \beta_m) (\alpha_1 \mid \alpha_2 \mid \dots \mid \alpha_n)^*$$

$$\begin{aligned} A &\rightarrow \beta_1 A' \mid \beta_2 A' \mid \dots \mid \beta_m A' \\ A' &\rightarrow \alpha_1 A' \mid \alpha_2 A' \mid \dots \mid \alpha_n A' \mid \varepsilon \end{aligned}$$

Case 3 (indirekte venstre-rekursjon)

$$\begin{aligned} A &\rightarrow Ba \mid Aa \mid c \\ B &\rightarrow Bb \mid Ab \mid d \end{aligned}$$

- Her er det feil i boka i algoritmen s.159
- Ikke med som pensum
- Egner seg bare til maskinell behandling

Det som står i boka (vi forsøker ikke å forstå det):

$$\begin{aligned} A &\rightarrow B a A' \mid c A' \\ A' &\rightarrow a A' \mid \varepsilon \\ B &\rightarrow B b \mid A b \mid d \\ A &\rightarrow B a A' \mid c A' \\ A' &\rightarrow a A' \mid \varepsilon \\ B &\rightarrow B b \mid B a A' b \mid c A' b \mid d \\ A &\rightarrow B a A' \mid c A' \\ A' &\rightarrow a A' \mid \varepsilon \\ B &\rightarrow c A' b B' \mid d B' \\ B' &\rightarrow b B' \mid a A' b B' \mid \varepsilon \end{aligned}$$

Eks. side 160 – Fjerning av venstre-rekursjon

Den tradisjonelle entydige grammatikken

$$\begin{aligned} \text{exp} &\rightarrow \text{exp addop term} \mid \text{term} \\ \text{addop} &\rightarrow + \mid - \\ \text{term} &\rightarrow \text{term mulop factor} \mid \text{factor} \\ \text{mulop} &\rightarrow * \\ \text{factor} &\rightarrow (\text{exp}) \mid \mathbf{number} \end{aligned}$$
$$A \rightarrow A\alpha \mid \beta$$

$$\begin{aligned} A &\rightarrow \beta A' \\ A' &\rightarrow \alpha A' \mid \varepsilon \end{aligned}$$

Med fjernet venstre-rekursjon (beholder entydigheten, men ikke assosiativiteten):

$$\begin{aligned} \text{exp} &\rightarrow \text{term exp}' \\ \text{exp}' &\rightarrow \text{addop term exp}' \mid \varepsilon \\ \text{addop} &\rightarrow + \mid - \\ \text{term} &\rightarrow \text{factor term}' \\ \text{term}' &\rightarrow \text{mulop factor term}' \mid \varepsilon \\ \text{mulop} &\rightarrow * \\ \text{factor} &\rightarrow (\text{exp}) \mid \mathbf{number} \end{aligned}$$

Venstrefaktorisering, problemet "B \rightarrow $\alpha \beta \mid \alpha \gamma \mid \dots$ "

$stmt\text{-}sequence \rightarrow stmt \ ; \ stmt\text{-}sequence \mid stmt$
 $stmt \rightarrow \mathbf{s}$

Blir til: $stmt\text{-}sequence \rightarrow stmt \ stmt\text{-}seq'$
 $stmt\text{-}seq' \rightarrow \ ; \ stmt\text{-}sequence \mid \epsilon$

 $if\text{-}setn \rightarrow \mathbf{if} (bet) \ setn\text{-}sekv \ \mathbf{end} \mid \mathbf{if} (bet) \ setn\text{-}sekv \ \mathbf{else} \ setn\text{-}sekv \ \mathbf{end}$

Blir til: $if\text{-}setn \rightarrow \mathbf{if} (bet) \ setn\text{-}sekv \ else\text{-}eller\text{-}end$
 $else\text{-}eller\text{-}end \rightarrow \ \mathbf{else} \ setn\text{-}sekv \ \mathbf{end} \mid \ \mathbf{end}$

 $if\text{-}setn \rightarrow \mathbf{if} (bet) \ setn \mid \mathbf{if} (bet) \ setn \ \mathbf{else} \ setn$

Blir til: $if\text{-}setn \rightarrow \mathbf{if} (bet) \ setn \ else\text{-}eller\text{-}tom$
 $else\text{-}eller\text{-}tom \rightarrow \ \mathbf{else} \ setn \mid \ \epsilon$

Avansert venstre-faktorisering

Et litt mer komplisert tilfelle:

$$A \rightarrow abc B \mid abC \mid aE$$

Etter steg 1: $A \rightarrow ab A' \mid aE$
 $A' \rightarrow c B \mid C$

Etter steg 2 $A \rightarrow a A''$
(og ferdig): $A'' \rightarrow b A' \mid E$
 $A' \rightarrow c B \mid C$

while there are changes to the grammar **do**
for each nonterminal A **do**

let α be a prefix of maximal length that is shared
by two or more production choices for A

if $\alpha \neq \varepsilon$ **then**

let $A \rightarrow \alpha_1 \mid \alpha_2 \mid \dots \mid \alpha_n$ be all the production choices for A
and suppose that $\alpha_1, \dots, \alpha_k$ share α , so that
 $A \rightarrow \alpha \beta_1 \mid \dots \mid \alpha \beta_k \mid \alpha_{k+1} \mid \dots \mid \alpha_n$, the β_j 's share
no common prefix, and the $\alpha_{k+1}, \dots, \alpha_n$ do not share α

replace the rule $A \rightarrow \alpha_1 \mid \alpha_2 \mid \dots \mid \alpha_n$ by the rules

$$A \rightarrow \alpha A' \mid \alpha_{k+1} \mid \dots \mid \alpha_n$$

$$A' \rightarrow \beta_1 \mid \dots \mid \beta_k$$

} Gjenta så lenge det er muligheter

} Velg lengst mulig prefiks

} Gjør som over. Pass på å få med alle alternativer med dette prefikset



Kap. 4: Parsering ovenfra-ned (top-down)

Dette bør leses om igjen etter kapittel 4:

- *First* og *Follow*-mengder
 - Boka tar det et stykke uti kap 4, vi tok det altså før kap 4
- *LL(1)-parsering* og boka og pensum:
 - Det som i *boka* kalles LL(1)-parsering (4.2.1, 4.2.2 og 4.2.4) er en metode for top-down parsering med en eksplisitt stakk. Dette er *ikke* med som pensum.
 - Vi konsentrerer oss i dette kapittelet om "recursive descent"-parsering, en intuitiv metode som mange sikkert har vært litt borti (INF2100, ++)
 - Ofte brukes betegnelsen LL(1)-parsering også om "rec. desc."-metoden brukt ut fra syntaksdiagrammer, EBNF eller ren BNF, men man har da gjerne et ikke-teknisk forhold til om ting helt sikkert fungerer riktig.
 - Vi skal se på det tekniske kravet for at en ren BNF-grammatikk kan parseres rett fram med "rec. desc."-metoden.
 - **LL(1)-grammatikk i vår betydning:**
Det er en ren BNF-grammatikk som tilfredstiller et antall krav vi skal se på (nevnt i punktet over)

Eksempel: «Recursive descent» av ren BNF

```
exp → exp addop term | term
addop → + | -
term → term mulop factor | factor
mulop → *
factor → ( exp ) | number
```

Neste token → Global variabel

```
if a + b * ( c + d ) <= ...
```

Hoved-idé:

- Skriv en funksjon/prosedyre/metode for hver ikke-terminal
- La denne finne **riktig alternativ (helst fra bare førstkomende token)**, og gjør parsering av den videre input ut fra det

"Typisk" rec.decent-prosedyre for siste produksjon over, som blir veldig enkel.

```
procedure factor ;
begin
  case token of
    ( : match( ) ;
      exp ;
      match( ) ;
    number :
      match(number) ;
    else error ;
  end case ;
end factor ;
```

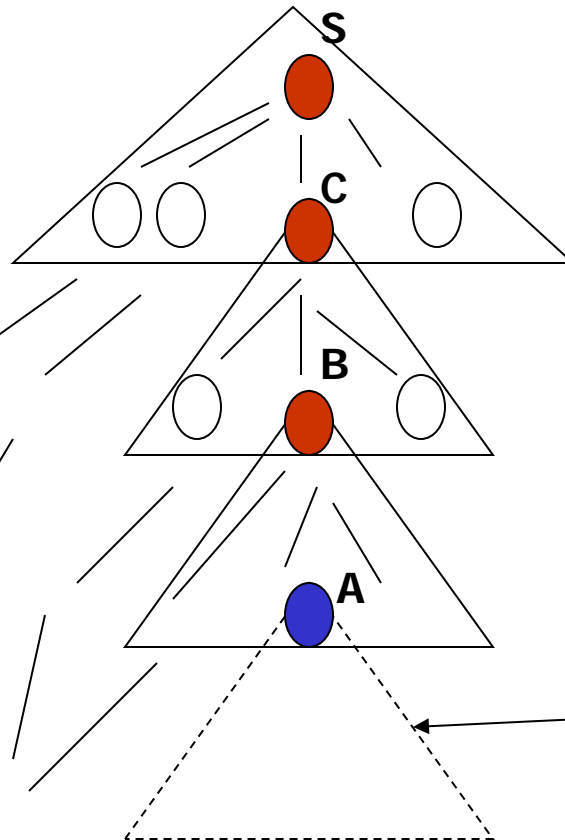
Sjekker at angitt terminal kommer, og "leser til neste". Brukes ofte *bare* for å lese (sjekken må slå til). Da er det egentlig nok å kalle "getToken" (men her kalles alltid "match")

```
procedure match ( expectedToken ) ;
begin
  if token = expectedToken then
    getToken ;
  else
    error ;
  end if ;
end match ;
```

Situasjonen under rekursiv parsing

Kalles altså "top down"-parsing (ovenfra-ned-parsing)

Trekantene er metode-kall

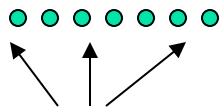


LL(1)-filosofi:

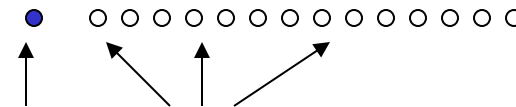
Hver metode kan alltid ved hjelp av (bare) det nåværende token, avgjøre hvilket alternativ som skal brukes for den aktuelle ikke-terminalen (her A)

Hvilket alternativ for A?

Input



Analysert og funnet OK



token Skal matches mot resten av det valgte alternativ for A

↖ Ligger i variabelen «token»

Ofte virker ikke den oppgitte BNF bra, men med venstrefaktorisering eller EBNF går det her greit

Opprinnelig

$if\text{-stmt} \rightarrow \mathbf{if} (exp) statement$
 $\quad \quad \quad | \mathbf{if} (exp) statement \mathbf{else} statement$

Ved $A = if\text{-stmt}$ og $token = \langle if \rangle$ er det to produksjoner å velge mellom. Altså ikke entydig!

Bokas forslag: Skrives ut som:

$if\text{-stmt} \rightarrow \mathbf{if} (exp) statement [\mathbf{else} statement]$

Rec. desc.-prosedyre:

```
procedure ifStmt ;
begin
  match (if) ;
  match ( ( ) ;
  exp ;
  match ( ) ;
  statement ;
  if token = else then
    match (else) ;
    statement ;
  end if ;
end ifStmt ;
```

NB: Kunne også bruke venstre-faktorisering. Da ville dette bli en egen prosedyre "elsePart":

$ifStmt \rightarrow \underline{\mathbf{if}} (exp) stmt \text{ elsePart}$
 $\text{elsePart} \rightarrow \epsilon \mid \underline{\mathbf{else}} stmt$