

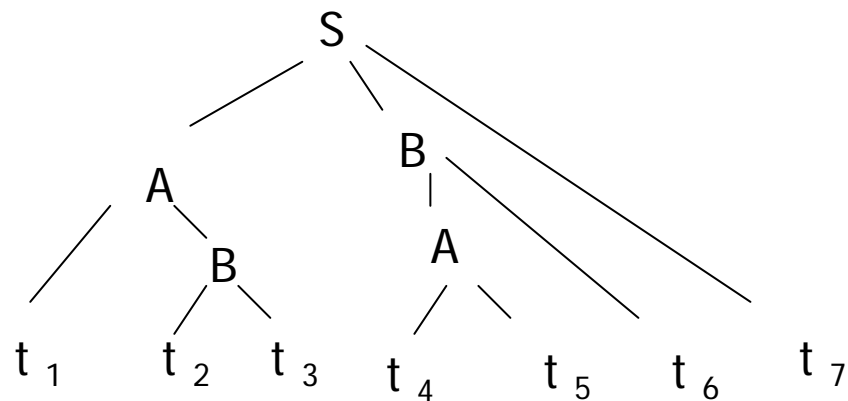
Kap. 5, del 1: Parsering nedefra-opp
(Bottom up parsing)
INF5110



NB: Disse foilene er litt justert og utvidet i forhold til de som er delt ut tidligere på en forelesning. Ta dem ut på nytt!

Stein Krogdahl
Ifi, UiO

"Bottom up" parsing (nedenfra-og-opp)



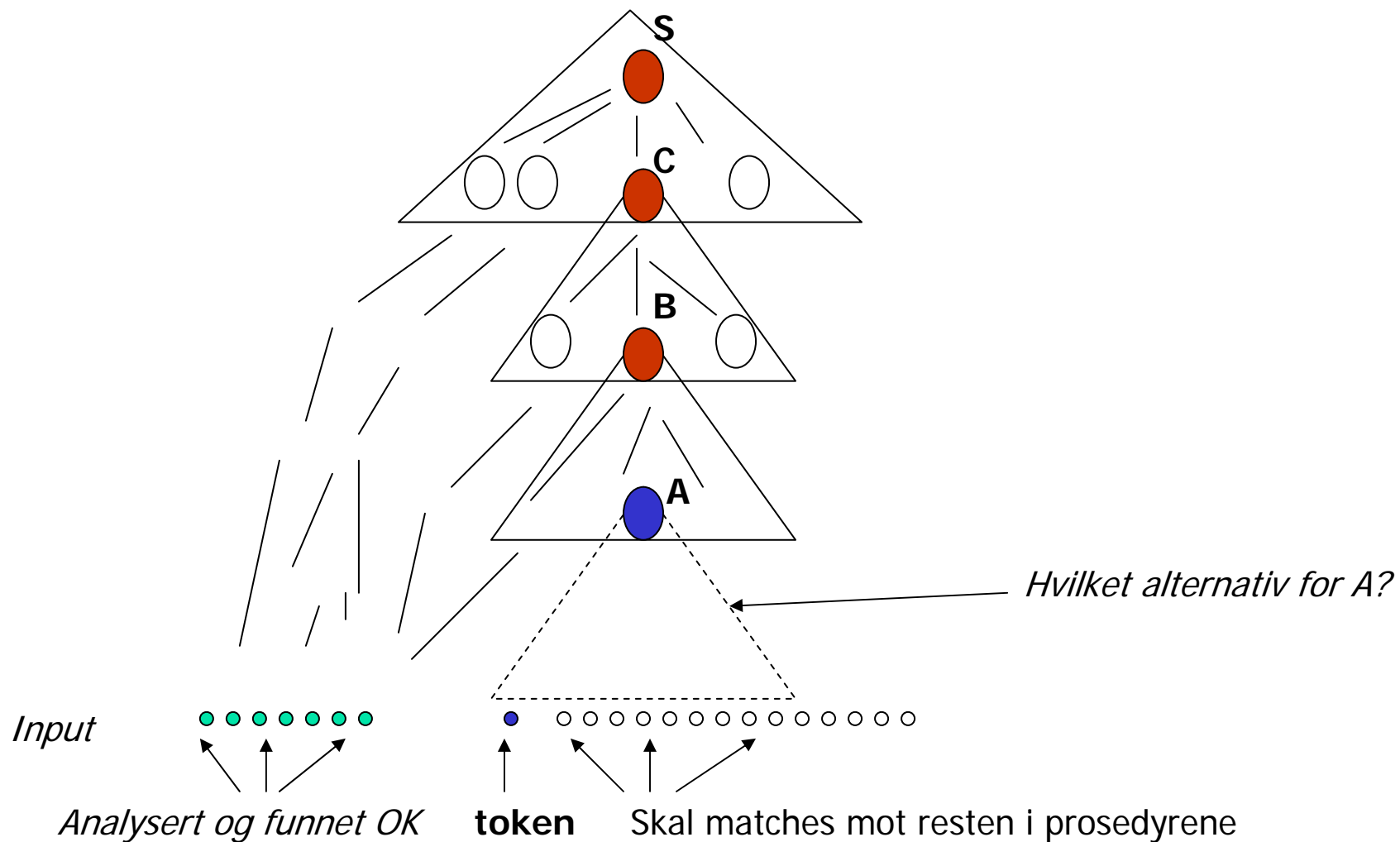
LR-parsing og grammatikker: Metodene angitt under kan i rekkefølge ta "vanskeligere og vanskeligere" grammatikker

- LR(0) Gir ca 300 "tilstander" på et vanlig programmeringsspråk
- SLR(1) -- Samme --
- LALR(1) -- Samme --
- LR(1) Gir ca 3000 "tilstander" på et vanlig programmeringsspråk

-Automatiserte verktøy som ut fra BNF-grammatikk leverer parserings-program:

-YACC, Bison, CUP (LALR(1))

Til sammenlikning: "Top down"-parsering



Prinsippet og datastrukturen for LR-parsering

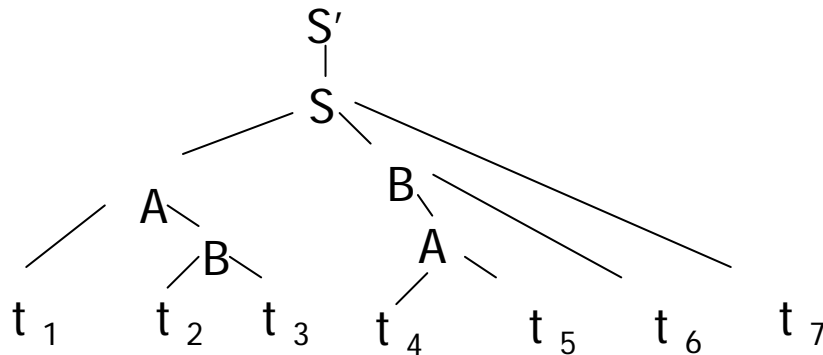
$S' \rightarrow S$

$S \rightarrow A B t_7 \mid \dots$

$A \rightarrow t_4 t_5 \mid t_1 B \mid$

$B \rightarrow t_2 t_3 \mid A t_6 \mid \dots$

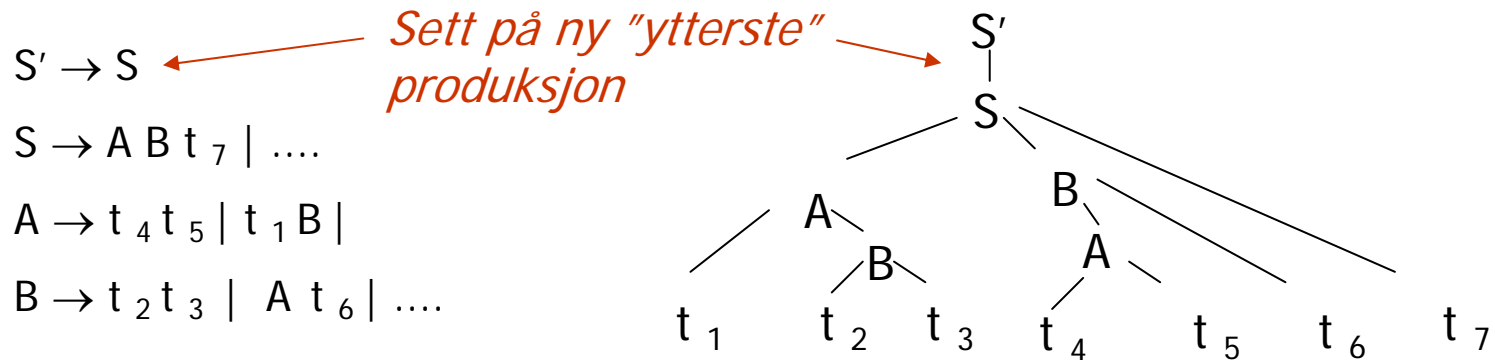
Anta at grammatikken er entydig, og at vi *kjenner syntaks-treet* for setningen:



LR-
parsering :

- Ha en "stakk" for *det som er lest*
- Gjør "reduksjonen" av et subtre når det ligger "på toppen av" stakken

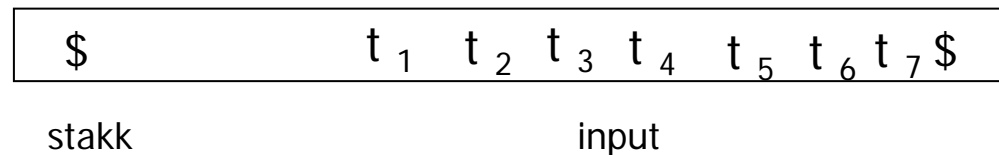
Prinsippet og datastrukturen LR-parsering II



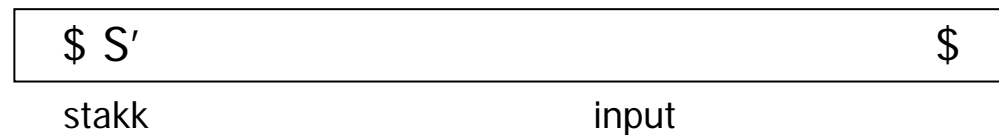
Anta at grammatikken er entydig, og anta at vi kjenner syntaks-treet for setningen:

- Ha en "stakk" for det som er lest (og "reduert"!)
- Gjør "reduksjonen" av et subtre når subtreet ligger "på toppen av" stakken.
- Da erstatter vi dette med den ikke-terminalen som produserte dette subtreet.
- Dette tilsvarer en bestemt produksjon brukt "omvendt"

Start-situasjonen:

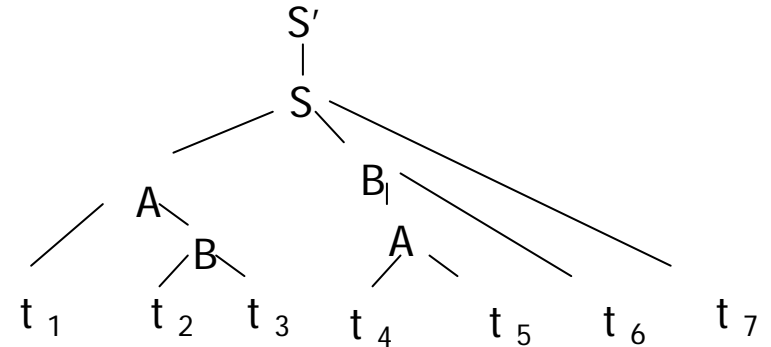


Slutt-situasjonen:



Prinsippet med LR-parsering III

$S' \rightarrow S$
 $S \rightarrow A B t_7 \mid \dots$
 $A \rightarrow t_4 t_5 \mid t_1 B \mid$
 $B \rightarrow t_2 t_3 \mid A t_6 \mid \dots$



stakk	input
\$	t ₁ t ₂ t ₃ t ₄ t ₅ t ₆ t ₇ \$
\$ t ₁	t ₂ t ₃ t ₄ t ₅ t ₆ t ₇ \$
\$ t ₁ t ₂	t ₃ t ₄ t ₅ t ₆ t ₇ \$
\$ t ₁ t ₂ t ₃	t ₄ t ₅ t ₆ t ₇ \$
\$ t ₁ B	t ₄ t ₅ t ₆ t ₇ \$
\$ A	t ₄ t ₅ t ₆ t ₇ \$
\$ A t ₄	t ₅ t ₆ t ₇ \$
\$ A t ₄ t ₅	t ₆ t ₇ \$
\$ A A	t ₆ t ₇ \$
\$ A A t ₆	t ₇ \$
\$ A B	t ₇ \$
\$ A B t ₇	\$
\$ S	\$

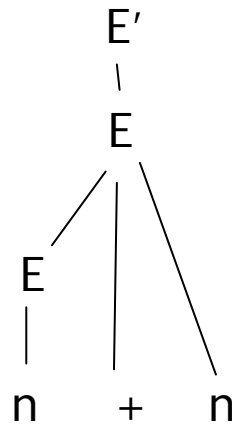
- Får to typer steg:
 - *Reduksjon* på toppen av stakken med bestemt $A \rightarrow \alpha$
 - *Lesing* ("skift") av input over til stakken
- Dersom man kjenner syntakstreet for den aktuelle setning, er det lett å angi de rette stegene.
- MEN: Hvordan gjøre dette underveis uten å kjenne resten av input og det fulle syntakstre??
- *Stakk + input*: Går gjennom en høyre-avledning, i omvendt rekkefølge

Husk at nå skal vi redusere input (bottom up) til startsymbolet S' , IKKE produsere input fra startesymbolet (slik vi gjorde "top-down" i kap 4)

Eksempel på LR-parsering

$$E' \rightarrow E$$

$$E \rightarrow E + n \mid n$$

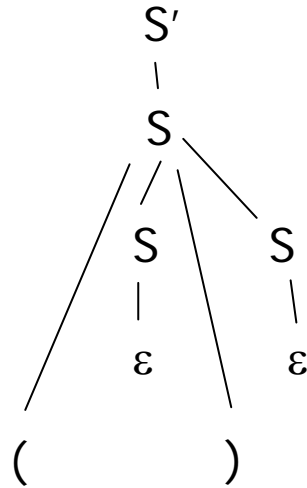


Boka: (Stakk + input) utgjør et stadium i en høyre-avledning. Den neste reduksjonen som skal gjøres, kalles situasjonens "handle" (håndtak).

	Parsing stack	Input	Action
1	\$	n + n \$	shift
2	\$ n	+ n \$	reduce $E \rightarrow n$
3	\$ E	+ n \$	shift
4	\$ E +	n \$	shift
5	\$ E + n	\$	reduce $E \rightarrow E + n$
6	\$ E	\$	reduce $E' \rightarrow E$
7	\$ E'	\$	accept

Eksempel på LR-parsering

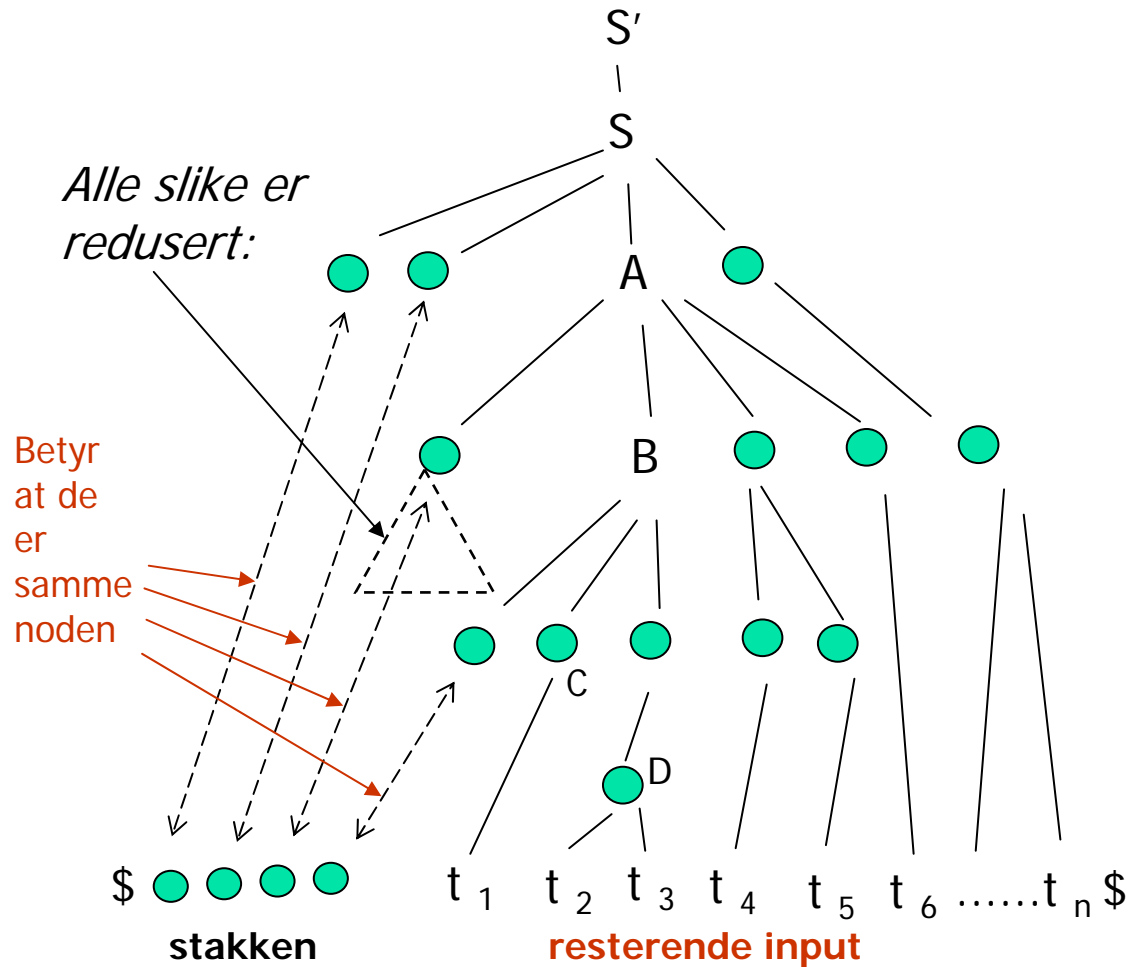
$S' \rightarrow S$
 $S \rightarrow (S) S \mid \varepsilon$



NB: $S \rightarrow \varepsilon$
blir litt "rar"

	Parsing stack	Input	Action
1	\$	()\$	shift
2	\$ ()\$	reduce $S \rightarrow \varepsilon$
3	\$ (S)\$	shift
4	\$ (S)	\$	reduce $S \rightarrow \varepsilon$
5	\$ (S) S	\$	reduce $S \rightarrow (S) S$
6	\$ S	\$	reduce $S' \rightarrow S$
7	\$ S'	\$	accept

Typisk situasjon under LR-parsering



$S_1 S_2 S_3 S_4 \dots S_k$

lest input (stakken er redusert utgave av dette)

Det neste reduksjonen som blir gjort er reduksjon med

$C \rightarrow t_1$

Deretter (etter noen skift):

$D \rightarrow t_2 t_3$

Gitt en entydig grammatikk G.

Denne behandles som følger for å lage en LR-parser:

(Ikke alt er beskrevet i boka, men bare boka er pensum)

- Vi ser på *alle* mulig stakker som kan opptre, og ser disse som strenger over alfabetet: $\{\text{terminaler}\} \cup \{\text{ikke-terminaler}\}$. Vi ser altså på:
 $\{S \mid S \text{ utgjør stakken på ett eller annet stadium i LR-parsering av en setning i } L(G)\}$
- Dette språket viser seg å være **regulært**, og kan beskrives av en NFA der alle tilstander er aksepterende:
 - Tilstandene angis av "itemer" av formen: $A \rightarrow X Y . Z$ (mer presist: "LR(0)-itemer")
 - Kantene kan beskrives nokså greit (kommer snart)
- Denne NFA-en gjør vi om til en DFA på vanlig måte (subset construction fra kap. 2)
- Tilstandene i denne DFA-en vil altså være **mengder av itemer**
- **LR-parseringens hovedsetning (uformelt):**
 - Se på en situasjon under LR-parsering av en setning, der vi har strengen S på stakken.
 - Strengen S vil da aksepteres av DFA-en angitt over. Anta at den bringer oss til DFA-tilstanden T.
 - Da vil mengden av *itemer* i T angi de mulige "lokale forhold" vi har ved det punktet vi nå er i i parseringen.
 - Selv om grammatikken er entydig er det ofte flere muligheter/valg (ett for hvert item). Dette kommer av at vi ikke har tatt hensyn til resten av input.



Altså "Itemer": Hver produksjon i BNF-grammatikken gir opphav til et antall slike.

- For produksjonen :

$$A \rightarrow \alpha X \beta$$

- Lager vi itemer med punktum på alle plasser (punktumet er et Meta-symbol):

$$A \rightarrow \cdot \alpha X \beta$$

$$A \rightarrow \alpha \cdot X \beta$$

$$A \rightarrow \alpha X \cdot \beta$$

$$A \rightarrow \alpha X \beta \cdot$$

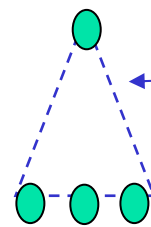
- Punktumet angir grovt sett hvor vi er hen i "dypeste aktive del" av parseringen i øyeblikket:
 - Det til venstre for punktum er gjenkjent fra input, enten bare lest eller at deler av det er redusert til en ikke-terminal
 - Det til høyre for punktum har vi enda ikke sett/lest

Kantene i NFA-en I

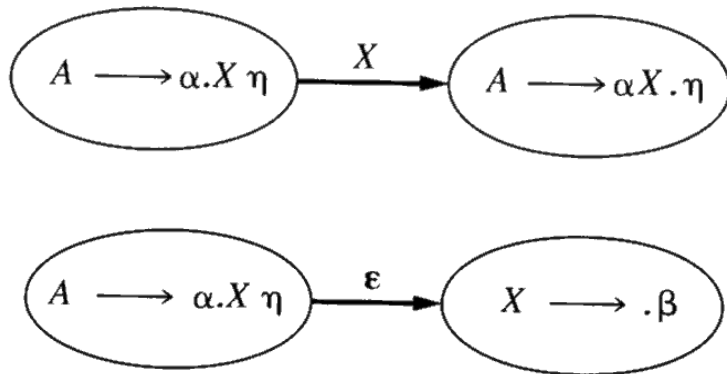
Dette er ikke fullt forklart i boka, og er ikke pensum

Skal lage en NFA som aksepterer alle slike "linjer", og intet annet

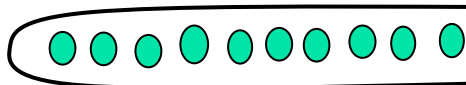
Gitt en grammatikk G og en situasjon under passering av en setning s i $L(G)$.



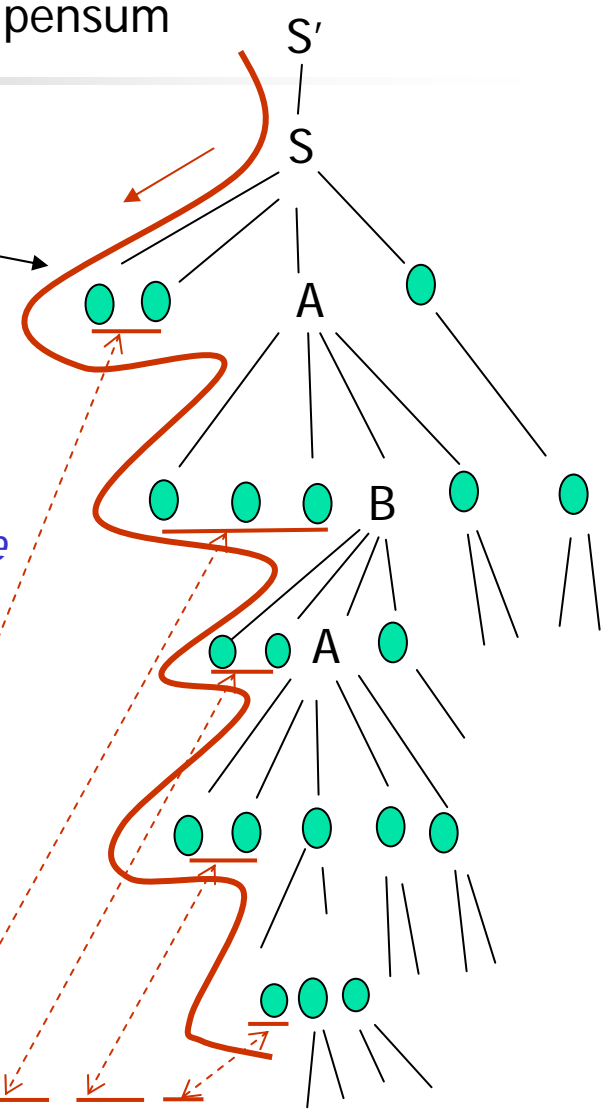
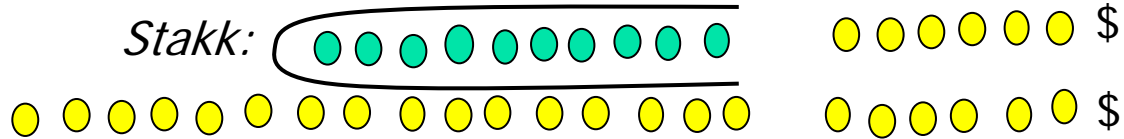
Husk: Ingen slik på venstre side. De er redusert!



Stakk:



Opprinnelig input:



Eksempel på NFA

$E' \rightarrow E$

$E \rightarrow E + n$

$E \rightarrow n$

$E' \rightarrow .E$

$E' \rightarrow E.$

$E \rightarrow .E + n$

$E \rightarrow E. + n$

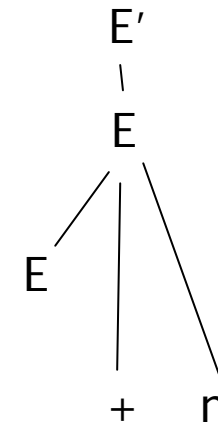
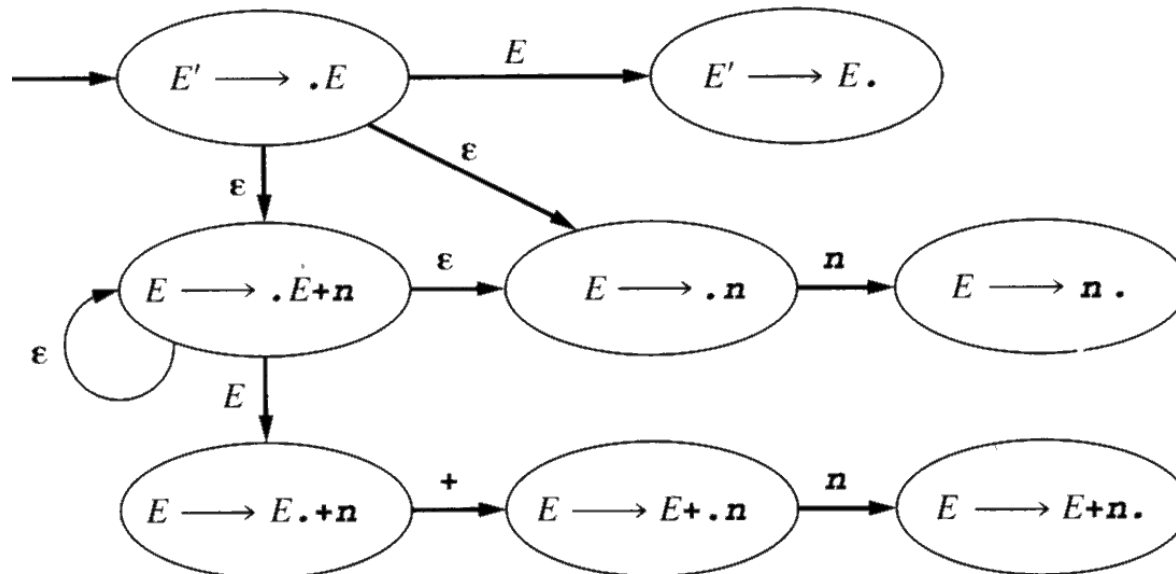
$E \rightarrow E + .n$

$E \rightarrow E + n.$

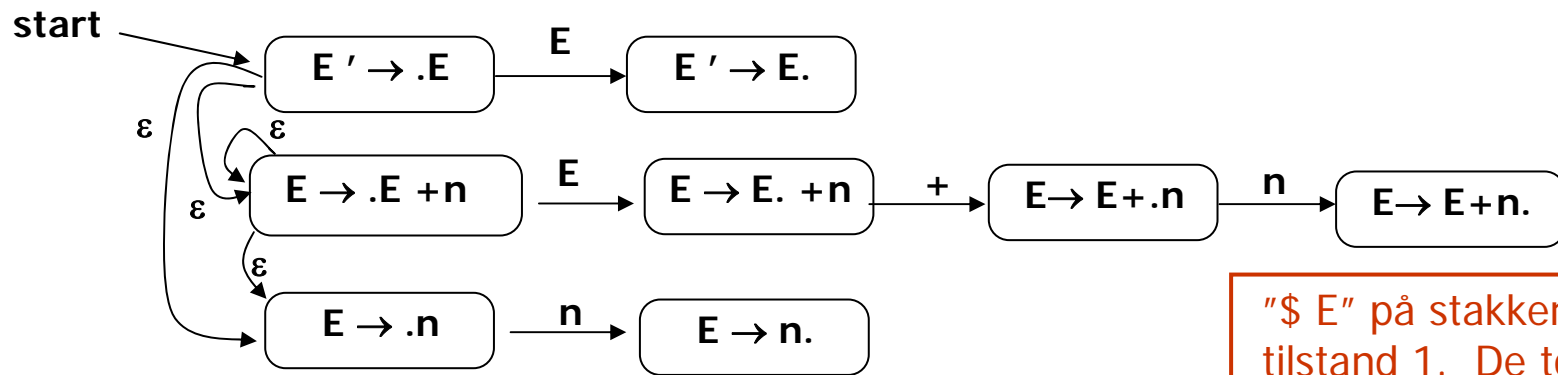
$E \rightarrow .n$

$E \rightarrow n.$

Itemer (LR(0)itemer)



LR(0)-NFA (litt mer systematisk enn boka):

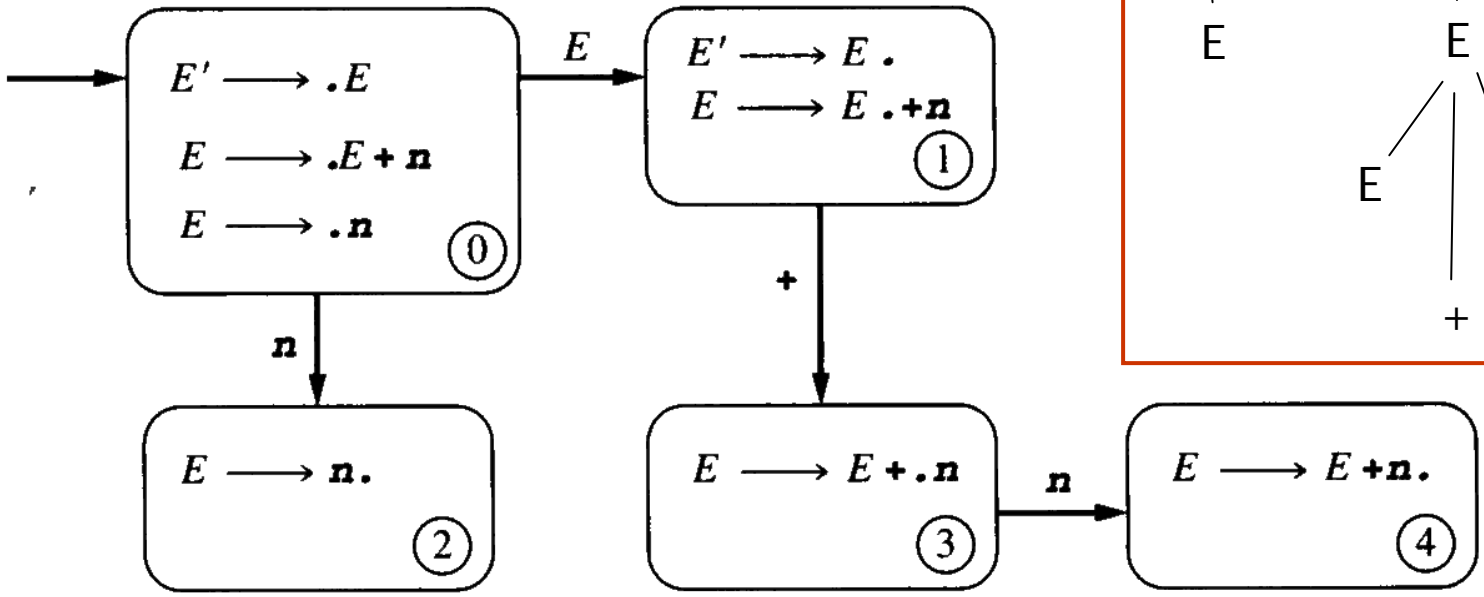


"\$ E" på stakken gir tilstand 1. De to mulighetene kan da eksemplifiseres ved:

The diagram shows two parse trees for the non-terminal E' :

- Tree 1: $E' \rightarrow E$
- Tree 2: $E' \rightarrow E + n$, where E further expands to $E + n$.

LR(0)-DFA, laget ved subset-konstruksjonen (kap 2):



Hvordan sette opp LR(0)-DFA'en direkte fra grammatikken (Rett fram bruk av subset-konstruksjonen)

- Tillukning av item-mengde I:

- Dersom:

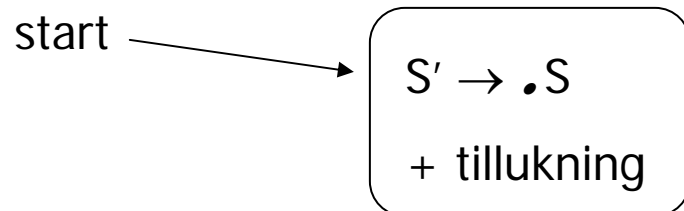
- $A \rightarrow \alpha \cdot B \gamma$ er med i I
 - B er en ikke-terminal
 - $B \rightarrow \beta_1 \mid \beta_2 \mid \dots$

Da er også Itemene:

- $B \rightarrow \cdot \beta_2$.
 - $B \rightarrow \cdot \beta_1$

med i I

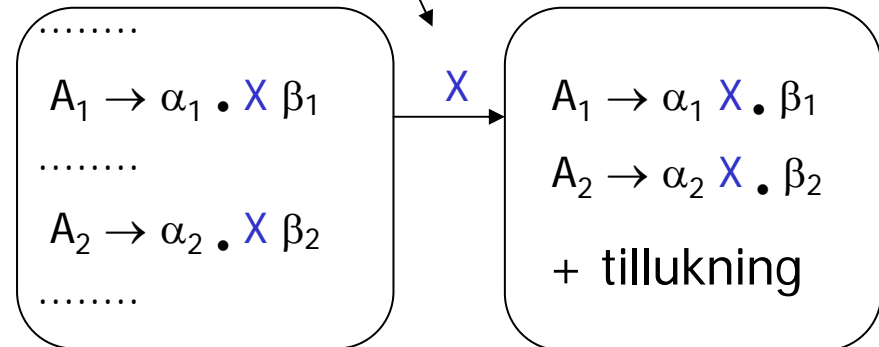
- Start-tilstanden til LR(0)-DFA'en er:



- Tilstandsovergang ved symbol X fra tilstand I

- X er terminal eller ikke-terminal

Tilstand I:



Få med *alle* av formen $A \rightarrow \alpha \cdot X \beta$

Hvordan sette opp LR(0)-DFA'en direkte fra grammatikken - II

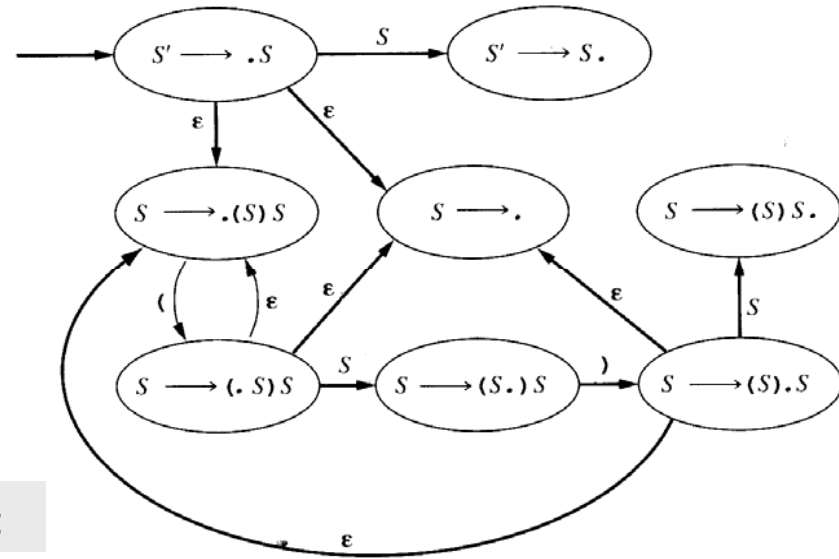
$S' \rightarrow S$
 $S \rightarrow (S) S \mid \epsilon$

Merk at $S \rightarrow \epsilon$ bare gir ett item, nemlig: $S \rightarrow \bullet$

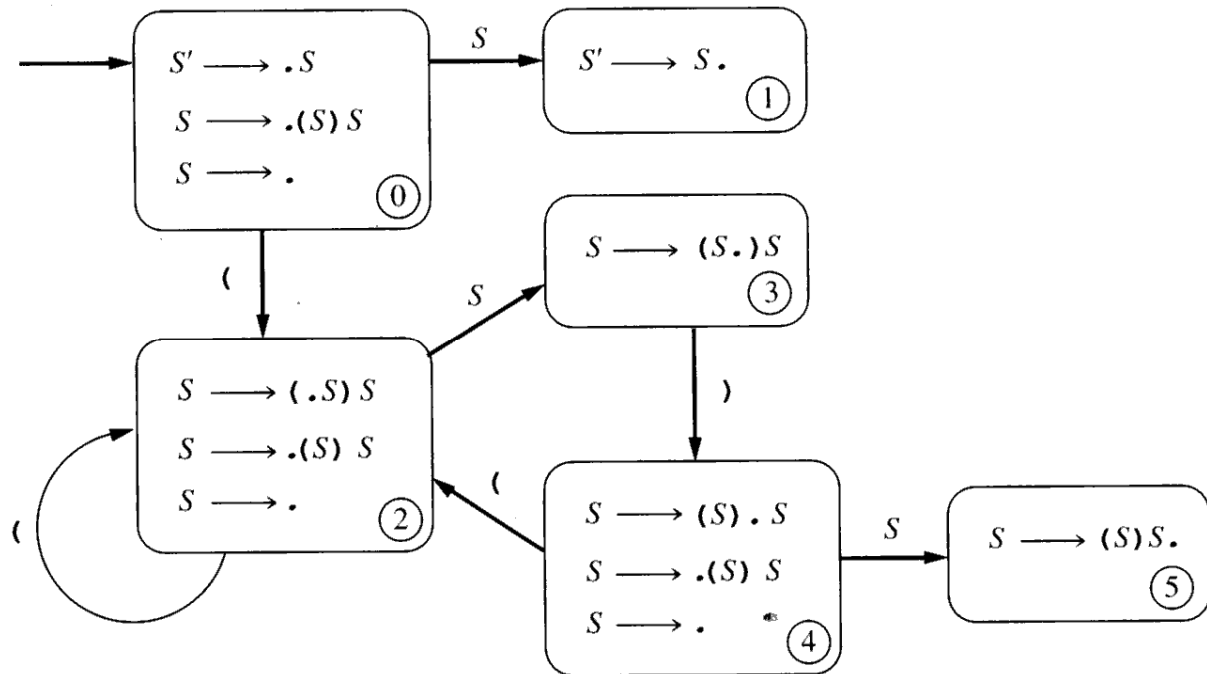
(Altså ikke: ~~$S \rightarrow \bullet \epsilon$~~
 og ~~$S \rightarrow \epsilon \bullet$~~)

- $S' \rightarrow \bullet S$
- $S' \rightarrow S \bullet$
- $S \rightarrow \bullet (S) S$
- $S \rightarrow (\bullet S) S$
- $S \rightarrow (S \bullet) S$
- $S \rightarrow (S) \bullet S$
- $S \rightarrow (S) S \bullet$
- $S \rightarrow \bullet$

LR(0) – NFA:



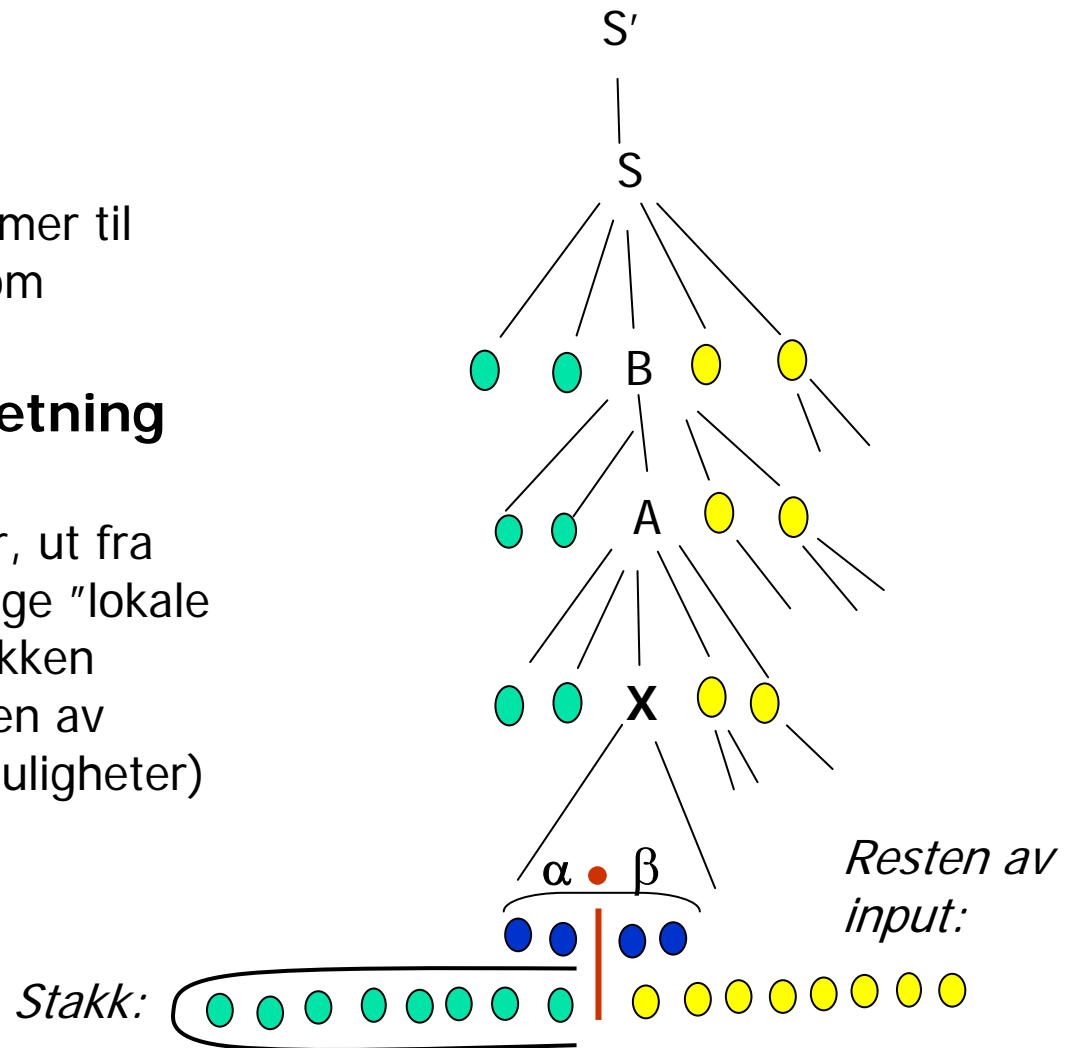
LR(0) – DFA:



Hva sier "topp-tilstanden"?

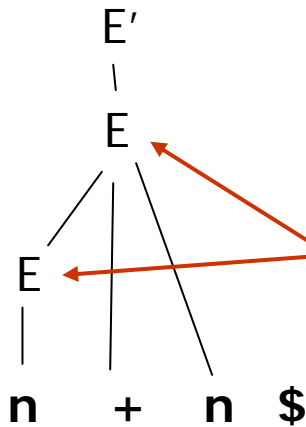
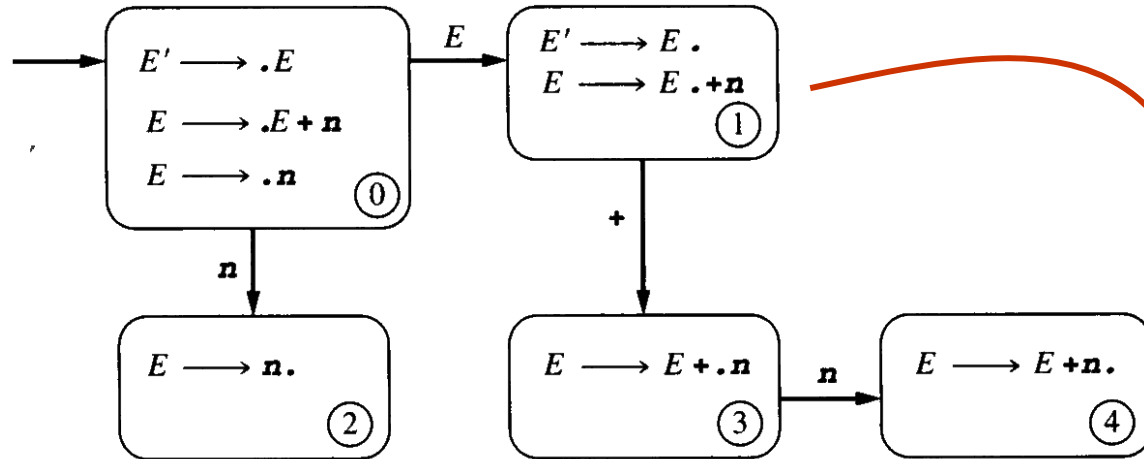
- Topp-tilstanden:
 - Den DFA-tilstanden vi kommer til ved å la stakken gå gjennom DFA'en.
- **LR-parseringens hovedsetning** (litt løselig, bevises ikke):
 - Itemene i topptilstanden er, ut fra det vi har sett, *alle* de mulige "lokale forhold" ved toppen av stakken (Siden vi ikke har sett resten av input kan det være flere muligheter)

Dersom itemet: $X \rightarrow \alpha \bullet \beta$ er med i topp-tilstanden kan situasjonen være som angitt



Hva sier topp-tilstanden? Eksempel:

$E' \rightarrow E$
 $E \rightarrow E + n \mid n$



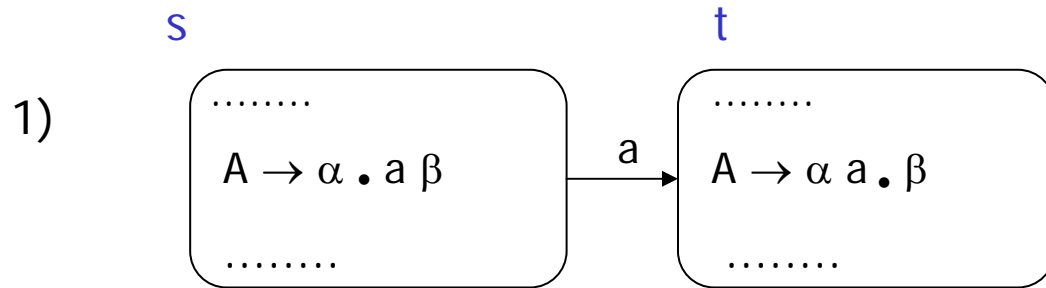
\$	n + n \$
\$ n	+ n \$
\$ E	+ n \$
\$ E +	n \$
\$ E + n	\$
\$ E	\$
\$ E'	\$

Skal skifte

Skal redusere med $E' \rightarrow E$

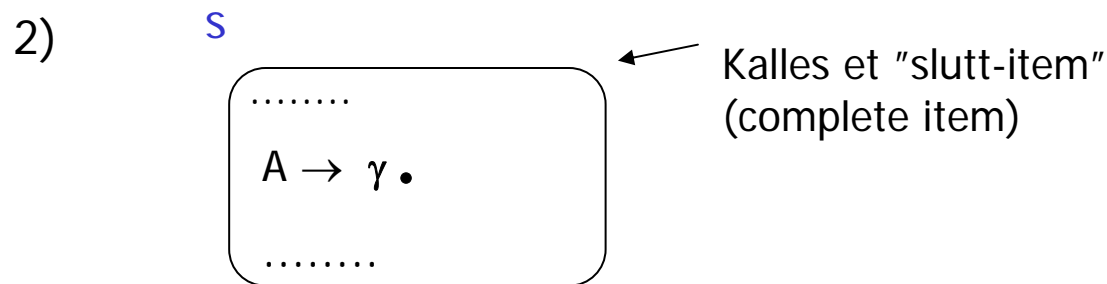
Derved: Denne grammatikken er ikke LR(0): For stakk "E" er neste skritt ubestemt 19

LR(0)-grammatikker og LR(0)-algoritmen:



Angir at neste skritt *kan* være skift, og at dette er lovlig om neste input er a (ny topptilstand blir t)

N.B : s, t, u, v, w, z står for nummere på tilstander fra DFAen. Disse ligger på stakken sammen med det vi foreløpig har skiftet inn fra input og evt. redusert dette til.

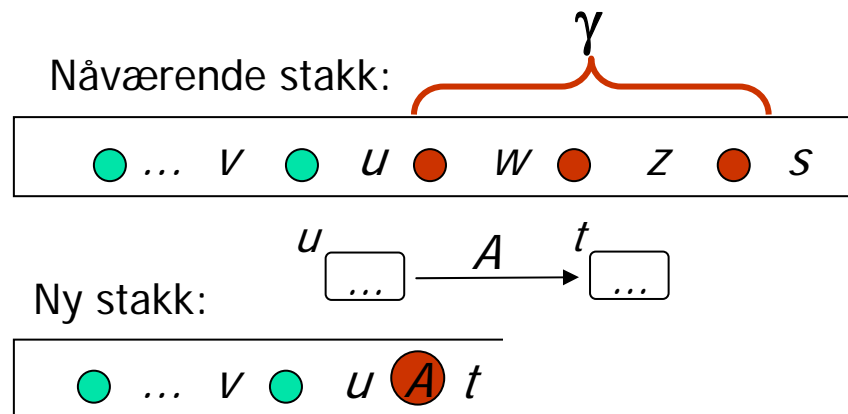


Angir at neste skritt *kan* være reduksjon $A \rightarrow \gamma$

LR(0) –grammatikk:
Dersom dette gir entydige aksjoner for alle tilstander (*kan=må*), er grammatikken LR(0)!

Da har vi en grei algoritme!

Vi holder tilsandene mellom stakk-symbolene



Reduser-steget: Pop av det som tilsvarer γ (og mellomliggende tilstander), og push på A , og finn riktig ny topptilstand ut fra u og A

Er eksempel-grammatikkene i Kap 5 LR(0) ?

Ser på tre grammatikker:

$$A \rightarrow (A) \mid a$$

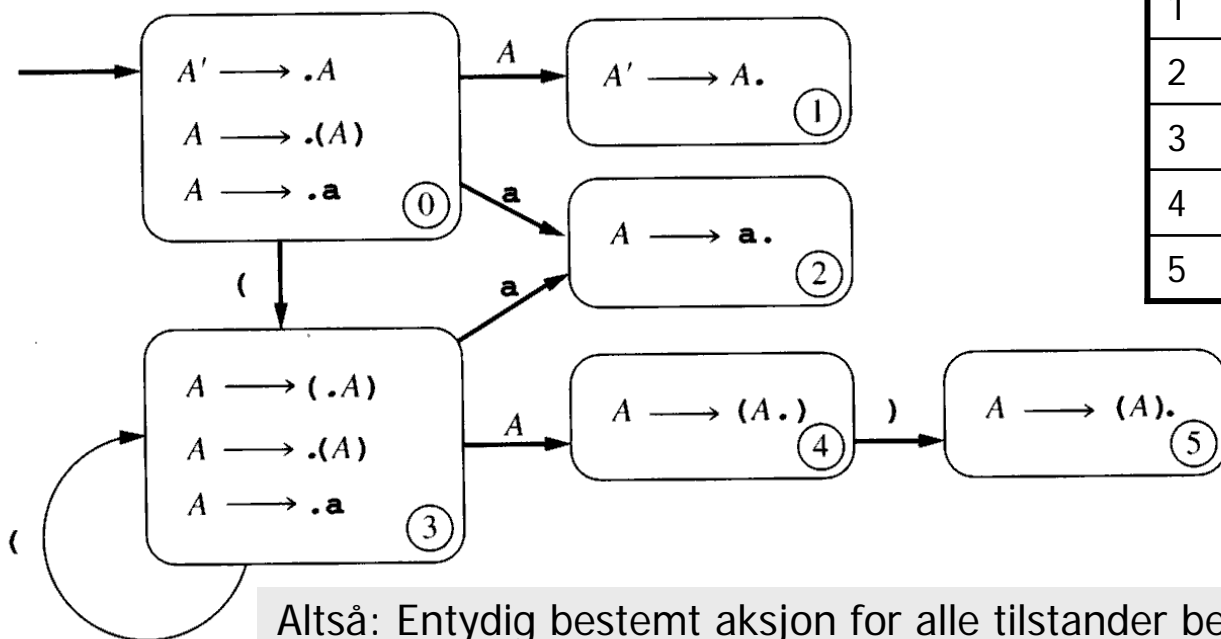
$$S' \rightarrow S$$

$$S \rightarrow (S) S \mid \varepsilon$$

$$E' \rightarrow E$$

$$E \rightarrow E + n \mid n$$

A gir følgende LR(0) – DFA:



Tilst.	Mulig aksjoner:
0	Bare skift mulig, for "(","a"
1	Bare red. mulig, med $A' \rightarrow A$
2	Bare red. mulig, med $A \rightarrow a$
3	Bare skift mulig, for "(","a"
4	Bare skift mulig, for ")"
5	Bare red. mulig, med $A \rightarrow (A)$

Altså: Entydig bestemt aksjon for alle tilstander betyr: **Grammatikken er LR(0)**

MERK: Der det er reduksjon må det ikke være tvil om med hvilken produksjon!

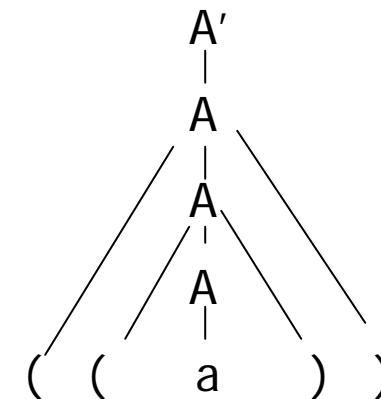
Tabell-oppsett for en LR(0) - grammatikk:

State	Action	Rule	Input			Goto
			(a)	
0	shift		3	2		1
1	reduce	$A' \rightarrow A$				4
2	reduce	$A \rightarrow a$				
3	shift		3	2		4
4	shift				5	
5	reduce	$A \rightarrow (A)$				

Hvis en reduksjon bringer oss tilbake til tilstand 0 eller 3, sier Goto hvilken tilstand A gir.

Parsering av setningen: ((a))

	Parsing stack	Input	Action
1	\$ 0	((a)) \$	shift
2	\$ 0 (3	(a)) \$	shift
3	\$ 0 (3 (3	a)) \$	shift
4	\$ 0 (3 (3 a 2)) \$	reduce $A \rightarrow a$
5	\$ 0 (3 (3 A 4)) \$	shift
6	\$ 0 (3 (3 A 4) 5) \$	reduce $A \rightarrow (A)$
7	\$ 0 (3 A 4) \$	shift
8	\$ 0 (3 A 4) 5	\$	reduce $A \rightarrow (A)$
9	\$ 0 A 1	\$	accept



Skal her redusere med $A' \rightarrow A$, og input tom: Ferdig

Parsering av noen gale strenger for: $A \rightarrow (A) \mid a$

State	Action	Rule	Input			Goto
			(a)	
0	shift		3	2		1
1	reduce	$A' \rightarrow A$				
2	reduce	$A \rightarrow a$				
3	shift		3	2		4
4	shift				5	
5	reduce	$A \rightarrow (A)$				

\$ 0	((a) \$
\$ 0 (3	(a) \$
\$ 0 (3	(a) \$
\$ 0 (3 (3	a) \$
\$ 0 (3 (3 a) \$
\$ 0 (3 (3 A) 5	\$
\$ 0 (3 (3 A 4	\$

\$ 0	() \$
\$ 0 (3) \$

Viktig: Skifter
aldri noe ulovlig
inn på stakken!

En (litt ruskete) formulering av LR(0)-kravet: Dersom følgende gir en entydig algoritme er grammatikken LR(0):

s er en DFA-tilstand med flere itemer

The LR(0) parsing algorithm. Let s be the current state (at the top of the parsing stack). Then actions are defined as follows:

1. If state s contains any item of the form $A \rightarrow \alpha.X\beta$, where X is a terminal, then the action is to shift the current input token onto the stack. If this token is X , and state s contains item $A \rightarrow \alpha.X\beta$, then the new state to be pushed on the stack is ~~the state containing the item $A \rightarrow \alpha.X\beta$~~ t , where $s \xrightarrow{X} t$. If this token is not X for some item in state s of the form just described, an error is declared.
2. If state s contains any complete item (an item of the form $A \rightarrow \gamma$), then the action is to reduce by the rule $A \rightarrow \gamma$. A reduction by the rule $S' \rightarrow S$, where S' is the start state, is equivalent to acceptance, provided the input is empty, and error if the input is not empty. In all other cases, the new state is computed as follows. Remove the string γ and all of its corresponding states from the parsing stack (the string γ must be at the top of the stack, according to the way the DFA is constructed). Correspondingly, back up in the DFA to the state from which the construction of γ began (this must be the state ~~u~~ uncovered by the removal of γ). Again, by the construction of the DFA, this state ~~u~~ must contain an item of the form $B \rightarrow \alpha.A\beta$. Push A onto the stack, and push (as the new state) the state ~~containing the item $B \rightarrow \alpha.A\beta$~~ t , where $u \xrightarrow{A} t$. (Note that this corresponds to following the transition on A in the DFA, which is indeed reasonable, since we are pushing A onto the stack.)

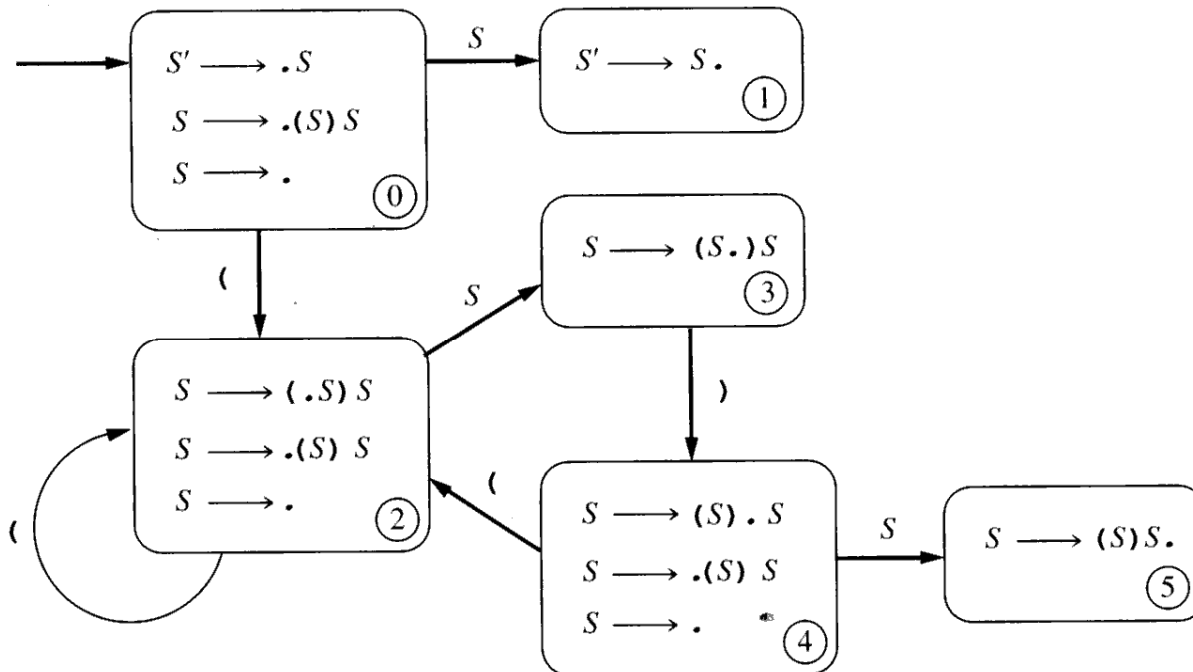
Kan forekomme i flere tilstander

Avslutning

Er denne grammatikken LR(0)?
 Nei, pga. tilstandene 0, 2 og 4

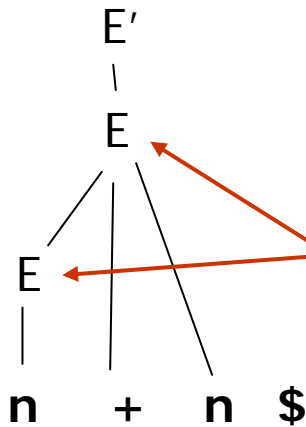
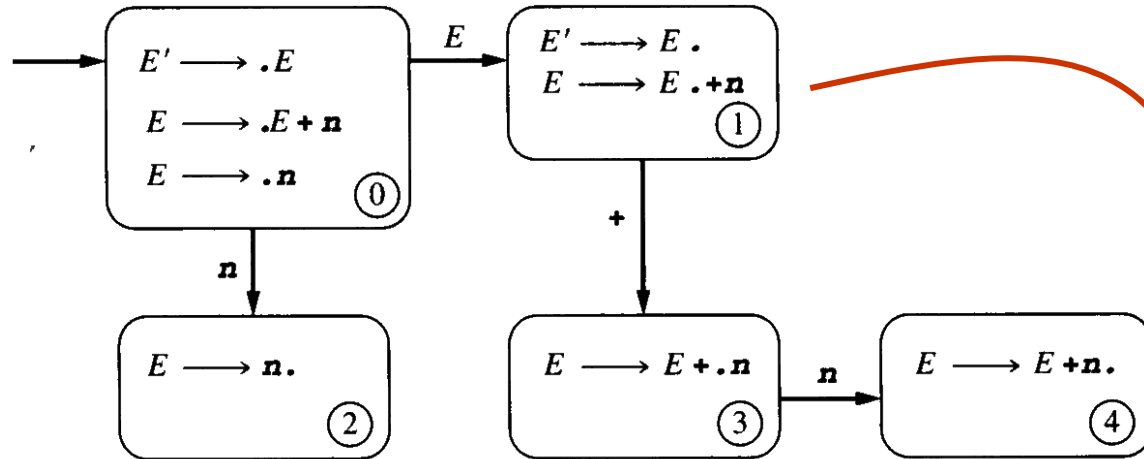
$S' \rightarrow S$
 $S \rightarrow (S) S \mid \varepsilon$

LR(0) – DFA:



Er den LR(0)? Nei, pga. tilstand 1:

$E' \rightarrow E$
 $E \rightarrow E + n \mid n$



\$	n + n \$
\$ n	+ n \$
\$ E	+ n \$
\$ E +	n \$
\$ E + n	\$
\$ E	\$
\$ E'	\$

Skal skifte

Skal redusere med $E' \rightarrow E$

Derved: Denne grammatikken er ikke LR(0): For stakk "E" er neste skritt ubestemt 26