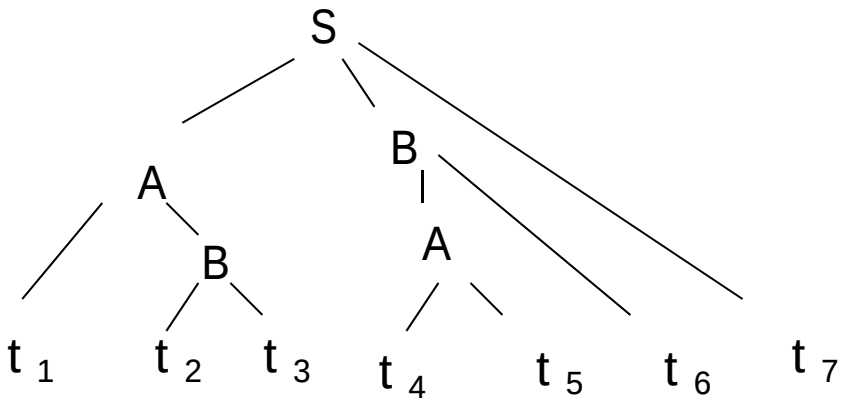
Kap. 5, del 1: Parsering nedenfra-opp (Bottom-up parsering)
INF5110

16/2-2011

Stein Krogdahl
Ifi, UiO

”Bottom up” parsering (nedenfra-og-opp)



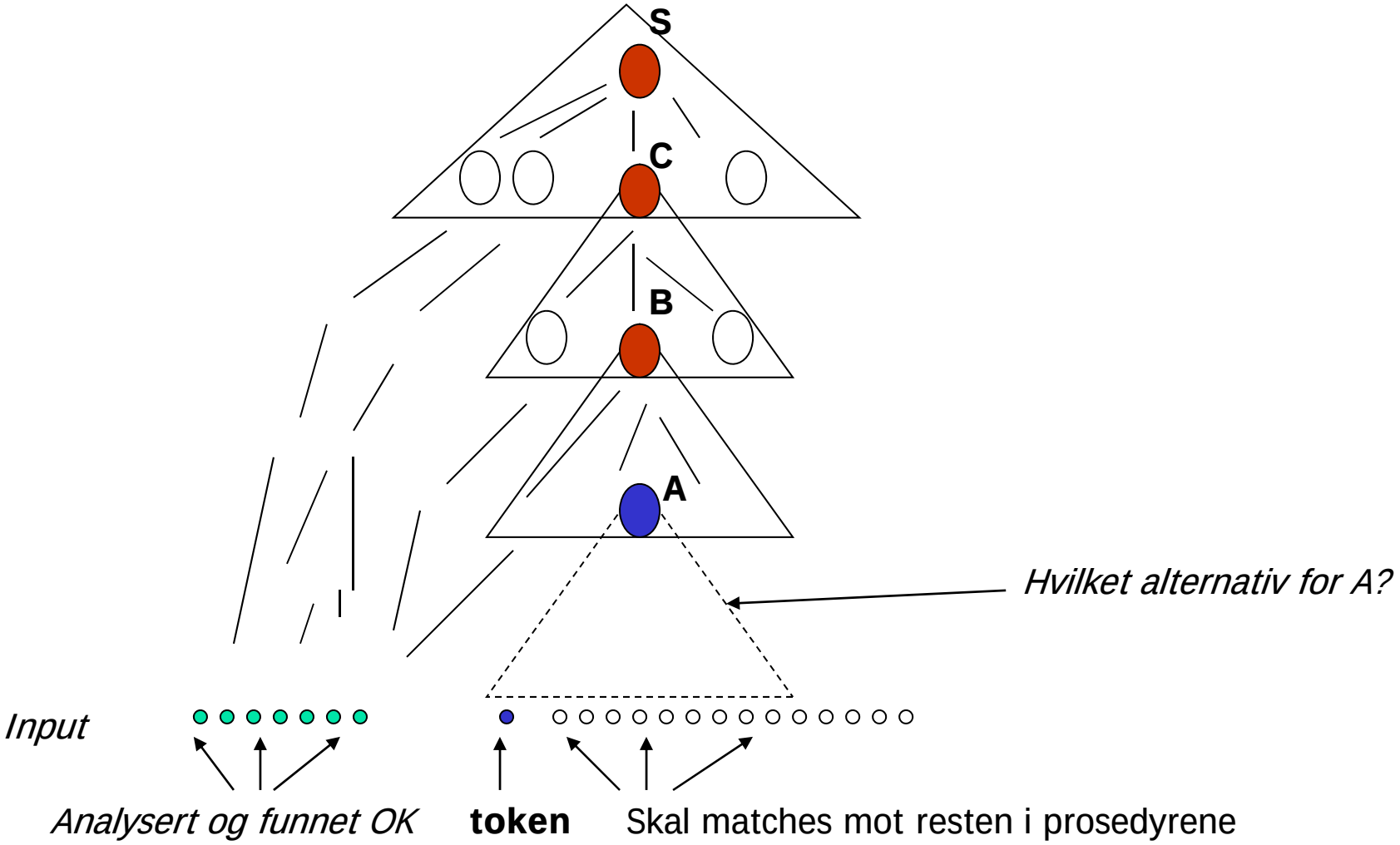
LR-parsering og grammatikker: Metodene angitt under kan i rekkefølge ta ”vanskeligere og vanskeligere” grammatikker

- 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

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

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

Til sammenlikning: ”Top down”-parsering



Datastrukturen for LR-parsering

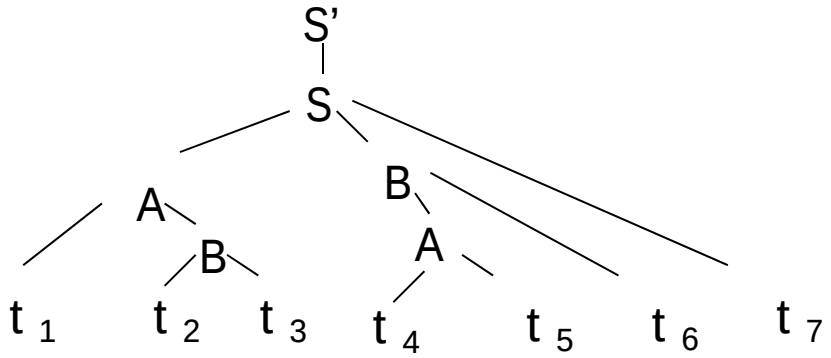
$S' \rightarrow S$

$S \rightarrow A\ B\ t_7 \mid \ldots.$

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

$B \rightarrow t_2\ t_3 \mid A\ t_6 \mid \ldots.$

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

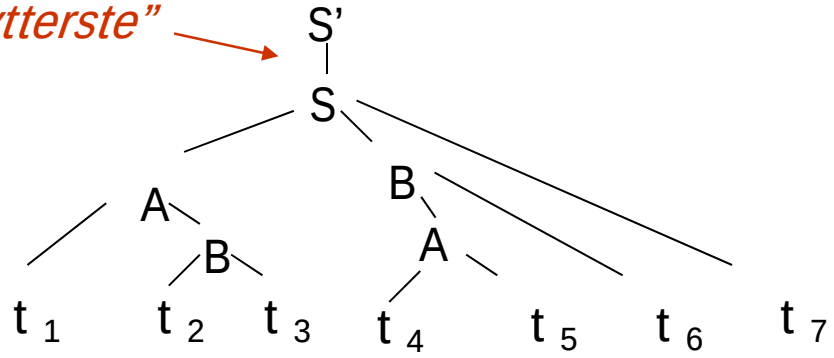
$S' \rightarrow S$

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

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

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

Sett på ny "ytterste" produksjon



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

- Ha en "stakk" for det som er lest (og "redusert"!)
- 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:

$	t_1 t_2 t_3 t_4 t_5 t_6 t_7 $
stakk	input

Slutt-situasjonen:

$ S'	$
stakk	input

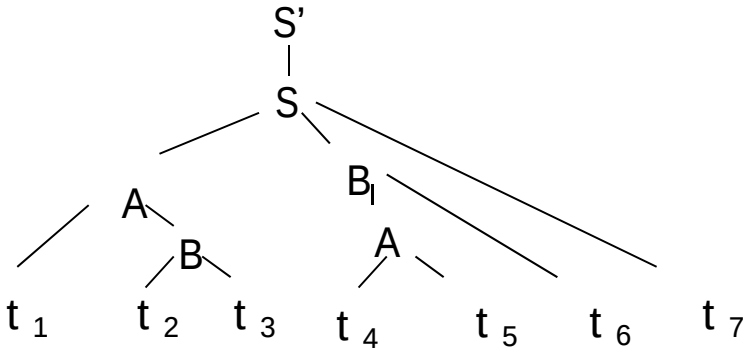
Prinsippet for LR-parsering og opersjonene: skift og redusèr

S' → S

S → A B t_7 | ....

A → t_4 t_5 | t_1 B |

B → t_2 t_3 | A t_6 | ....



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	$

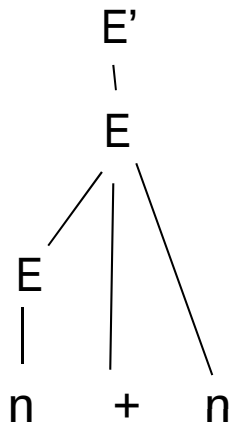
- Det blir to typer steg:
  - *Skift* ("les") neste input over til stakken
  - *Reduksjon* på toppen av stakken med bestemt regel A → α
  - Dersom man kjenner syntakstreet for den aktuelle setning, er det lett å angi de rette stegene.
- MEN: Hvordan gjøre dette underveis *uten* å kjenne resten av input og det fulle syntakstre??

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 (når vi kjenner treet!)

$E' \rightarrow E$

$E \rightarrow E + n \mid n$



I boka, men *vi* stresser det ikke:

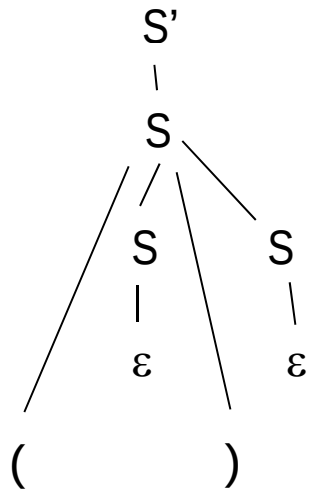
- Den neste reduksjonen som skal gjøres, kalles situasjonens "handle" (håndtak).
- "stakk + input" utgjør et stadium i en høyre-avledning, og de kommer i omvendt rekkefølge av en høyre-avledning

Høyre-avledning

	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

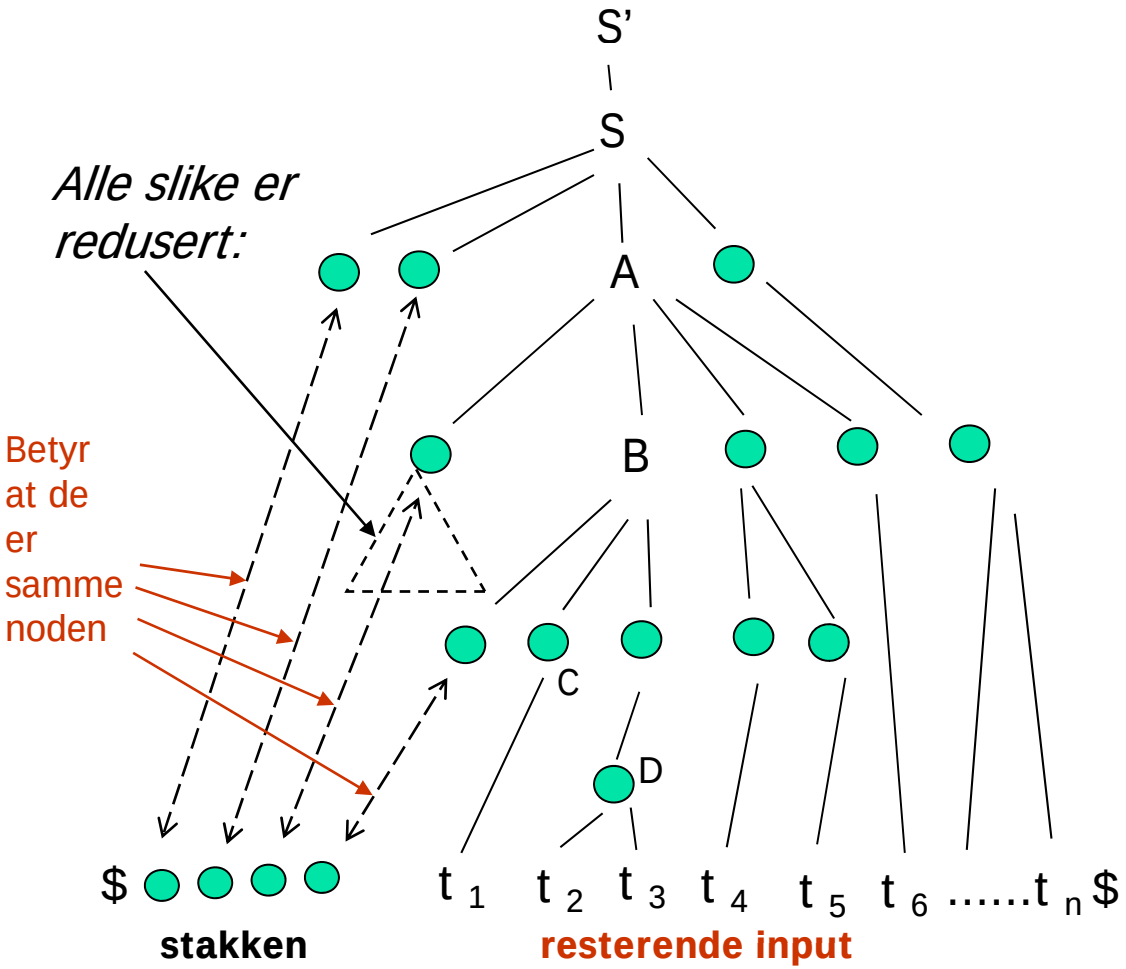
$S' \rightarrow S$
$S \rightarrow (\ S \ ) \ \ S \mid \varepsilon$



NB: S → ε blir litt ”rar”:
Ved reduksjon med denne dukker en ikke-termial bare opp på enden av stakken, her og her.

	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

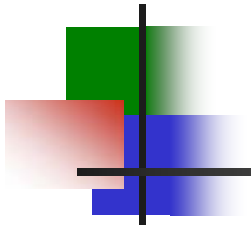


Det neste reduksjonen som blir gjort er reduksjon med

C -> t1

Deretter (etter noen skift):

D -> t2 t3



Gitt en entydig grammatikk G.

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

(Ikke alt er beskrevet i boka, og bare det i boka er pensum)

- Vi ser på *alle* mulig stakker som kan opptre, og ser disse som strenger over alfabetet: {terminaler}∪{ikke-terminaler} . Vi ser altså på:

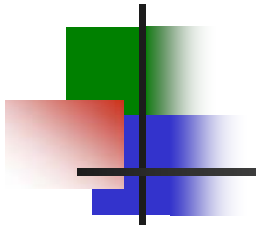
  {S | S utgjør stakken på ett eller annet stadium i LR-parsering av en setning i L(G)}

- 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 DFA-en er mengder av NFA-tilstander, og vil her 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 ennå ikke har sett på resten av input.



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

- For produksjonen :

  $A \rightarrow X\ Y\ Z$

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

  $A \rightarrow \bullet X\ Y\ Z$

  $A \rightarrow X \bullet Y\ Z$

  $A \rightarrow X\ Y \bullet Z$

  $A \rightarrow X\ Y\ Z \bullet$

- Punktumet angir grovt sett hvor vi er hen i ”dypeste aktive del” av parseringen i øyeblikket:
  - Det til venstre for punktum er lest 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

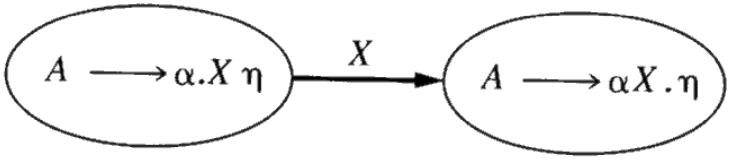
Kantene i NFA-en I

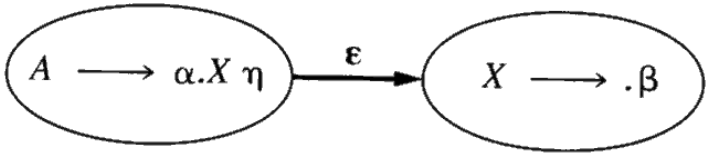
Dette er ikke fullt forklart i boka, og er ikke pensum

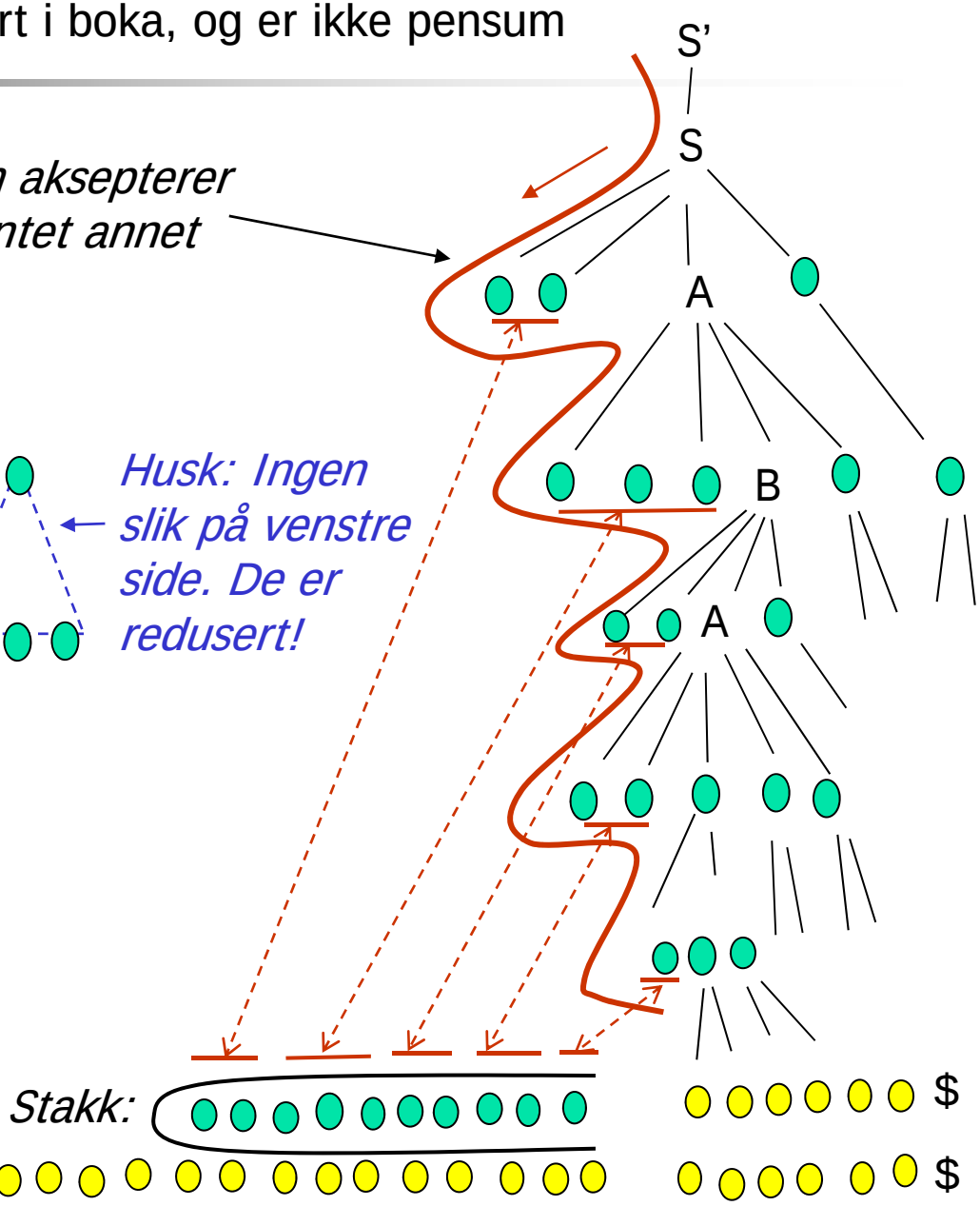
Skal lage en NFA som aksepterer alle slike "linjer", og intet annet

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!

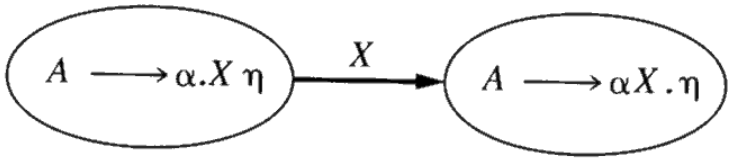




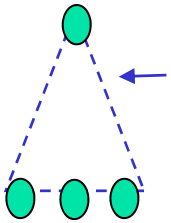


Kantene i NFA-en II

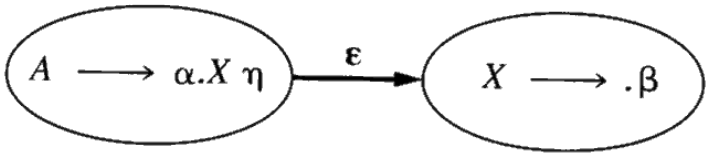
Dette er ikke fullt forklart i boka, og er ikke pensum



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

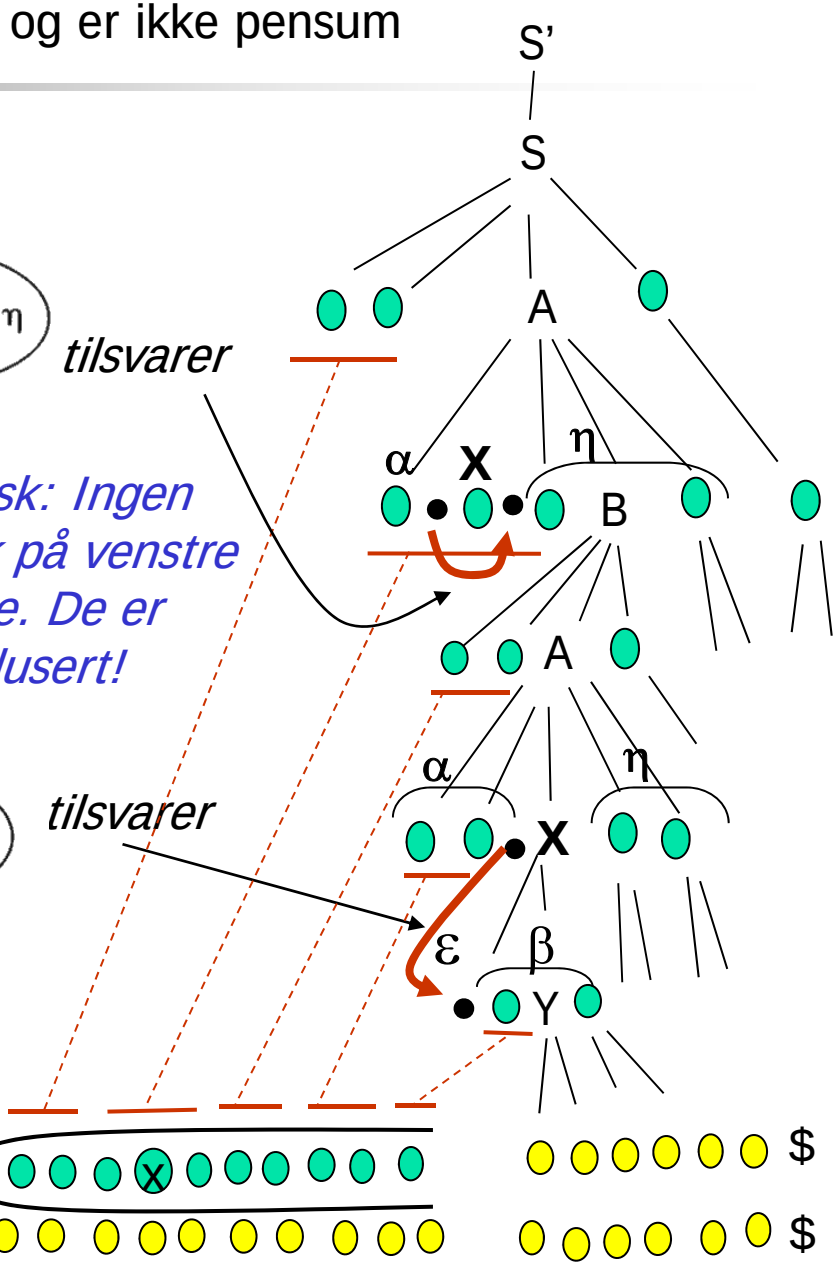


Husk: Ingen slik på venstre side. De er redusert!



tilsvarer

tilsvarer



Stakk:

Opprinnelig input:

NFA'en som beskriver alle mulige stakker

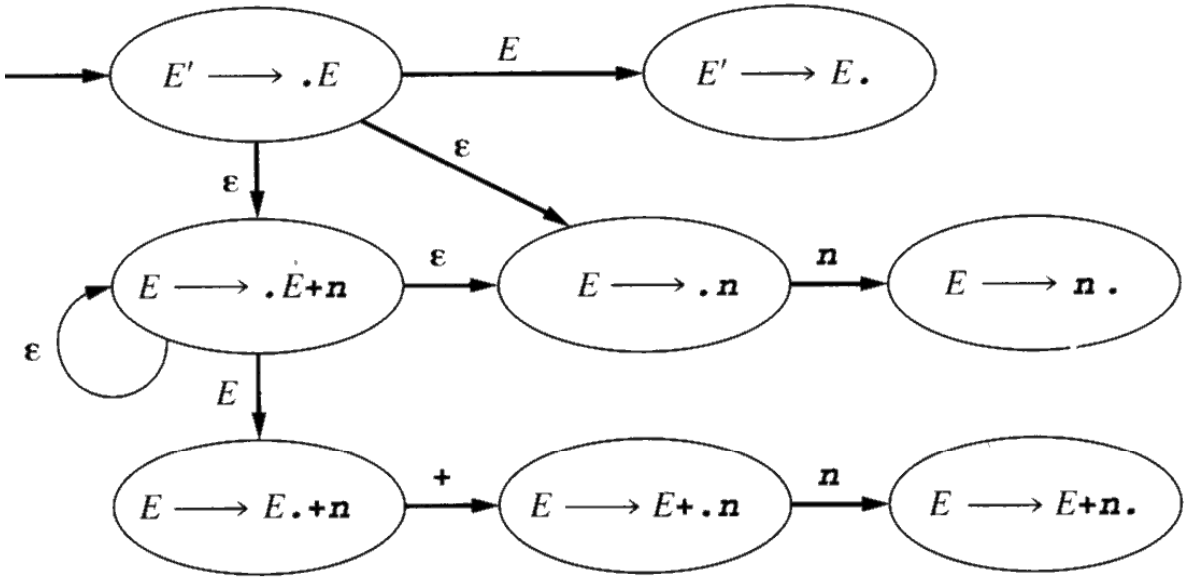
E ' → E

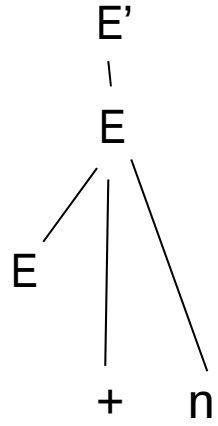
E → E + n

E → n

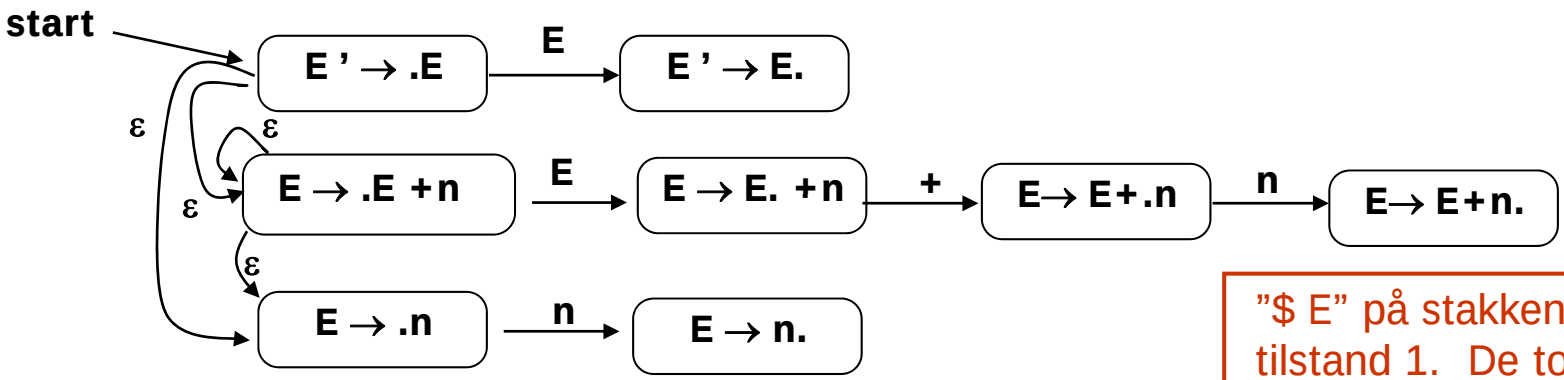
$E' \rightarrow .E$
$E' \rightarrow E.$
$E \rightarrow .E + n$
$E \rightarrow E.+ n$
$E \rightarrow E + .n$
$E \rightarrow E + n.$
$E \rightarrow .n$
$E \rightarrow n.$

Itemer (LR(0)itemer)

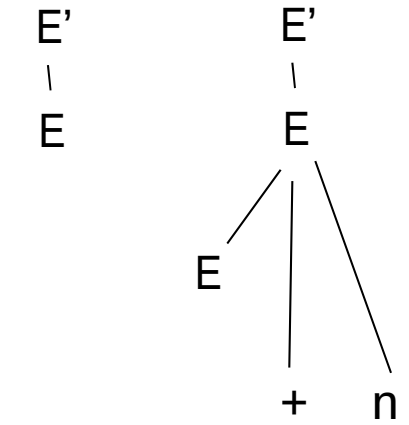




LR(0)-NFA (tegnet litt mer systematisk enn i boka):



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



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. Bør trenes på!)

- Tillukning av item-mengde I:
  - Dersom:
    - $A \rightarrow \alpha \bullet B\gamma$ er med i I
    - B er en ikke-terminal
    - $B \rightarrow \beta_1 \mid \beta_2 \mid \ldots$

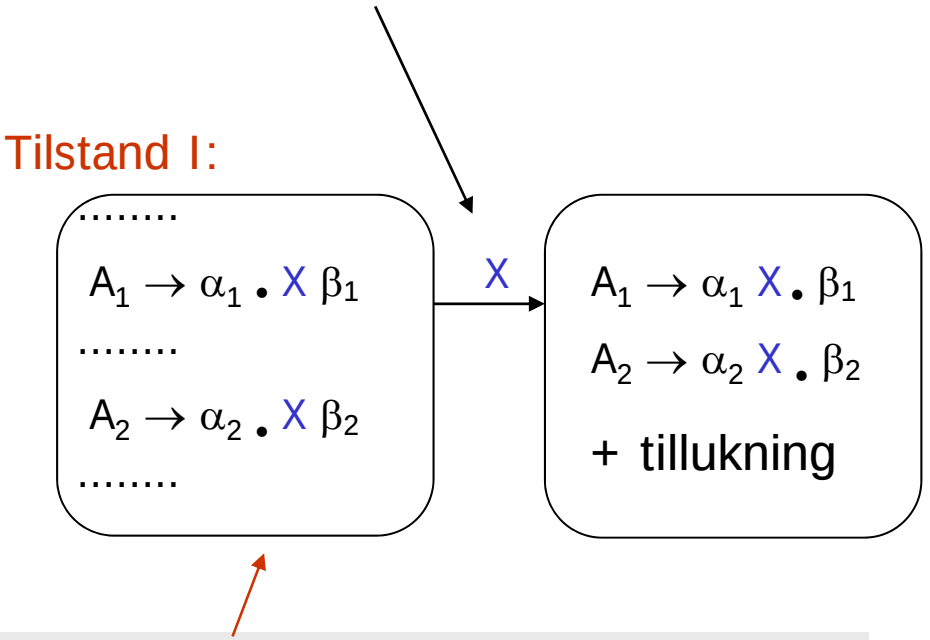
  Da er også Itemene:
    - $B \rightarrow \bullet \beta_2$
    - $B \rightarrow \bullet \beta_1$
    - ...

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

  $S' \rightarrow \bullet S$

  + tillukning

- Tilstandsovergang ved symbol X fra tilstand I
  - X er terminal eller ikke-terminal (behandles likt!)



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

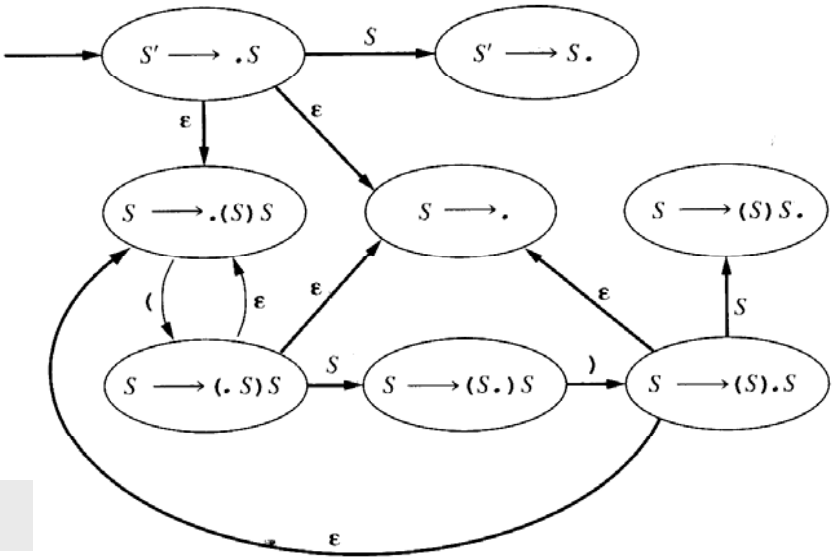
$$S' \rightarrow S$$
$$S \rightarrow (S) \quad S \mid \varepsilon$$

Merk at S → ε bare gir ett item, nemlig: S → •

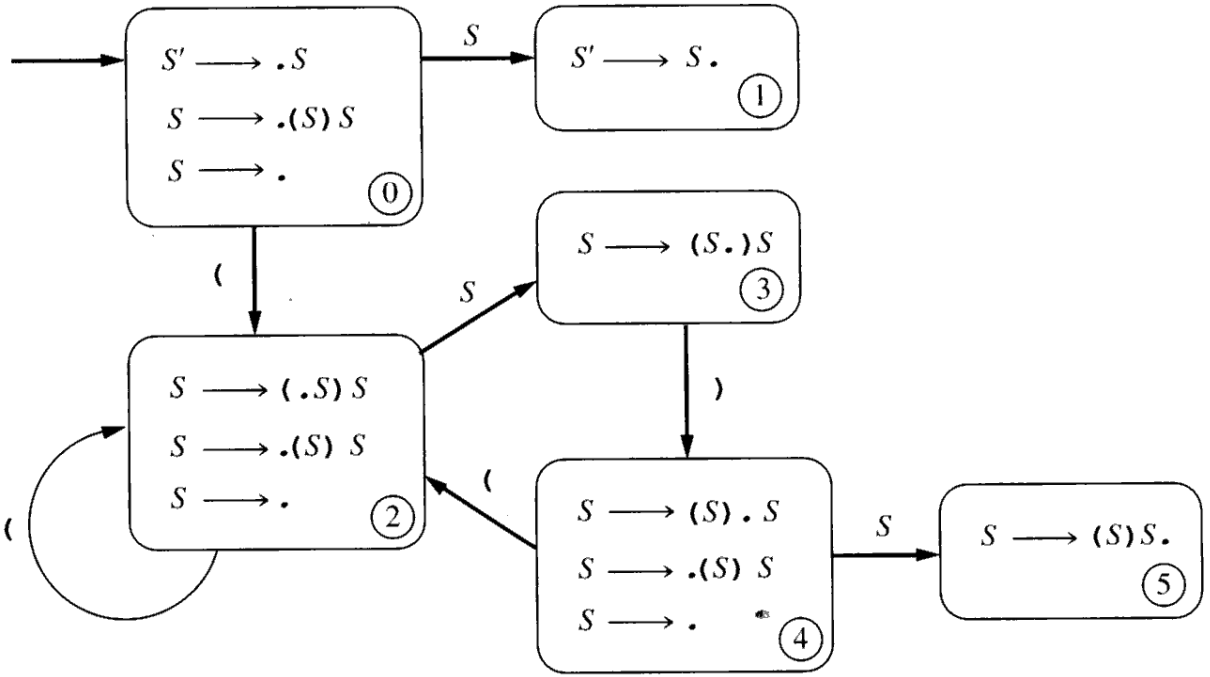
(Altså ikke: ~~S → • ε~~ og ~~S → ε •~~)

$$S' \rightarrow .S$$
$$S' \rightarrow S.$$
$$S \rightarrow .(S)S$$
$$S \rightarrow (.S)S$$
$$S \rightarrow (S.)S$$
$$S \rightarrow (S).S$$
$$S \rightarrow (S)S.$$
$$S \rightarrow .$$

LR(0) – NFA:



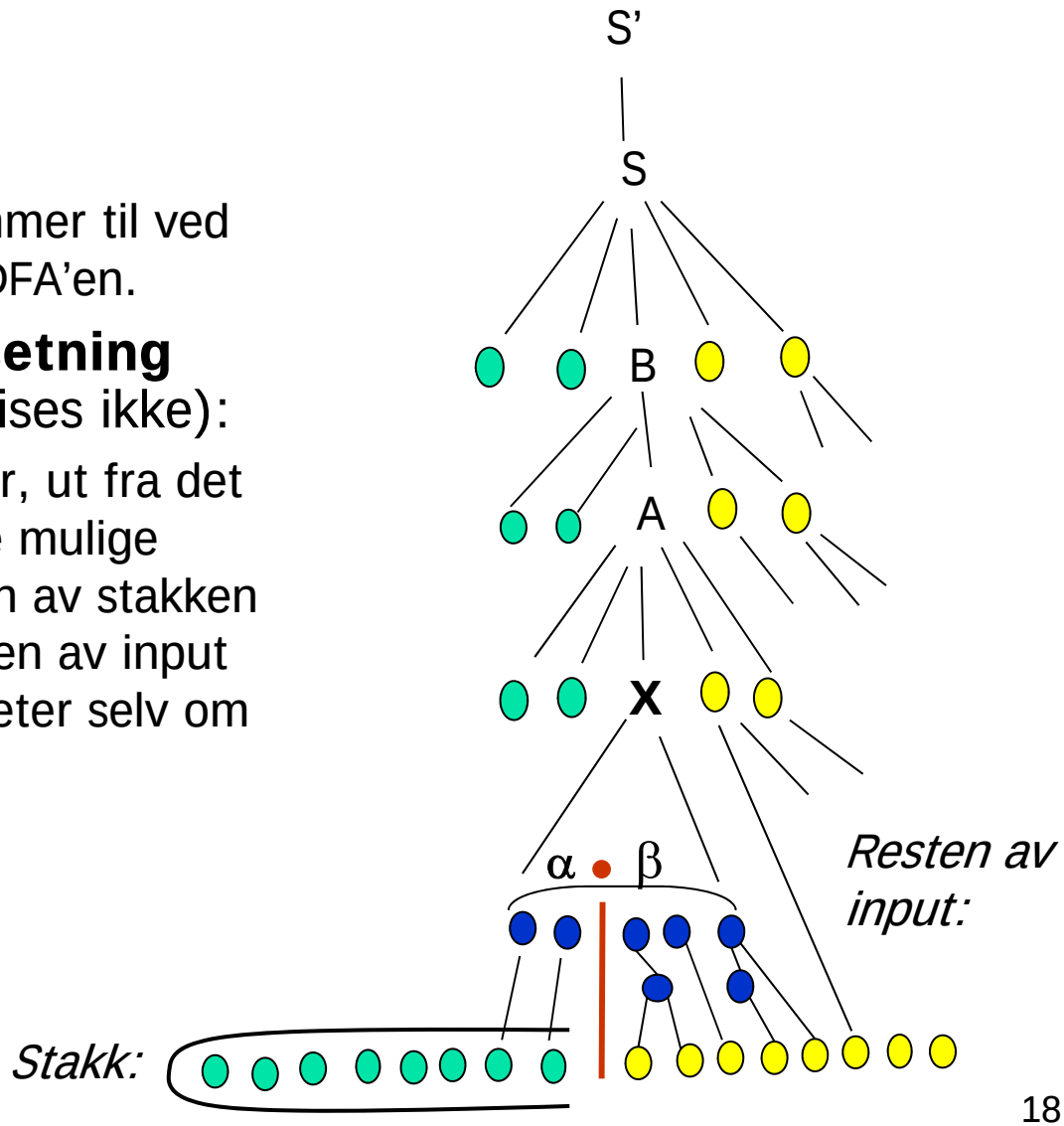
LR(0) – DFA:



Hva sier ”topp-tilstanden”?

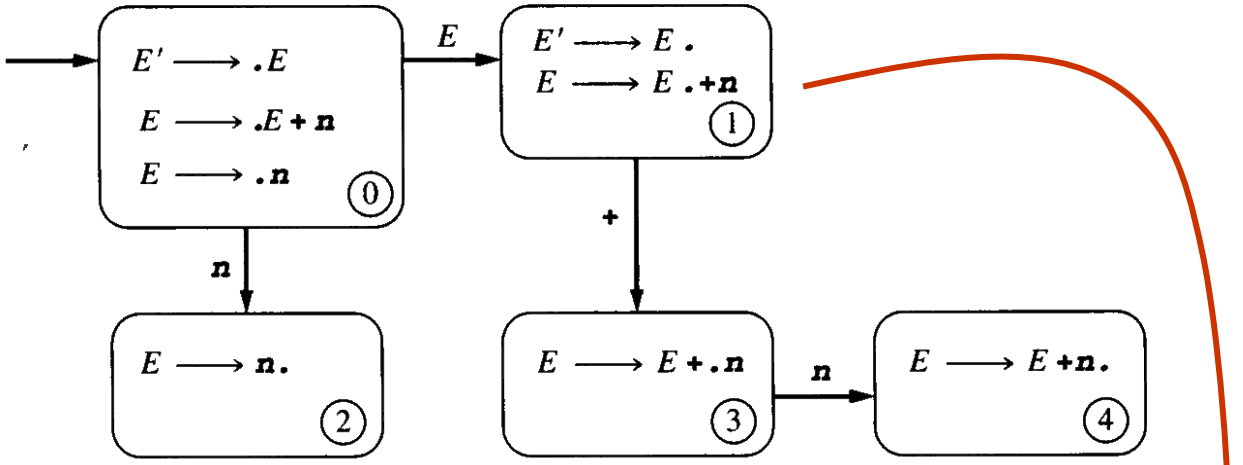
- **Topp-tilstanden:**
  - Den DFA-tilstanden vi kommer til ved å la stakken gå gjennom DFA’en.
- **LR-parseringens hovedsetning** (En gang til, litt løselig, bevises ikke):
  - Itemene i topptilstanden er, ut fra det vi har lest av input, *alle* de mulige ”lokale forhold” ved toppen av stakken (Siden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

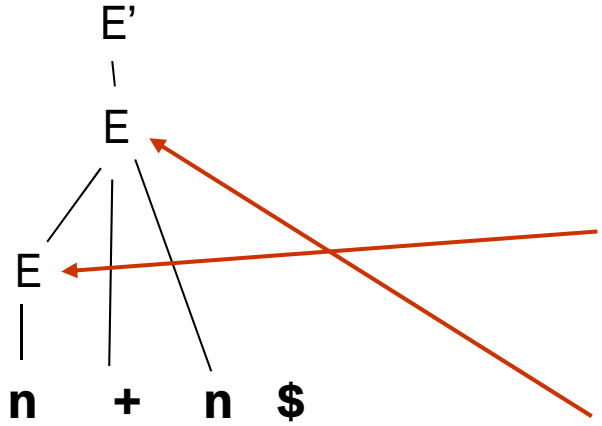
Dersom itemet $X \rightarrow \alpha \bullet \beta$ er med i topp-tilstanden kan situasjonen altså være som angitt til høyre.



Hva sier topp-tilstanden? Eksempel:

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



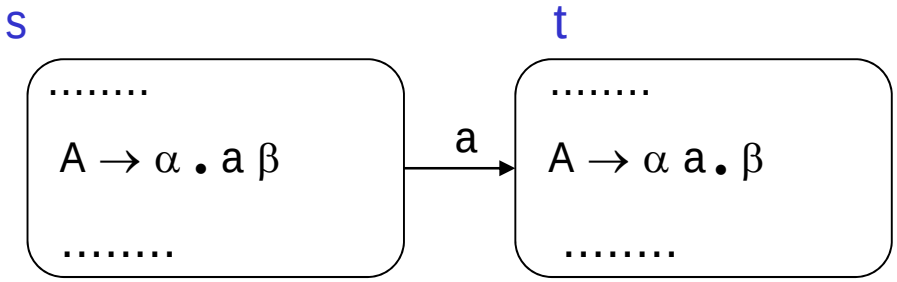


$	n + n $
$ n	+ n $
$ E	+ n $
Skal skifte	
$ E +	n $
$ E + n	$
$ E	$
Skal redusere med E'→E	
$ E'	$

LR(0)-grammatikk: Topptilstanden bestemmer alltid entydig neste steg.
Derved: Denne grammatikken er ikke LR(0). For stakk ”E” er neste skritt ubestemt.

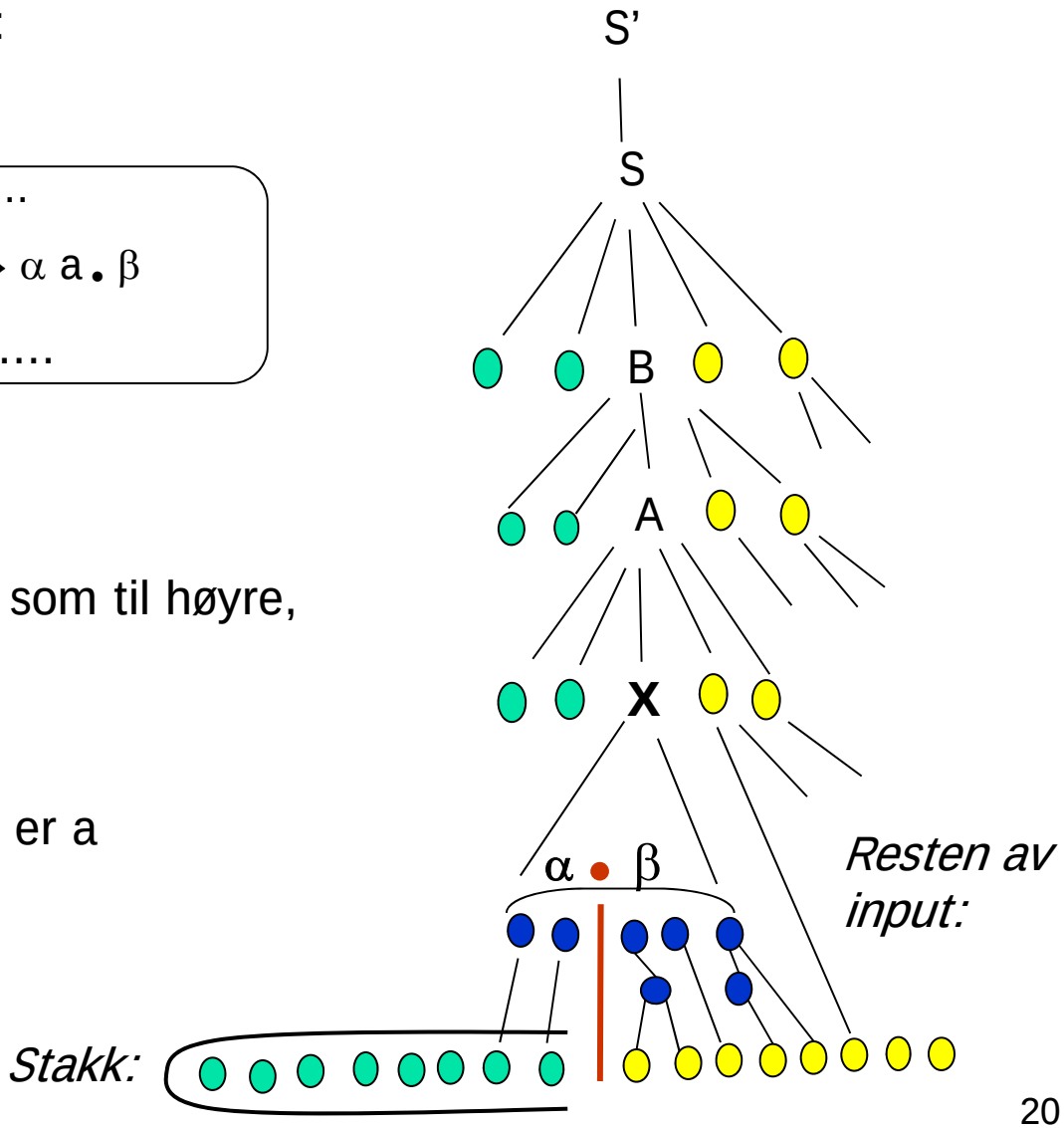
Når er ”skift” en mulighet?

Anta vi er i tilstand s under:



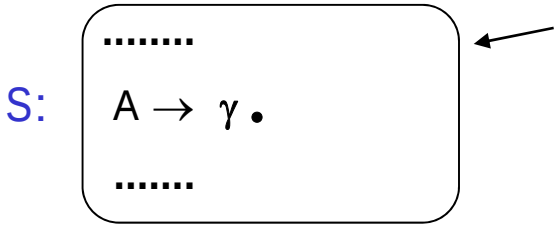
Angir at situasjonen *kan* være som til høyre, og derved:

- Neste skritt kan være skift
- Skift er lovlig om neste input er a
- Ny topptilstand blir i så fall t



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

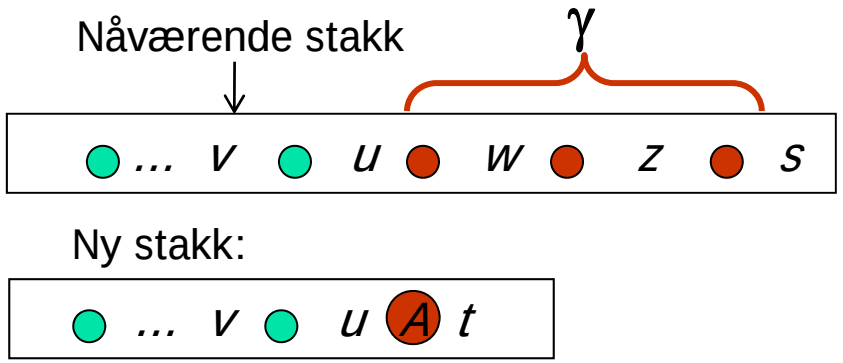
Anta at topptilstanden er s, og at den er slik:



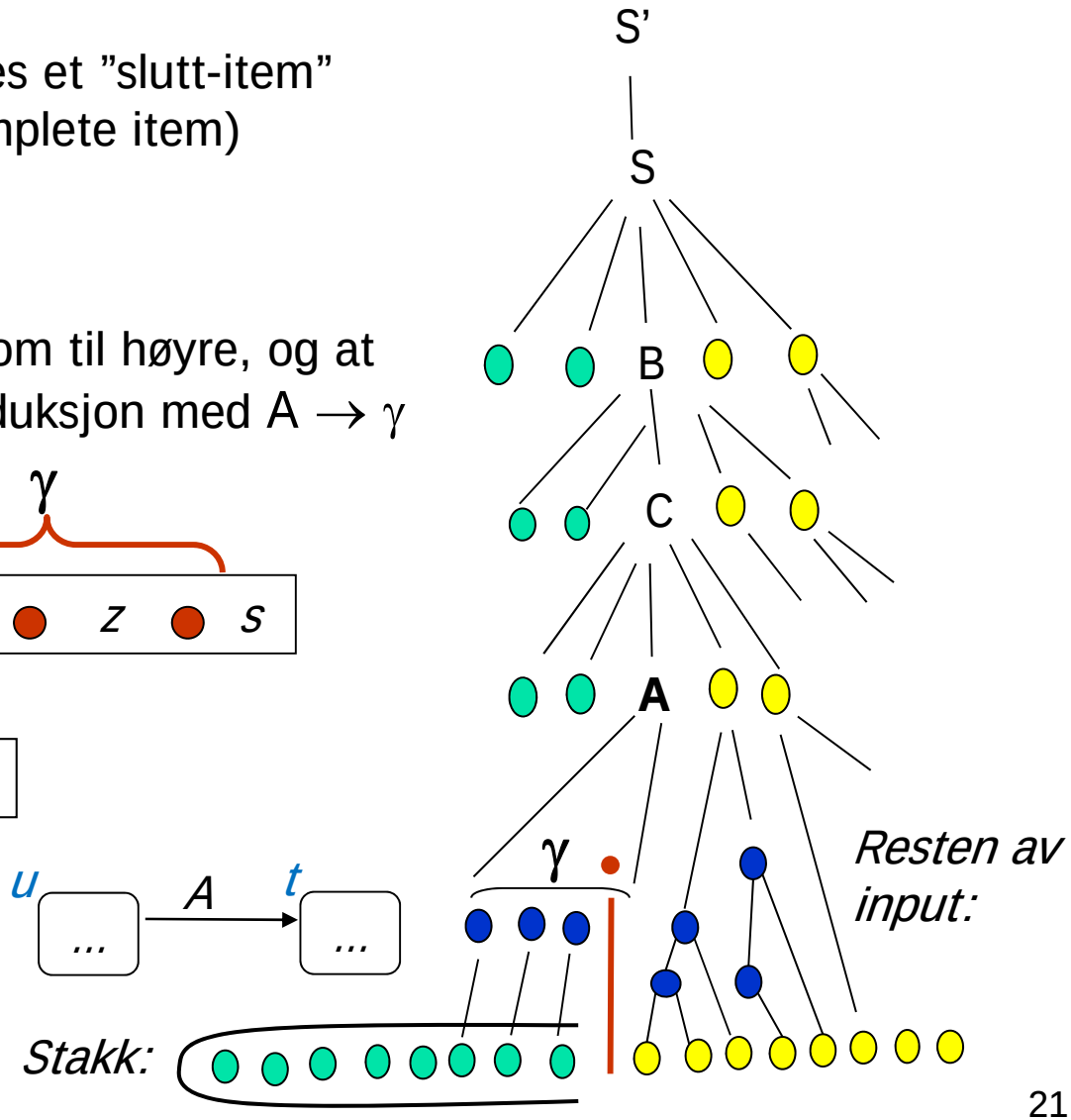
Kalles et "slutt-item" (complete item)

Angir at situasjonen *kan* være som til høyre, og at neste skritt derved kan være reduksjon med A → γ

NB: 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:

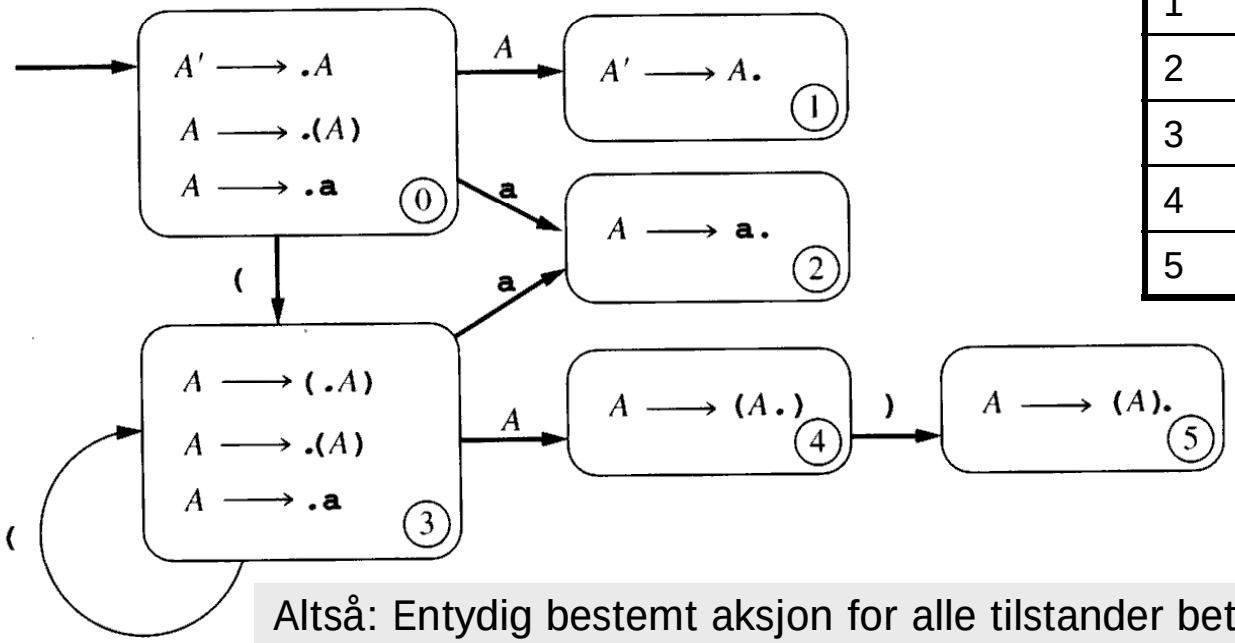
$$A \rightarrow (A) \mid \mathbf{a}$$

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

Har vi sett på:
Er ikke LR(0)

$$E' \rightarrow E$$
$$E \rightarrow E + \mathbf{n} \mid \mathbf{n}$$

A gir følgende LR(0) – DFA:



Tilst.	Mulig aksjoner:
0	Bare skift mulig, for ”(” og ”a”
1	Bare red. mulig, med A’ → A
2	Bare red. mulig, med A → a
3	Bare skift mulig, for ”(” og ”a”
4	Bare skift mulig, for ”)”
5	Bare red. mulig, med A → (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!

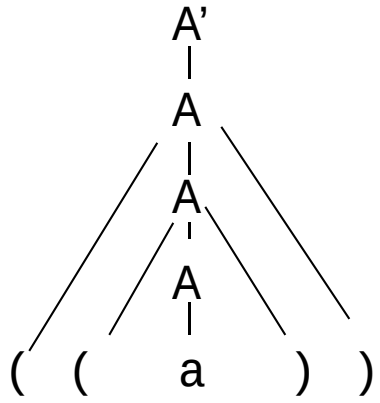
Tabell-oppsett for en LR(0) - grammatikk:

State	Action	Rule	Input			Goto
			(	a	)	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	$A \rightarrow (A)$				

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' → A, og input tom: Ferdig

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

State	Action	Rule	Input			Goto
			(	a	)	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 $A \to \alpha . X\, \beta$, 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 $A \to \alpha . X\, \beta$, then the new state to be pushed on the stack is ~~the state containing the item $A \to \alpha X . \beta$~~. If this token is not X for some item in state s of the form just described, an error is declared.

   t, where $s \xrightarrow{X} t$

2. If state s contains any complete item (an item of the form $A \to \gamma .$), then the action is to reduce by the rule $A \to \gamma$. A reduction by the rule $S' \to S$, 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 uncovered by the removal of γ). Again, by the construction of the DFA, this state must contain an item of the form $B \to \alpha . A\, \beta$. Push A onto the stack, and push (as the new state) the state ~~containing the item $B \to \alpha A . \beta$~~. (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.)

   t, where $u \xrightarrow{A} t$

Avslutning

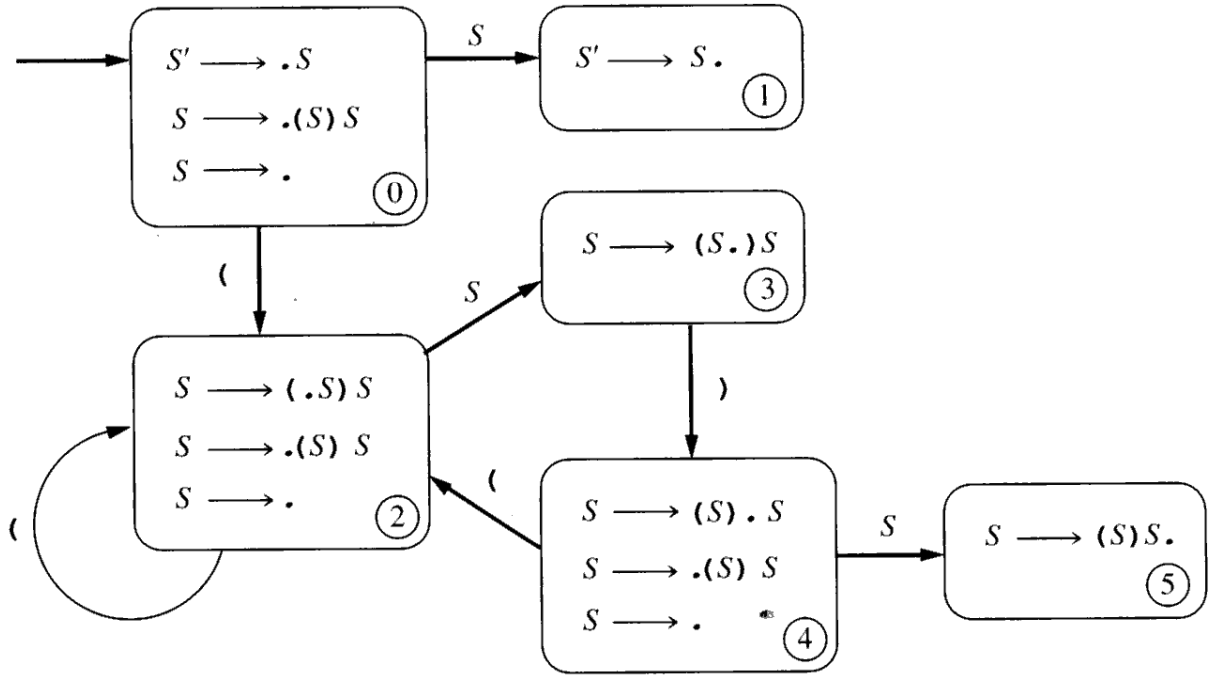
Kan forekomme i flere tilstander

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

$S' \rightarrow S$

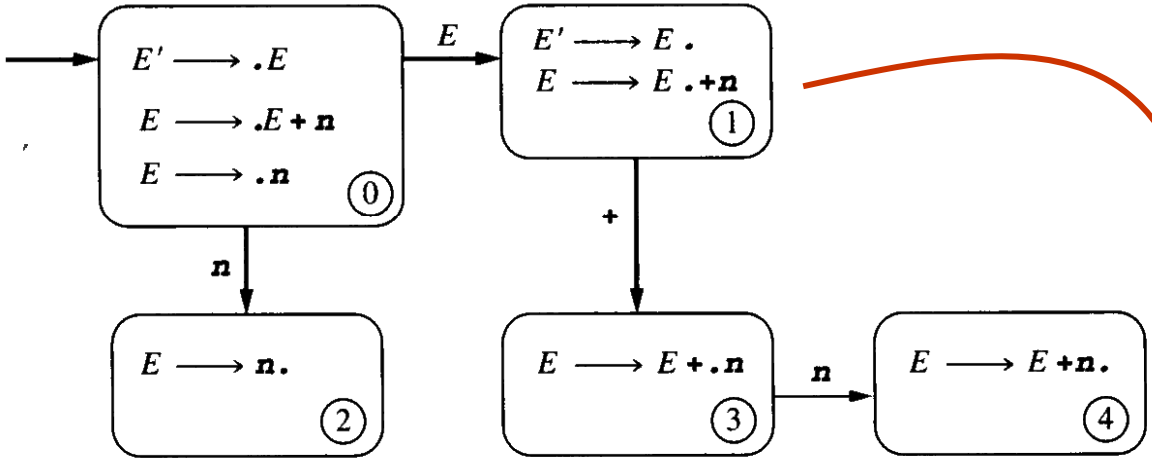
$S \rightarrow (\; S \;) \;\; S \mid \varepsilon$

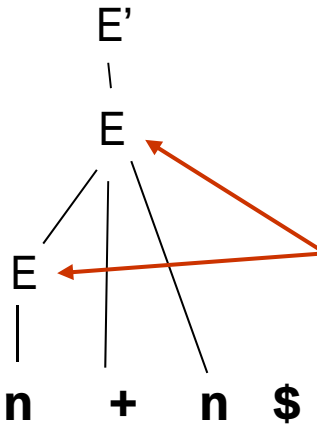
LR(0) – DFA:



Er den LR(0)? Nei, pga. tilstand 1:
(har vi sett på tidligere)

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





$	n + n $
$ n	+ n $
$ E	+ n $
Skal skifte	
$ E +	n $
$ E + n	$
$ E	$
Skal redusere med E'→E	
$ E'	$

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