Oppgaver til INF 5110, kapittel 5

- **Fra boka:** 5.3 Vi har sett litt på denne på en forelesning
  5.11 Vi har tidligere sett på: A -> (A) | a
  5.18 Forsøk også sette alternativet A -> A A til slutt

- **Utvid** grammatikken på den foilen (i Kap 5, del 2) som ser på den flertydige grammatikken:

  E' -> E E -> E + E | E * E | n

  med høyreassosiativ opphøying slik:

  E' -> E E -> E + E | E * E | E ** E | n

  og avgjør hvordan konfliktene da skal løses.

- **Oppgave 2** fra Eksamen 2006 (se neste side).

- **Kanskje også titte på eksamen 2007**, opg 1 a , b og c.
  Se på slutten av undervisningsplanen for 2008. Svarforslag ligger på:

www.uio.no/studier/emner/matnat/ifi/INF5110/v08/undervisningsmateriale/INF5110-eksamen2007+svar.pdf

Eksamen 2006, oppgave 2 (minus ett punkt)

Betrakt følgende grammatikk G, hvor S og T er ikketerminal-symboler, # og a er terminalsymboler, og S er startsymbolet.

$S \rightarrow T\ S$

$S \rightarrow T$

$T \rightarrow \#\ T$

$T \rightarrow a$

a) Finn First og Follow-mengdene til T og S (og la $ betegne 'end-of-file' som i boka).

b) Formulér med dine egne ord hvilke sekvenser av terminalsymboler du kan lage ut fra S'.

c) Avgjør om du kan lage et regulært uttrykk som uttrykker disse sekvensene av # og a som du

kan utlede fra S, og hvis svaret er 'ja', gi et slikt regulært uttrykk.

d) Innfør et nytt start-symbol S' → S og lag LR(0)-DFA-en for G rett fra denne grammatikken. Nummerér tilstandene.

f) Lag parseringstabellen for G ut fra den typen grammatikk den er.

g) Vis hvordan setningen: "a#a" vil bli parsert ved å skrive opp, som i boka, stakk-innholdet

og input for hver av skift- eller reduser-operasjon

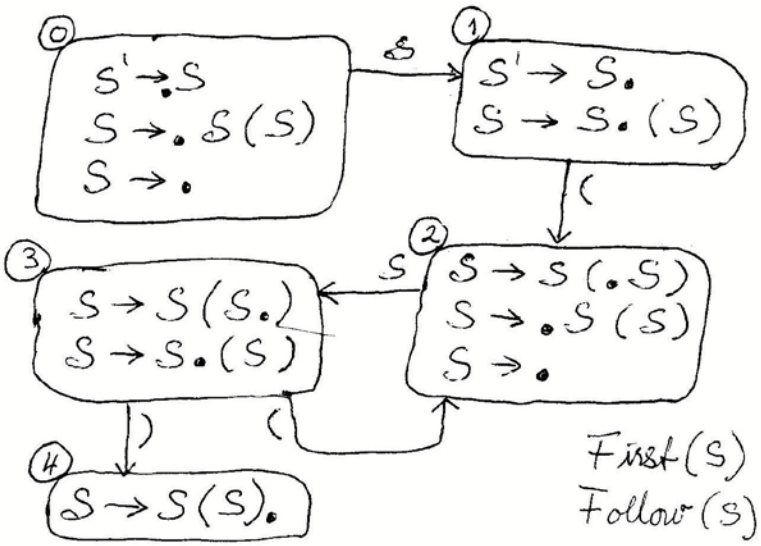
Fra boka: Oppgave 5.3

Kommentar: Denne er, i motsetning til den med S-> (S)S, ikke LL(1) (se kommentar i forbindelse med en av oppgavene til kap 4)

5.3 Consider the following grammar:

$$S \rightarrow S\ (\ S\ )\ |\ \varepsilon$$

a. Construct the DFA of LR(0) items for this grammar.
b. Construct the SLR(1) parsing table.
c. Show the parsing stack and the actions of an SLR(1) parser for the input string (()()).
d. Is this grammar an LR(0) grammar? If not, describe the LR(0) conflict. If so, construct the LR(0) parsing table, and describe how a parse might differ from an SLR(1) parse.



First(S) = {ε, (}
Follow(S) = {$, (,)}

	(	)	$	S	
0	r(S→ε)	r(S→ε)	r(S→ε)	1	accept!
1	s2		r(S'→S)		
2	r(S→ε)	r(S→ε)	r(S→ε)	3	
3	s2	s4			
4	r(S→S(S))	r(S→S(S))	r(S→S(S))		

SLR(1)-tabell

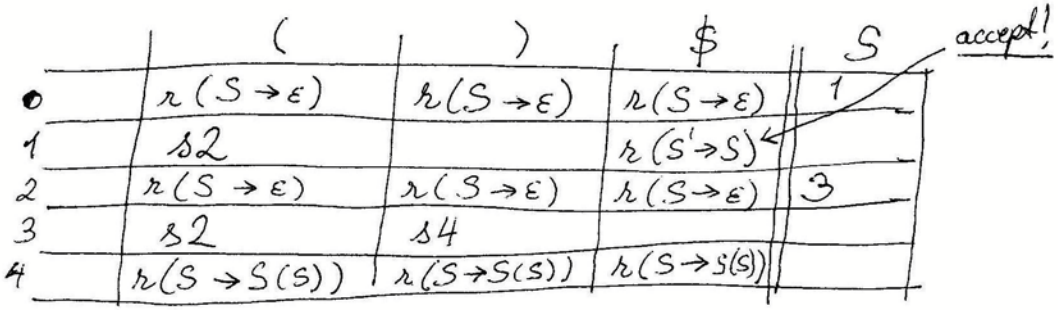
Den er ikke LR(0) på grunn av tilst. 1.

Merk at tilst. 0 og 2 ikke er problematiske siden det ikke er lovlig å skifte for noe terminalsymbol. Redusering er altså eneste mulighet

Den er SLR(1) fordi i tilstand 1 er red(S' -> S) bare aktuelt for "$", mens skift bare er aktuelt for "(".

Man kan også ekvivalent si at den er SLR(1) fordi tabellen ble entydig.

Oppgave 5.3, fortsatt

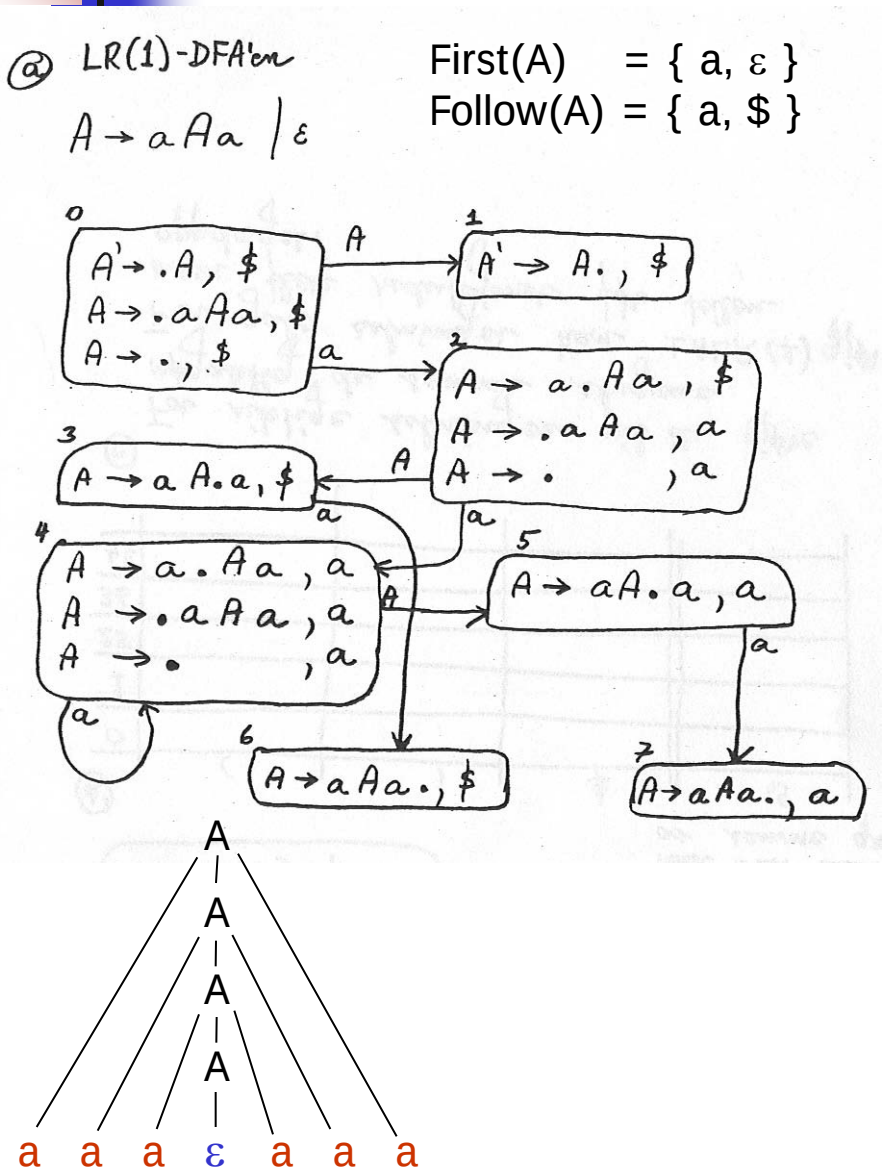


$0	(( ) ( )) $
$0 S 1	(( ) ( )) $
$0 S 1 (2	( ) ( )) $
$0 S 1 (2 S 3	( ) ( )) $
$0 S 1 (2 S 3 (2	) ( )) $
$0 S 1 (2 S 3 (2 S 3	) ( )) $
$0 S 1 (2 S 3 (2 S 3) 4	( )) $
$0 S 1 (2 S 3	( )) $
$0 S 1 (2 S 3 (2	)) $
$0 S 1 (2 S 3 (2 S 3	)) $
$0 S 1 (2 S 3 (2 S 3) 4	) $
$0 S 1 (2 S 3	) $
$0 S 1 (2 S 3) 4	$
$0 S 1	$

accept!

To røde streker under betyr at reduksjone med S -> ε *er utført*. En strek under betyr at det *skal* reduseres med det understrekede, mens piler betyr skift.

5.11 – a og b (se liknende eksempel på neste foiler)



Merk at det her er snakk om LR(1)-grammatikker, som ikke er pensum i 2013. Vi kan bare ta vekk "lookahead"-symbolene etter komma, og se på det som et SLR-problem.

For både tilstand 2 og 4 gjelder at på input 'a' så "foreslås" det både å skifte og å redusere med $A \rightarrow \varepsilon$. Altså er grammatikken ikke LR(1).

(b) Grammatikken genererer alle strenger med et partall antall 'a'-er, og den er entydig siden den bare kan gjøre dette på en måte (se figur).

Ekstra: Med denne grammatikken må vi imidlertid lese helt til slutten av setningen for å finne når vi skal gå over fra å skifte til å redusere. Det skal skje på midten.

Vi kan dermed se at grammatikken ikke er LR(k) for noen k.

Andre grammatikker som gir de samme setninger, og som helt kurant er SLR(1) er:

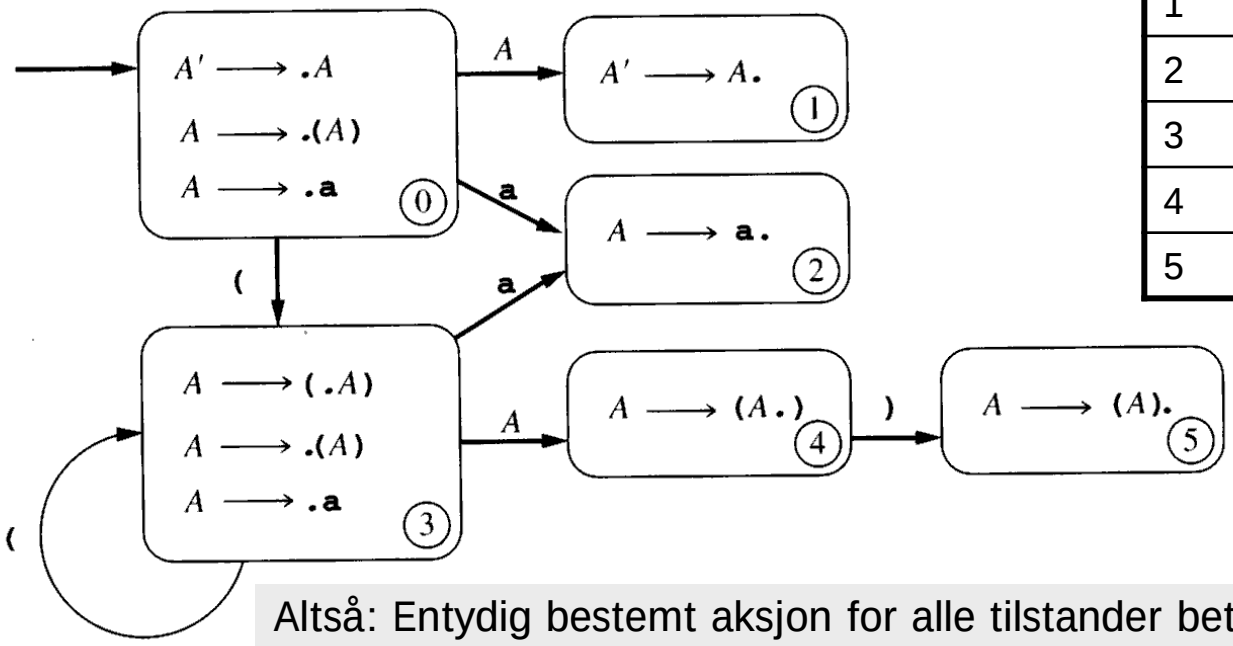
$A \rightarrow A\ a\ a \mid \varepsilon$ eller: $A \rightarrow a\ a\ A \mid \varepsilon$

Fra en tidligere foil med en liknende grammatikk
Er eksempel-grammatikkene i Kap 5 LR(0) ?

Ser på grammatikken: A -> (A) | a

Her midtpunktet markert ved en 'a', og da er den plutselig LR(0) !!

Den første 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 reduksjon!

Og vi fikk altså en entydig LR(0)-tabell:

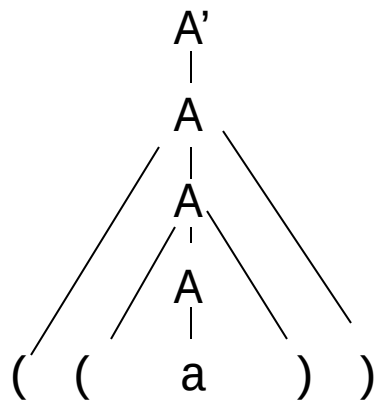
NB: Den er litt annerledes enn for SLR(1), LALR(1), LR(1), som alle er like

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



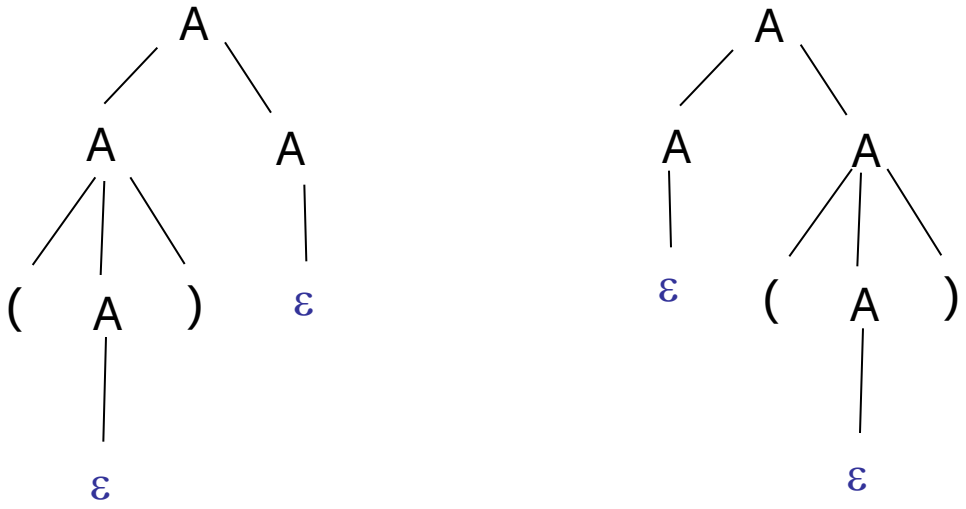
Skal her redusere med
A' → A, og input tom: Ferdig

Oppgave 5.18

Vi ser på grammatikken:
A -> A A | (A) | ε

Denne grammatikken genererer samme språk som de velkjente:
S -> (S) S | ε og S -> S (S) | ε
Nemlig: Alle korrekte parentesstrukturer.

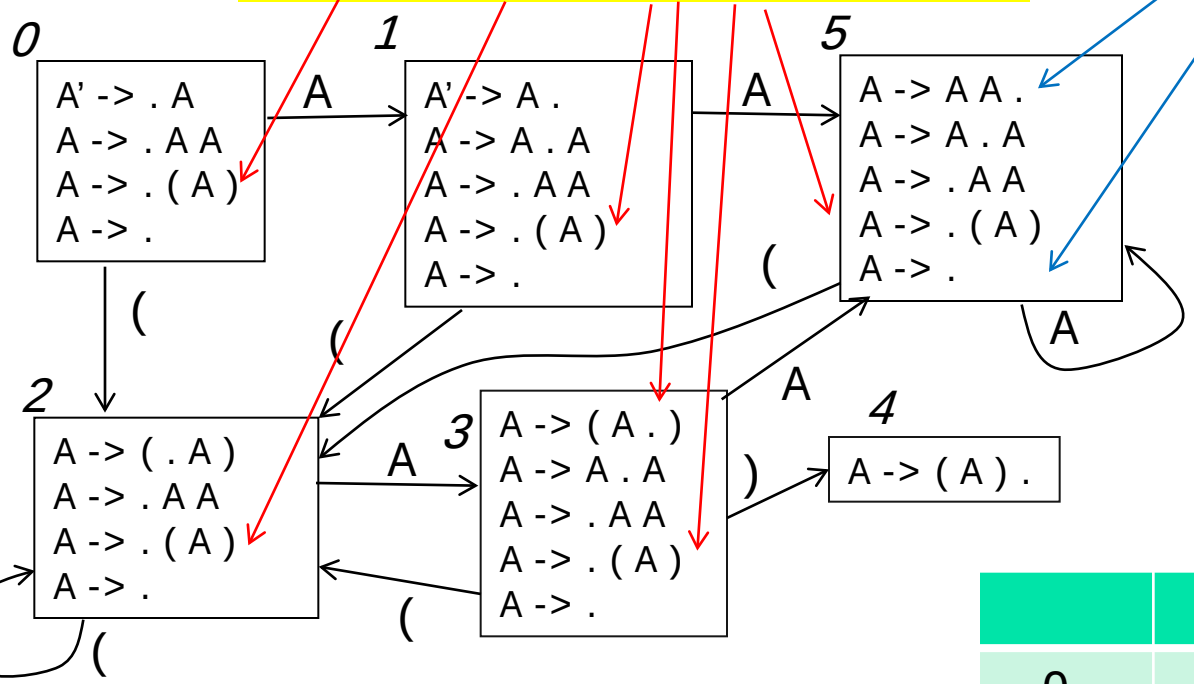
Men den nye er ”opplagt” flertydig! Vis det.
Svar: Her er to forskjellige trær for strengen: ” ( ) ”



Oppgave 5.18, side 2

Her kan man skifte for ”(” eller ”)”
Velger alltid det heller enn å redusere

Red./red.-konflikt.
YACC (og CUP?) velger den som står først, altså prod. (1)



Vi ser på grammatikken:

(1) A -> A A
(2) A -> (A)
(3) A -> ε

Follow(A) = { (,), $ }

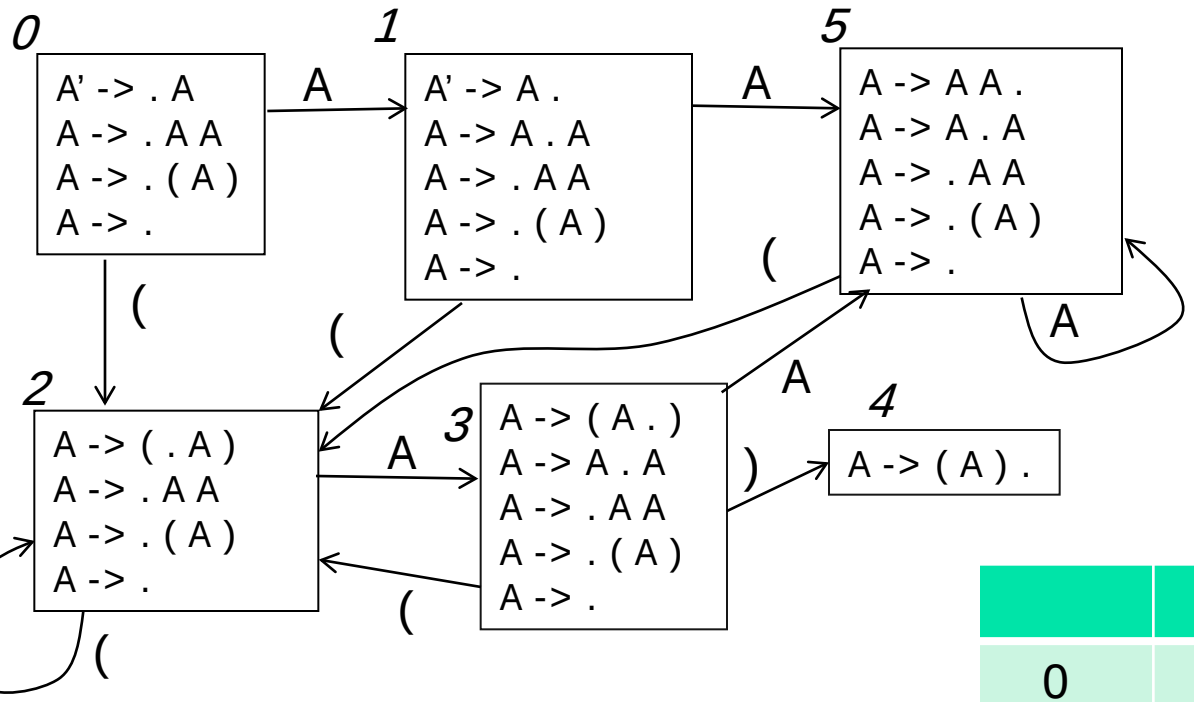
	(	)	$	A
0	s2	r(3)		1
1	s2		acc.	5
2	s2	r(3)	r(3)	3
3	s2	s4		5
4	r(2)	r(2)	r(2)	
5	s2	r(1)	r(1)	5

Cup/Yacc velger altså å redusere bare om lesing ikke er mulig, og om den *må* redusere velger den det alternativet som ligger først i grammatikken. Derfor: Om vi hadde byttet om prod. (1) og (3) ville Yacc/Cup altså valgt å redusere med A -> ε også for ”)” og ”$” i tilstand 5. Men, da ville automaten *hvertfall* ikke virke siden den aldri ville redusere med A -> A A (og det *må* den jo av og til gjøre).

Oppgave 5.18, side 3:

Vil denne LR-tabellen godkjenne alle setninger i språket??

Vi så jo på forrige foil (nederst til venstre) at uheldige valg ødela automaten!



Vi ser på grammatikken:

(1) A -> A A

(2) A -> (A)

(3) A -> ε

Follow(A) = { (,), $ }

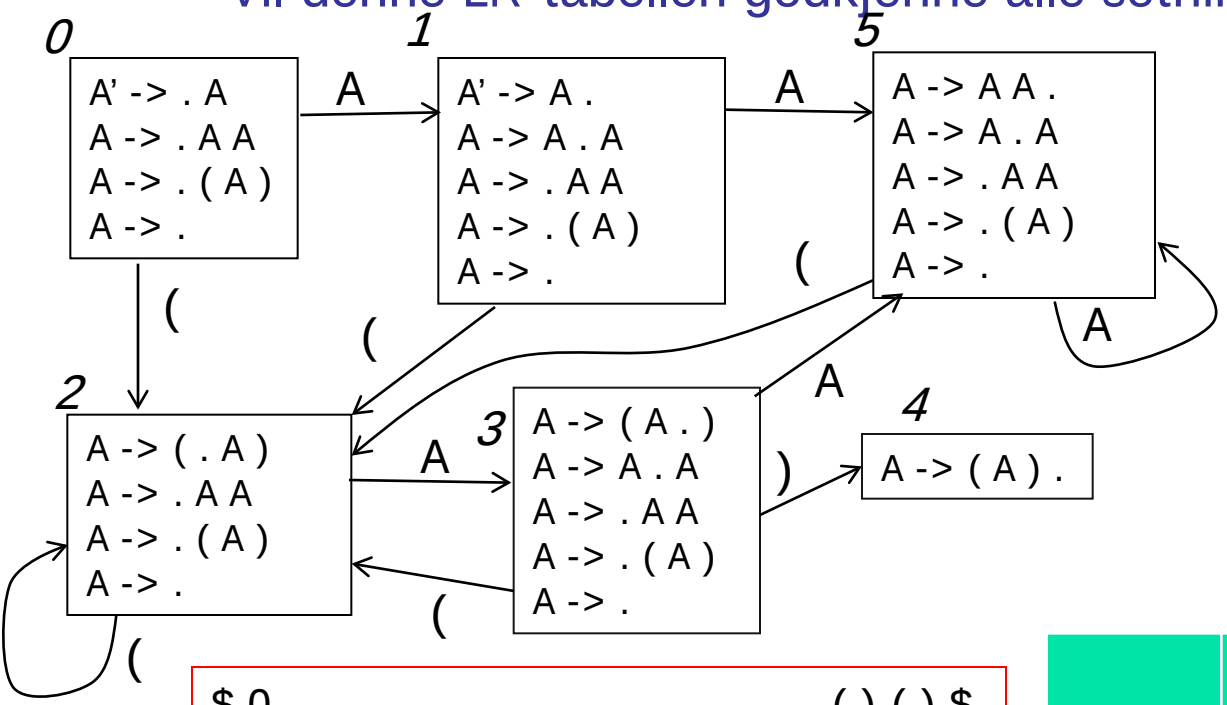
	(	)	$	A
0	s2	r(3)		1
1	s2		acc.	5
2	s2	r(3)	r(3)	3
3	s2	s4		5
4	r(2)	r(2)	r(2)	
5	s2	r(1)	r(1)	5

Men: Vil denne tabellen (med ”lure” valg) godkjenne akkurat de samme setningen som den gitte flertydige grammatikken?

Vi har ingen metode for å avgjøre dette, og må i stedet se på eksempler. Se neste side.

Oppgave 5.18, side 4:

Vil denne LR-tabellen godkjenne alle setninger i språket??



Vi ser på grammatikken:

(1) A -> A A
(2) A -> (A)
(3) A -> ε

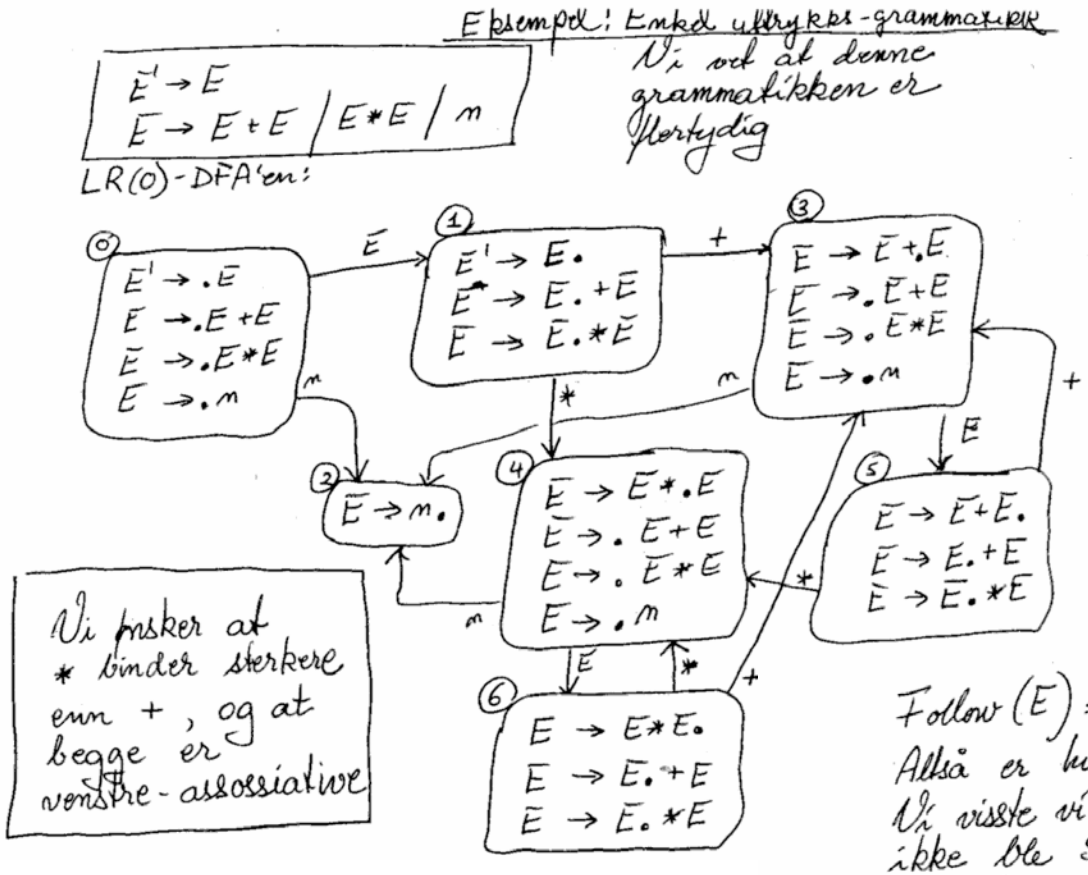
Follow(A) = { (,), $ }

	(	)	$	A
0	s2	r(3)		1
1	s2		acc.	5
2	s2	r(3)	r(3)	3
3	s2	s4		5
4	r(2)	r(2)	r(2)	
5	s2	r(1)	r(1)	5

$ 0	( ) ( ) $
$ 0 (2	) ( ) $
$ 0 (2 A 3	) ( ) $
$ 0 (2 A 3) 4	( ) $
$ 0 A 1	( ) $
$ 0 A 1 (2	) $
$ 0 A 1 (2 A 3	) $
$ 0 A 1 (2 A 3) 4	$
$ 0 A 1 A 5	$
$ 0 A 1	$

Det ser i hvert fall *lovende* ut mht. å klare alle setninger!

Gammel foil: Innfør her ** (høyeste presedens, og høyreassossitiv)



Fordel ved flertydige grammatikker:
De er som regel enklere å sette opp, se f.eks. til venstre her, og tidligere grammatikker for if-setningen

Konflikter må oppstå, men:
man kan løse mange konflikter med å angi presedens, assosiativitet, m.m. Dette kan angis f.eks. i CUP og Yacc

Tilstand 5: Stakk= .....E+E Input=
$: reduser, fordi skift ikke lovlig for $
+: reduser, fordi + er venstreassosiativ
*: skift, fordi * har presedens over +

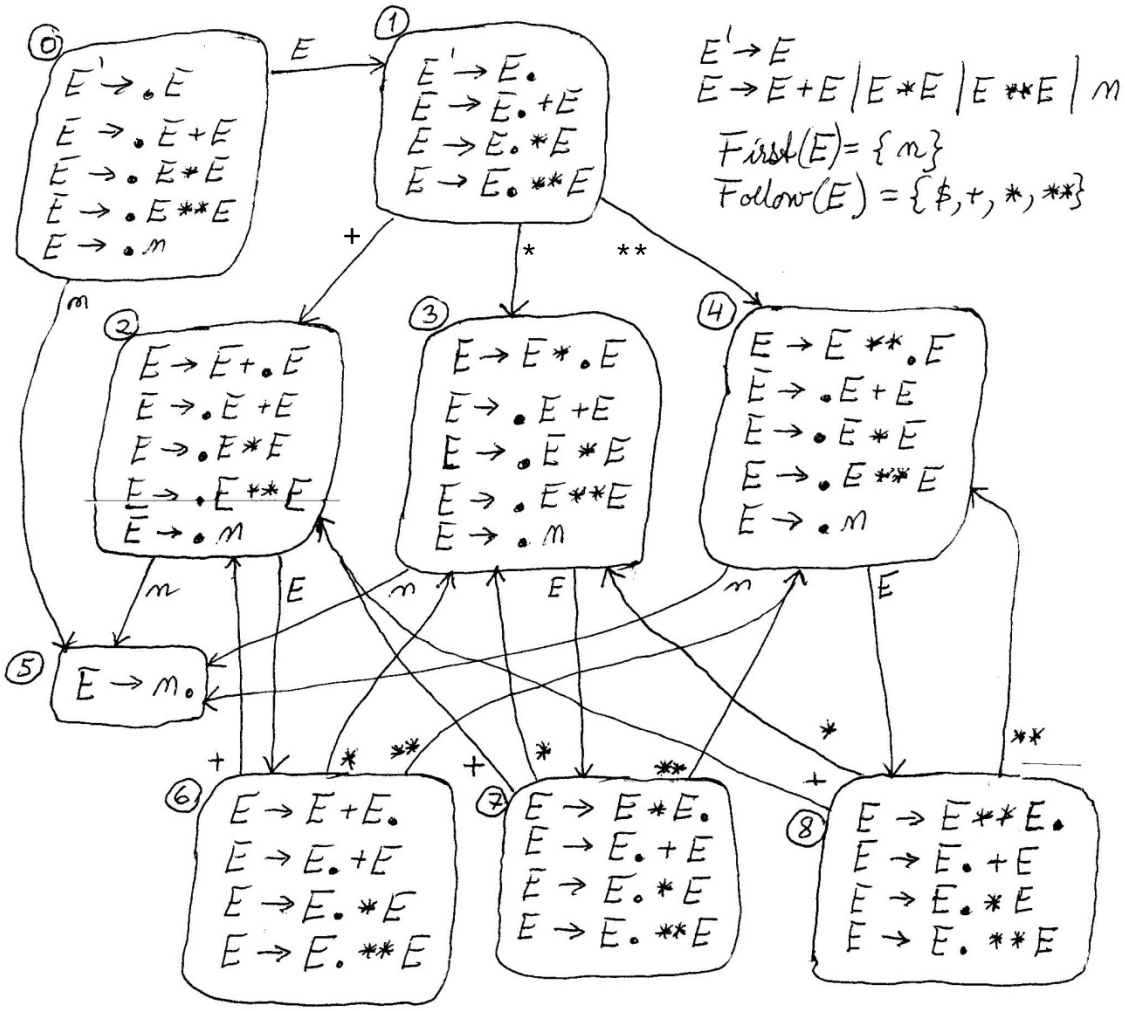
Tilstand 6: Stakk= .....E*E Input=
$: reduser, fordi skift ikke lovlig for $
+: reduser, fordi * har presedens over +
*: reduser, fordi * er venstreassossiativ)

Hva om også **? (høyreass.). Blir oppgave!

Follow(E) = {+, *, $}
Altså er hverken (5) eller (6) SLR-tilstander.
Vi visste vi måtte få konflikter slik at grammatikken ikke ble SLR-spiser ingen flertydige grammatikker er SLR.
Hva skal vi så gjøre i tilstand (5) og (6)?

	n	+	*	$	E
0	s2				1
1		s3	s4	accept	
2		r(E→n)	r(E→n)		5
3	s2				6
4	s2				
5		r(E→E+E)	s4	r(E→E+E)	
6		r(E→E*E)	r(E→E*E)	r(E→E*E)	

Innføring av høyre-assosiativ ** med høyeste presedens



Her måtte det bli konflikter, siden grammatikken er flertydig, og konfliktene opptrer i tilstandene 6, 7 og 8. Ut fra presedens og assosiativitet løser vi dette slik:

Tilstand 6 (E+E øverst på stakken)

\+ red(E-> E+E) (venstre-ass.)

* skift 3

** skift 4

Tilstand 7 (E*E øverst på stakken)

\+ red(E -> E*E)

* red(E -> E*E) (venstr-ass.)

** skift 4

Tilstand 8 (E**E øverst på stakken)

\+ red(E -> E**E)

* red(E -> E**E)

** skift 4 (høyre-ass.)

NB: Dette skjer altså automatisk i CUP når man har oppgitt presedens og assosiativitet riktig.

Eksamen 2006, oppgave 2
(se undervisningsplanen 2008).

Betrakt følgende grammatikk G, hvor S og T er ikke-terminaler, # og a er terminalsymboler, og S er startsymbolet.

$S \rightarrow T\ S$

$S \rightarrow T$

$T \rightarrow \#\ T$

$T \rightarrow a$

a) Finn First og Follow-mengdene til T og S (og la $ betegne 'end-of-file' som i boka).

b) Formulér med dine egne ord hvilke sekvenser av terminalsymboler du kan lage ut fra S'.

c) Avgjør om du kan lage et regulært uttrykk som uttrykker disse sekvensene av # og a som du kan utlede fra S, og hvis svaret er 'ja', gi et slikt regulært uttrykk.

d) Innfør et nytt start-symbol $S' \rightarrow S$ og lag LR(0)-DFA-en for G rett fra denne grammatikken. Nummerér tilstandene.

e) Lag parseringstabellen for G ut fra den type grammatikk den er.

f) Vis hvordan setningen: "a#a" vil bli parsert ved å skrive opp, som i boka, stakk-innholdet og input for hver av skift- eller reduser-operasjon

Eksamen 2006, del 2

2a

S → T S	Gjør at Fi(S) skal ha alle fra Fi(T), og at Fo(T) skal ha alle fra Fi(S)
S → T	Gjør, som over, at Fi(S) skal ha alle fra Fi(T), og at Fo(T) skal ha alle fra Fo(S)
T → # T	Gjør at Fi(T) skal inneholde #
T → a	Gjør at Fi(T) skal inneholde a

	First	Follow
S	**a #**	**$**
T	**a #**	**a # $**

2b

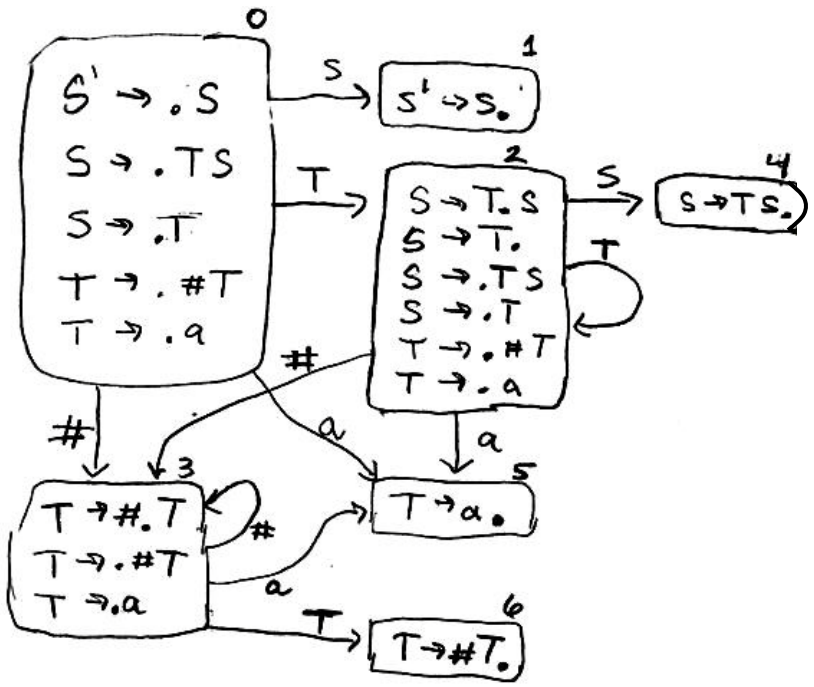
Vi kan fra S generere en-eller-flere 'a'-er hvor hver a har null eller flere # foran seg.

2c

Vi har flg. regulære uttrykk { {#}* a}+

Eksamen 2006 del 2, oppgave d

2d LR(0)-DFA'en blir som følger:



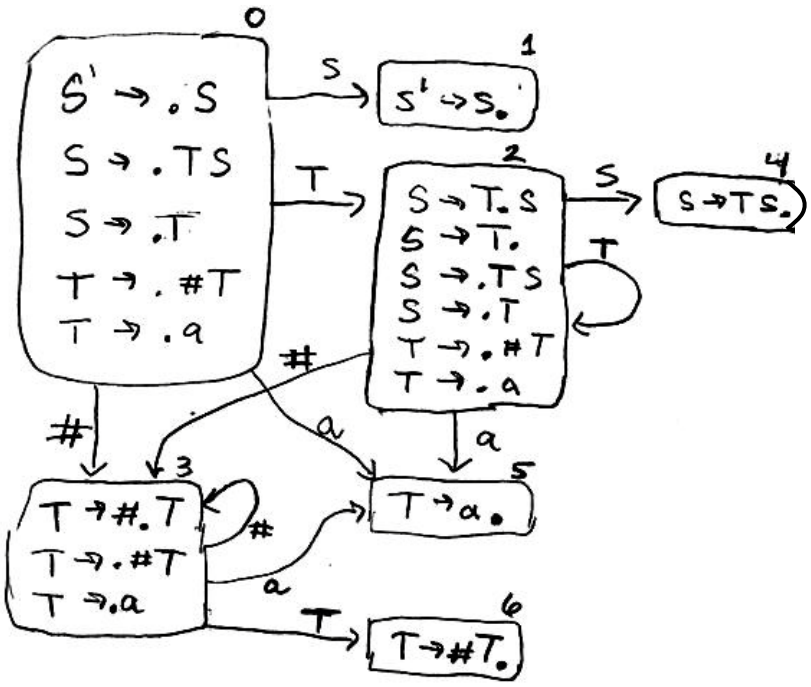
S → T S
S → T
T → # T
T → a

Eksamen 2006 del 2, oppgave 2 e

2e

	a	#	$	S	T
0	s5	S3		1	2
1			accept		
2	s5	s3	r(S->T)	4	2
3	s5	s3			6
4			r(T->TS)		
5	r(T->a)	r(T->a)	r(T->a)		
6	r(T->#T)	r(T->#T)	r(T->#T)		

(Merk at i tilstand 2 har vi både skift og reduksjon, valgt ut fra look-ahead-symbolet)



	First	Follow
S	**a #**	**$**
T	**a #**	**a # $**

Eksamen 2006 del 2, oppgave 2f

2f

	a	**#**	**$**	**S**	**T**
0	s5	S3		1	2
1			accept		
2	s5	s3	r(S->T)	4	2
3	s5	s3			6
4			r(T->TS)		
5	r(T->a)	r(T->a)	r(T->a)		
6	r(T->#T)	r(T->#T)	r(T->#T)		

Analyse av a # a :

$ 0	a # a $
$ 0 a 5	# a $
$ 0 T 2	# a $
$ 0 T 2 # 3	a $
$ 0 T 2 # 3 a 5	$
$0 T 2 # 3 T 6	$
$0 T 2 T 2	$
$ 0 T 2 S 4	$
$ 0 S 1	$
accept	