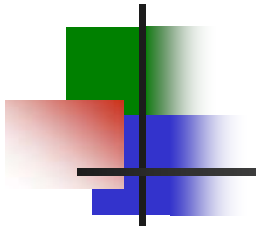


Oppgaver til INF 5110, kapittel 5, med svarforslag

Gjennomgått torsdag 26. febr. 2008. Dette er versjon fra 28/7

OPPGAVER:

- **Fra boka:** 5.3, 5.4, 5.11, 5.12, 5.13.
- **Oppgave 2** fra Eksamen 2006.
- **Utvid grammatikken på foil 4 (nr. 6 i siste utgave)** for kap 5, del2, med høyreassosiativ opphøying (dobbel stjerne), og se på hvordan konflikter da skal løses.

- Torsdag 28/2: Egentlig ingen undervisning.
- Men det ble nok litt mange oppgaver, så vi kommer neppe gjennom alt
- Derfor håper jeg folk går hjem og studerer svarforslagene. Om det er uklarheter kommer jeg kl. 10.15 på torsdag 28/2, og svarer på spørsmål som noen måtte ha.
- NB: Er i Lille aud. minste et kvarter (til 10.30), men går da om det ikke er flere spørsmål !

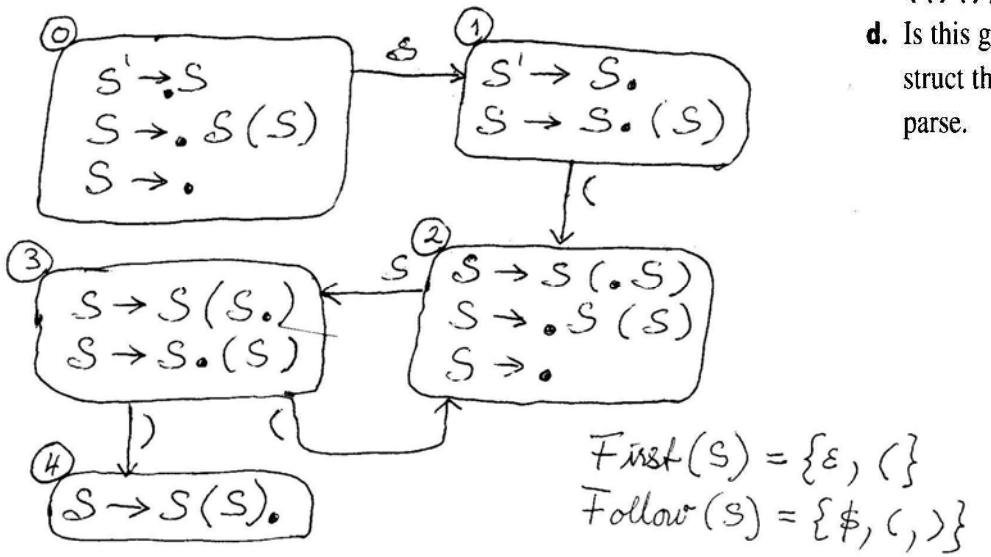
Fra boka: Oppgave 5.3

Kommentar: Denne er, i motsetning til den med S-> (S)S, ikke LL(1) (se kommentar i forbindelse med én 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
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))	

accept!

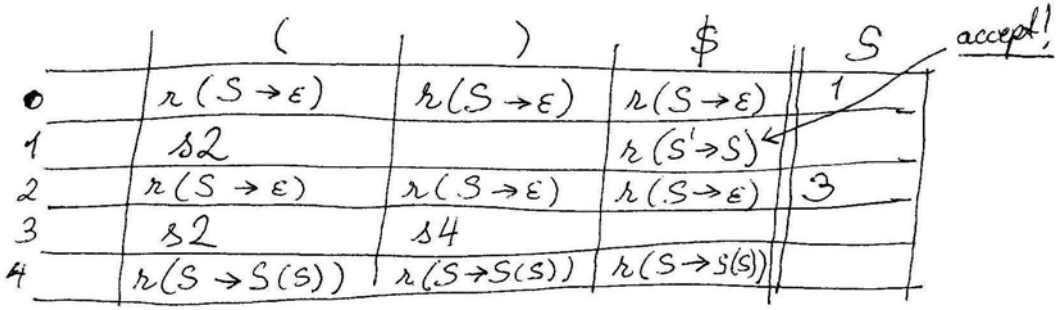
← SLR(1)-tabell

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

Merk at tilst. 0 og 2 ikke er problematiske siden det ikke er aktuelt å skifte for noe terminalsymbol.

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.



$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.

Oppgave 5.4

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

5.4 Consider the grammar of

a. Construct the DFA of LR(1) items for this grammar.

b. Construct the general LR(1) parsing table.

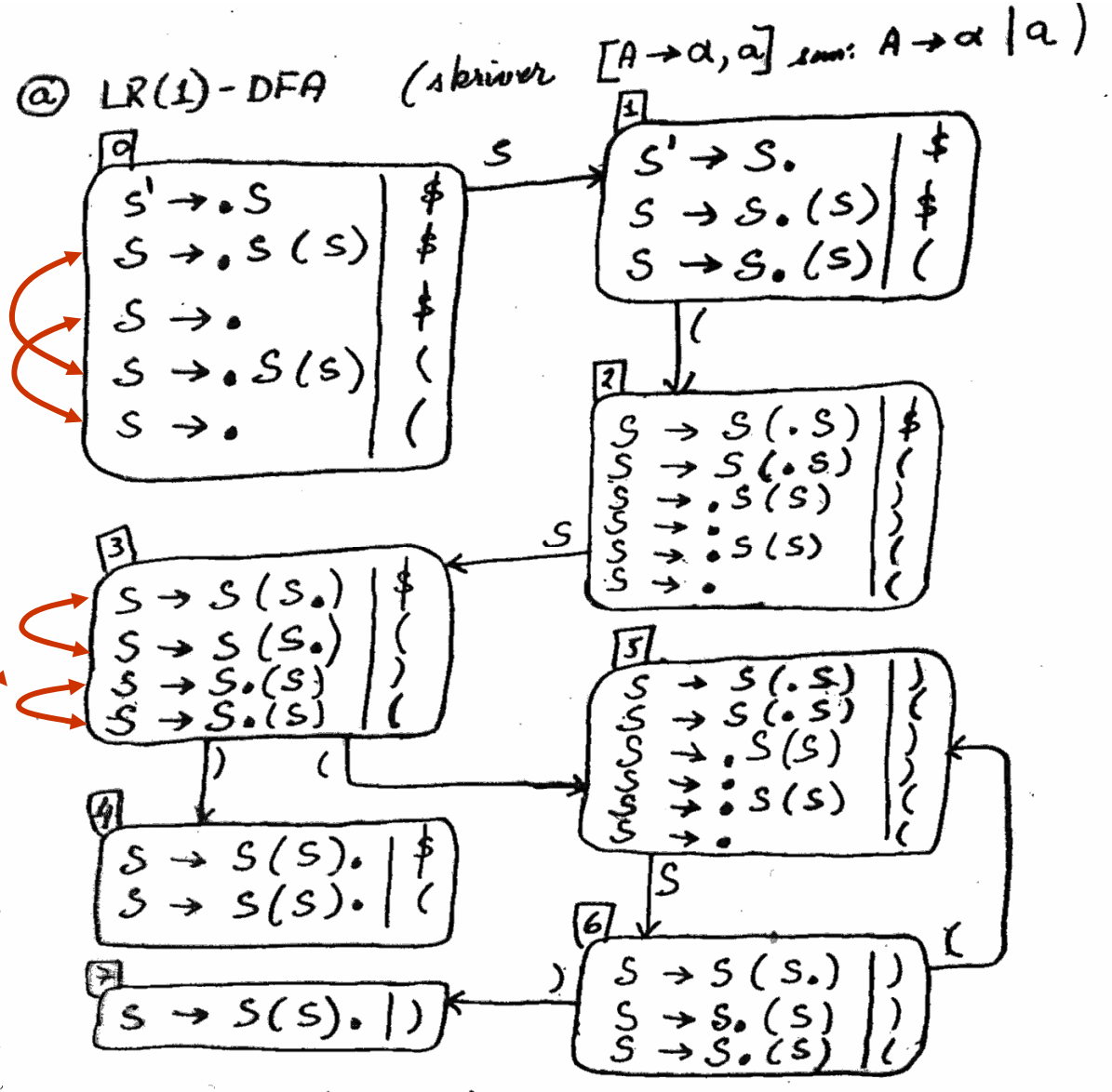
c. Construct the DFA of LALR(1) items for this grammar.

d. Construct the LALR(1) parsing table.

e. Describe any differences that might occur between the actions of a general LR(1) parser and an LALR(1) parser.

Oppgave 5.4 (a)

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



Samme "kjerne" (core) men annen lookahead. Går igjen hele veien. I den basale LR(1)-måten skriver man disse itemene hver for seg.

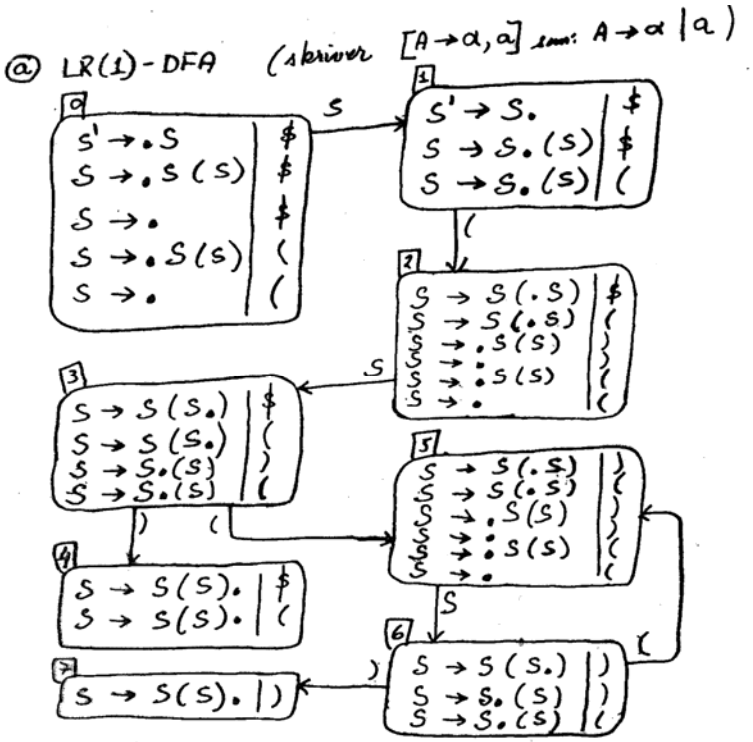
I LALR(1)-DFA'er vil man derimot skrive disse to som én linje, slik:

… S → S • (S) …	) (

Oppgave 5.4 (a og b)

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

Merk at lookahead-symbolet ikke betyr noe for tabell-oppsettet når punktumet står inni, bare når det står til slutt!

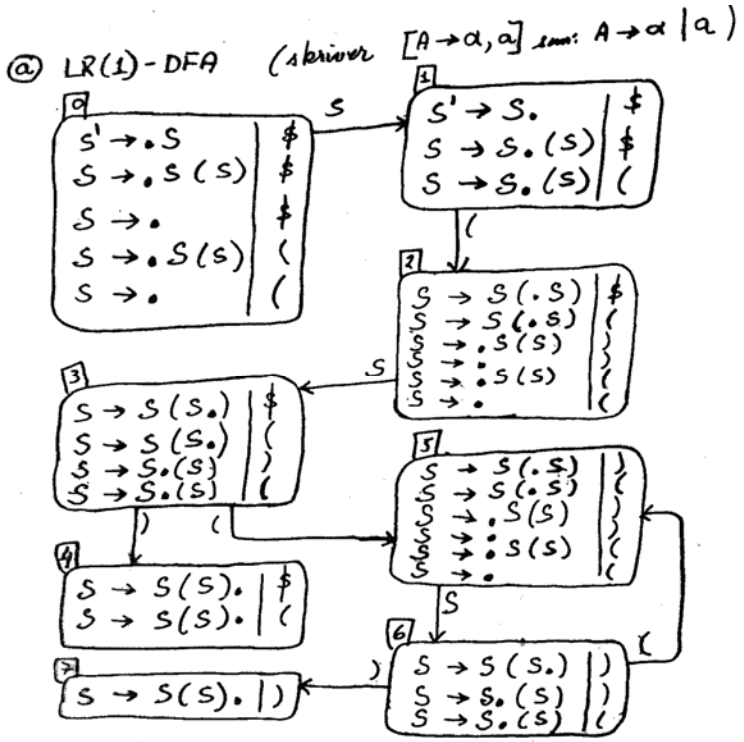


b)

	(	)	$	S
0				
1				
2				
3				
4				
5				
6				
7				

Oppgave 5.4 (a og b)

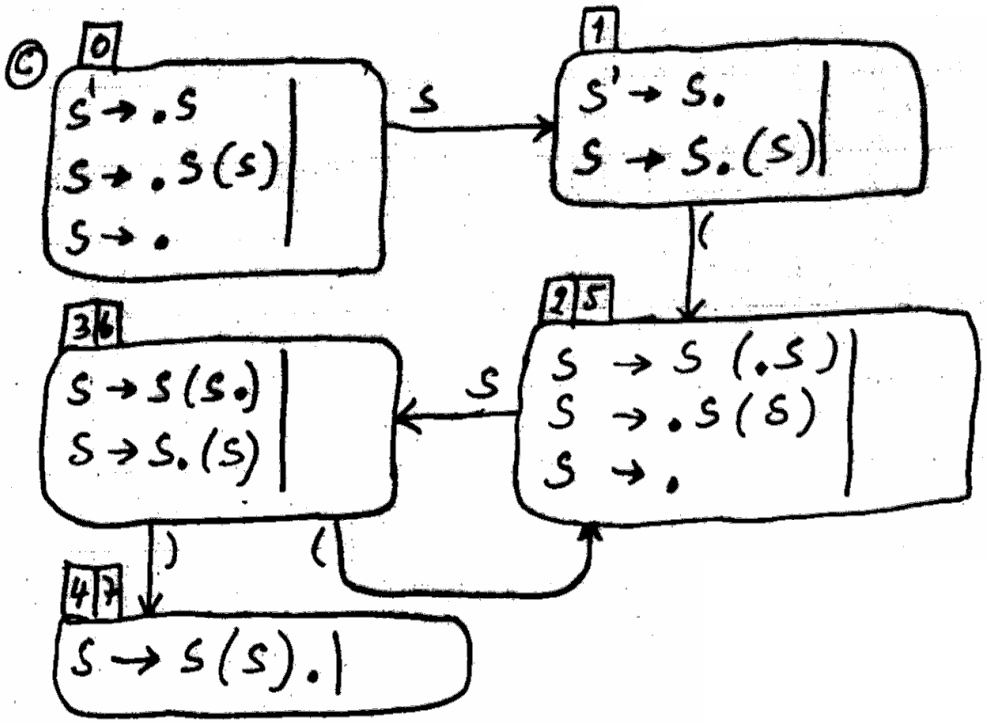
$$S \rightarrow S\ (\ S\ )\ |\ \varepsilon$$



b)

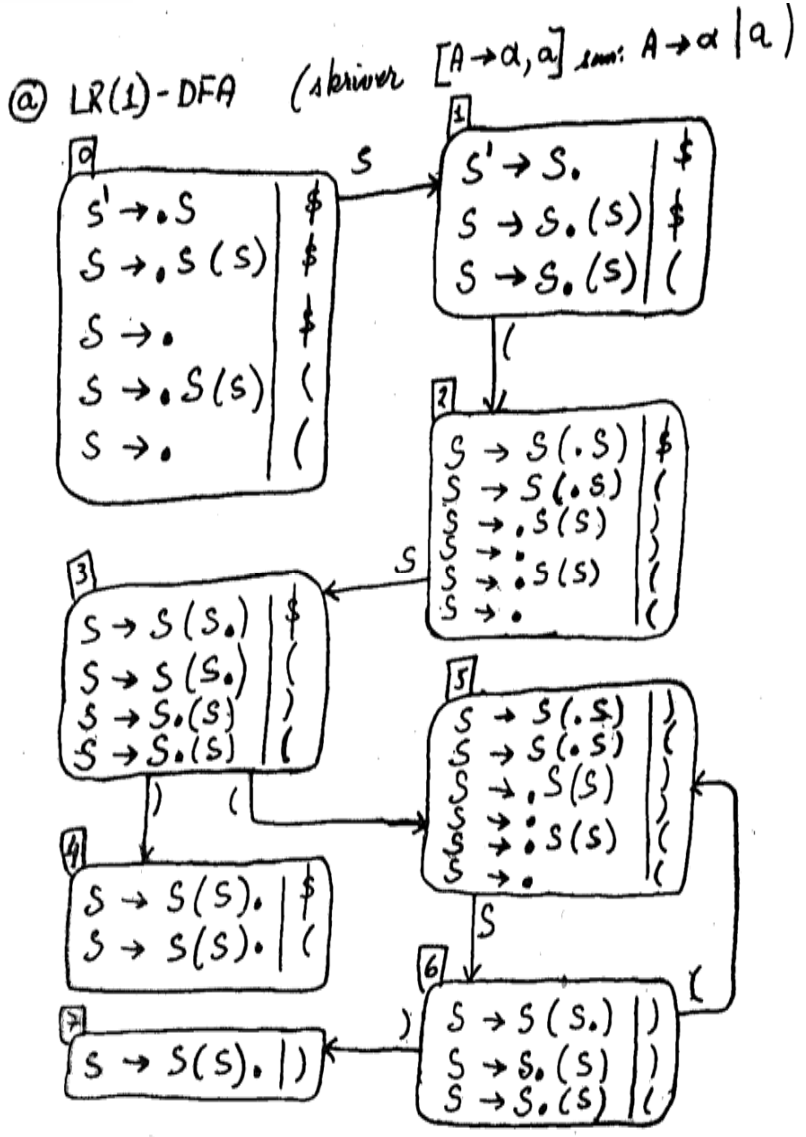
	(	)	$	S
0	r(S → ε)		r(S → ε)	1
1	s 2		accept	
2	r(S → ε)	r(S → ε)		3
3	s 5	s 4		
4	r(S → S (S))		r(S→ S (S))	
5	r(S → ε)	r(S → ε)		6
6	s 5	s 7		
7		r(S → S (S))		

Oppgave 5.4 -c

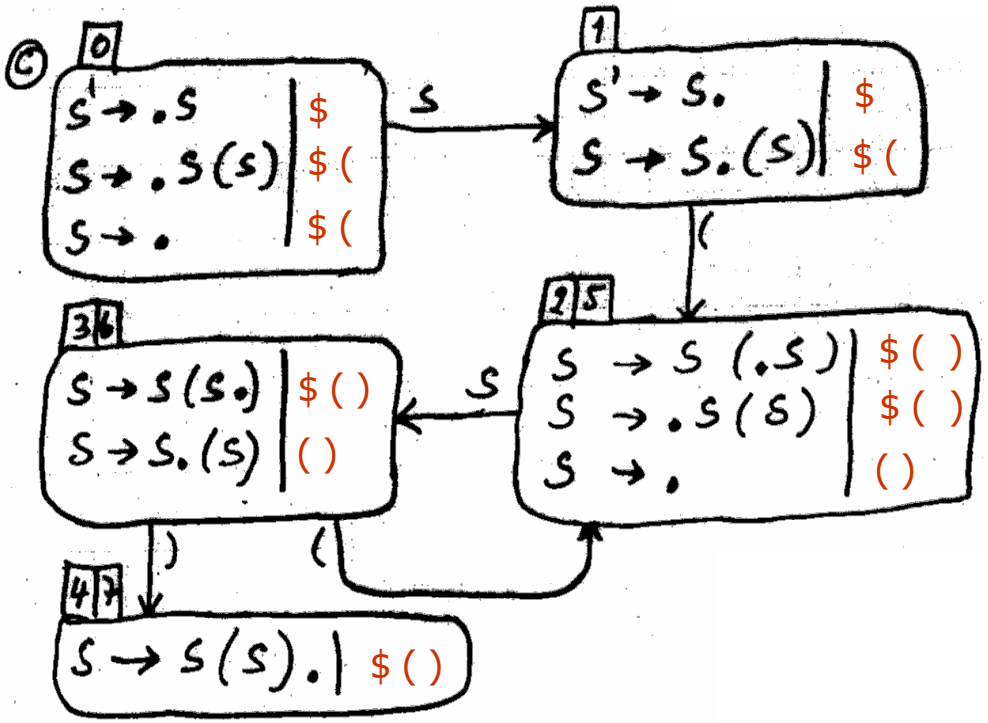


Over er angitt forberedelsen til en LALR(1)-DFA. De som har lik "core" er slått sammen i forhold til LR(1)-DFA'en til høyre (og vi har derfor fått LR(0)-DFA'en), men vi har ennå ikke fylt inn lookahead-mengdene som altså skal være unionen av look-aheadene fra alle tilsvarende LR(1)-itemer. Se neste foil.

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



Oppgave 5.4 -c

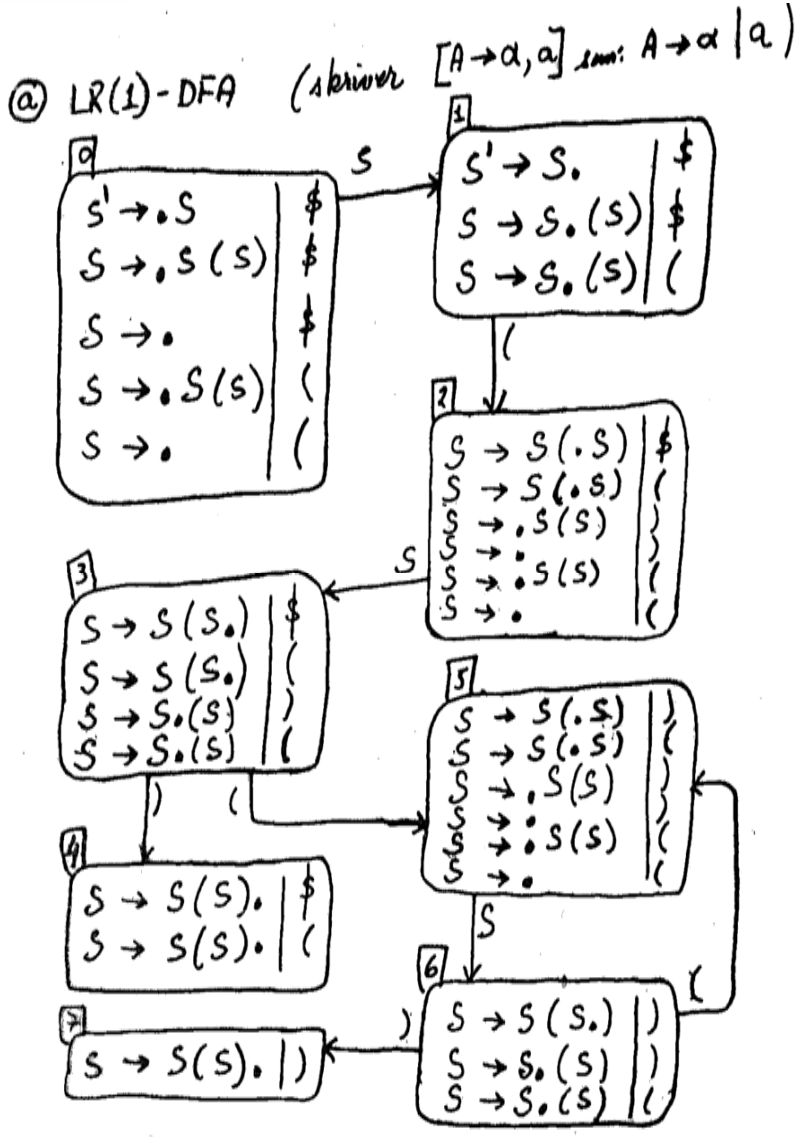


Follow(S) = { (,), $ }

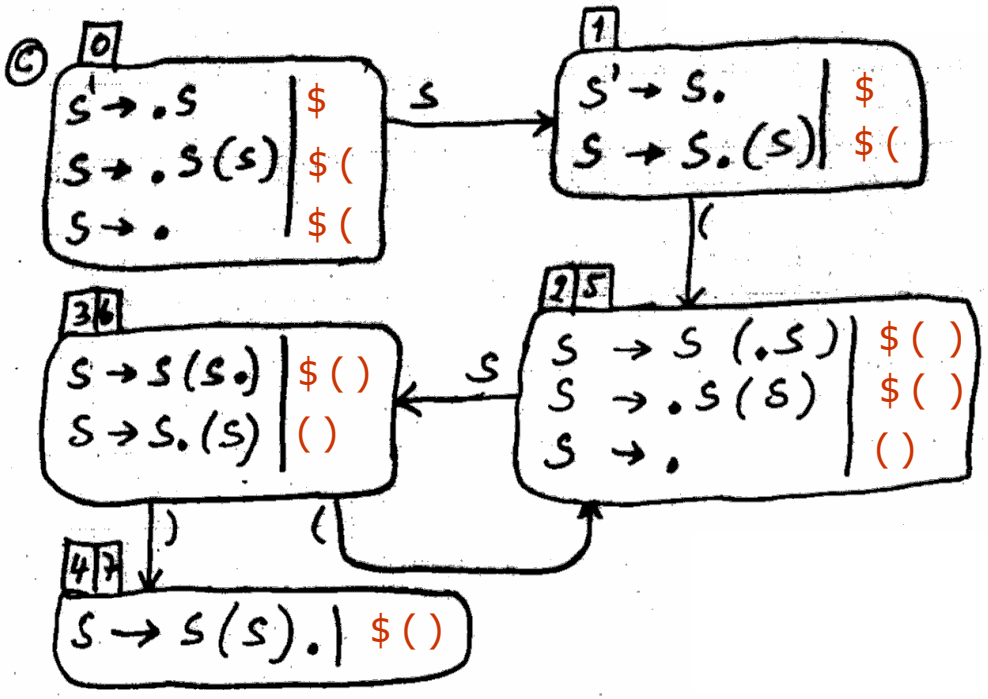
For en del slutt-itemer får vi altså færre lookahead-symboler enn de i Follow(S).

(Dette kunne gitt kraftigere analyse enn for SLR(1), men det får ikke betydning her fordi grammatikken allerede er SLR(1)).

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



Oppgave 5.4 - d og e



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

Deloppgave (e):

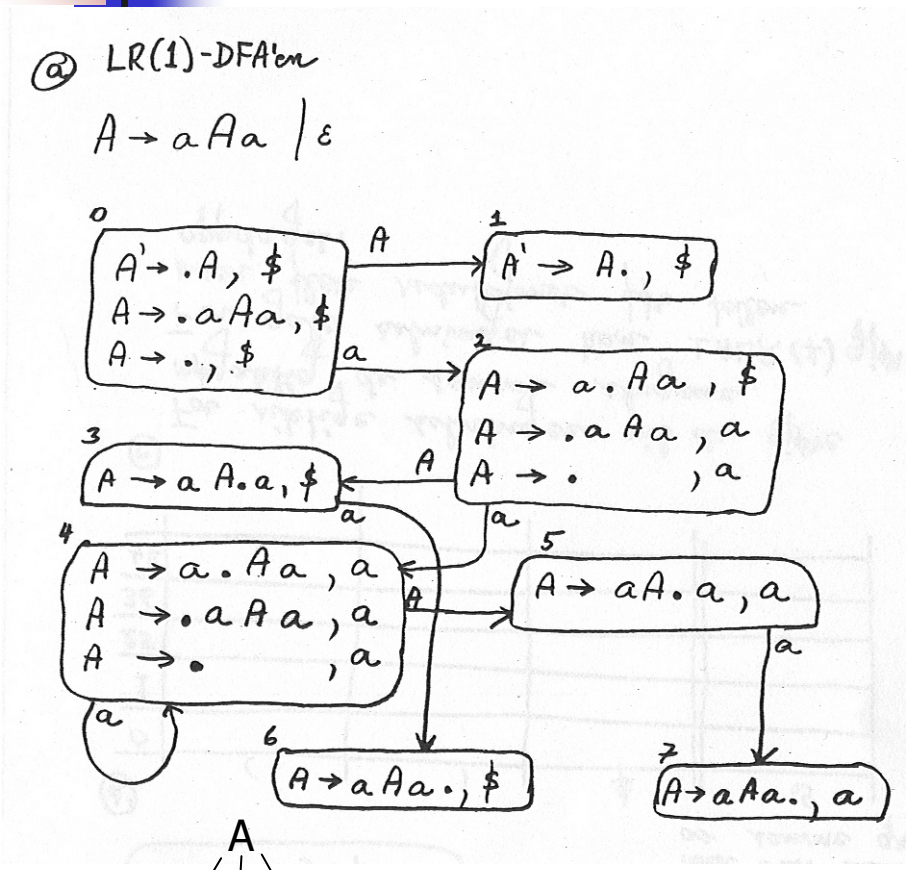
For riktige setninger vil en med LALR(1)-tabellen og LR(1)-tabellen gjøre nøyaktig de samme stegene.

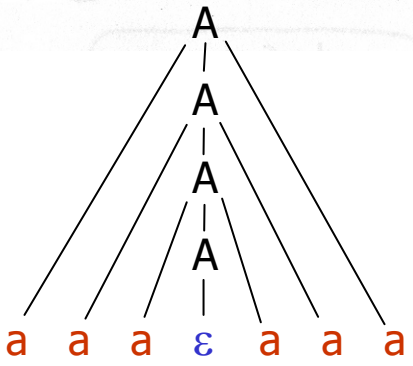
For gale setninger kan man med LALR(1)-tabellen gjøre noen flere reduksjoner enn med LR(1)-tabellen før feilen oppdages

(d)

	(	)	$	S
0	r(S → ε)		r(S → ε)	1
1	s25		accept	
25	r(S → ε)	r(S → ε)		36
36	s25	s47		
47	r(S → S (S))	r(S → S (S))	r(S → S (S))	

5.11 – a og b





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.

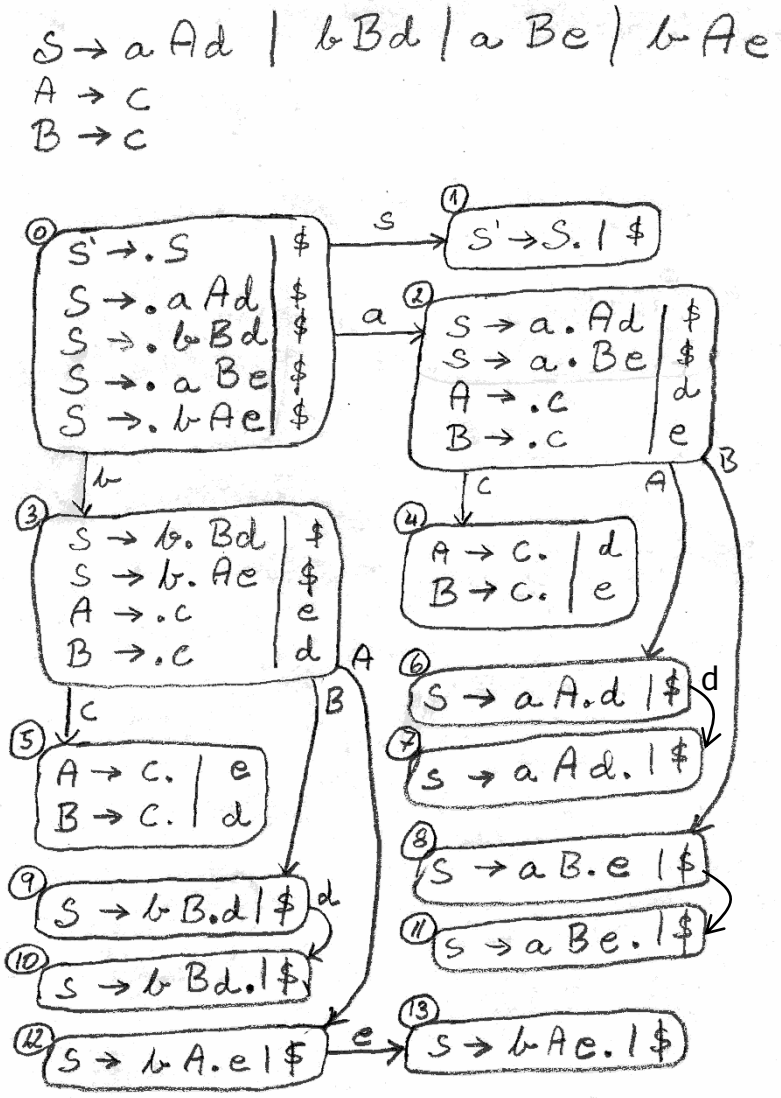
Andre grammatikker som gir de samme setninger, og som helt kurant er SLR(1)

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

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

Oppgave 5.12

En rimelig idiotisk grammatikk som er LR(1), men ikke LALR(1)



LR(1)-DFA'en blir som angitt til venstre. De eneste stedene det kunne bli LR-konflikt er tilstandene 4 og 5 (reduse/reduser-konflikter)

Men vi ser de lar seg løse i LR(1), siden det i hver tilstand er slik at lookahead-symbolet er forskjellige for de to reduksjonene.

Men, når vi skal lage LALR(1)-DFA'en ser vi at tilstandene 4 og 5 blir slått sammen, og da får begge reduksjonene lookahead-mengden {d, e}. Dermed lar ikke konflikten seg løse i LALR(1)-DFA'en.

Kunne vi også laget et eksempel på en LR(1)-grammatikk der LALR(1)-DFA'en fikk en skift/red.-konflikt?

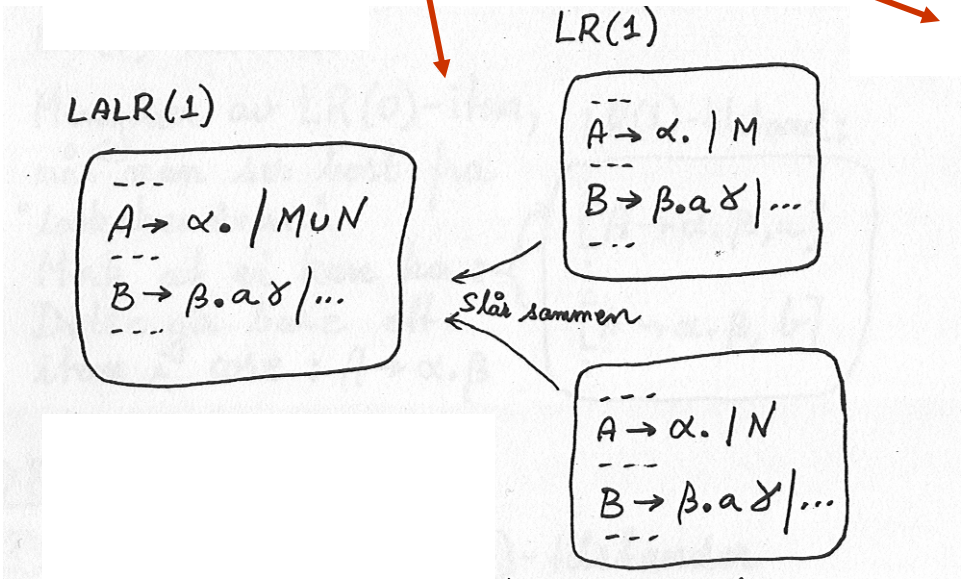
Svaret er *NEI*, se oppg. 5.13 neste lysark

5.13:

Anta at en grammatikk G er LR(1) men ikke LALR(1). Vis at LALR(1)-DFA'en til G da bare kan ha red./red.-konflikter (altså kan ikke ha skift/red.-konflikter)

Vi viser: Dersom LALR(1)-DFA'en til en grammatikk har en skift/red.-konflikt, så vil også LR(1)-DFA'en ha en skift/red.-konflikt.

Kobling: Anta vi viser det til venstre, og at G er LR(1), men at den ikke er LALR(1) bl.a. på grunn av en skift/red.-konflikt. Dette blir en selvmotsigelse, for vi viser til venstre at en av tilstandene i LR(1)-DFA'en da *også må* ha en skift/red.-konflikt, og at grammatikken dermed ikke kan være LR(1).



M og N er mengder av lookahead-symboler

Vi antar at skift/red.-konflikten i LALR(1)-DFA'en består i at $a \in M \cup N$.

Da må enten $a \in M$ eller $a \in N$, og det betyr at det måtte være en skift/red.-konflikt i en av LR(1)-tistandene.

Eksamen 2006, oppgave 2

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 $S' \rightarrow S$ og lag LR(0)-DFA-en for G rett fra denne grammatikken. Nummerér tilstandene.

e) Gi et kort argument som bestemmer hvike(n) av følgende fem grupper G hører med i:

   a. LR(1)
   b. LALR(1)
   c. SLR(1)
   d. LR(0)
   e. Ingen av de overstående.

Hint: Finn ut hvilke mulige konflikter du har i DFA-en og/eller om grammatikken er entydig..

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

Eksamen 2006, oppgave 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

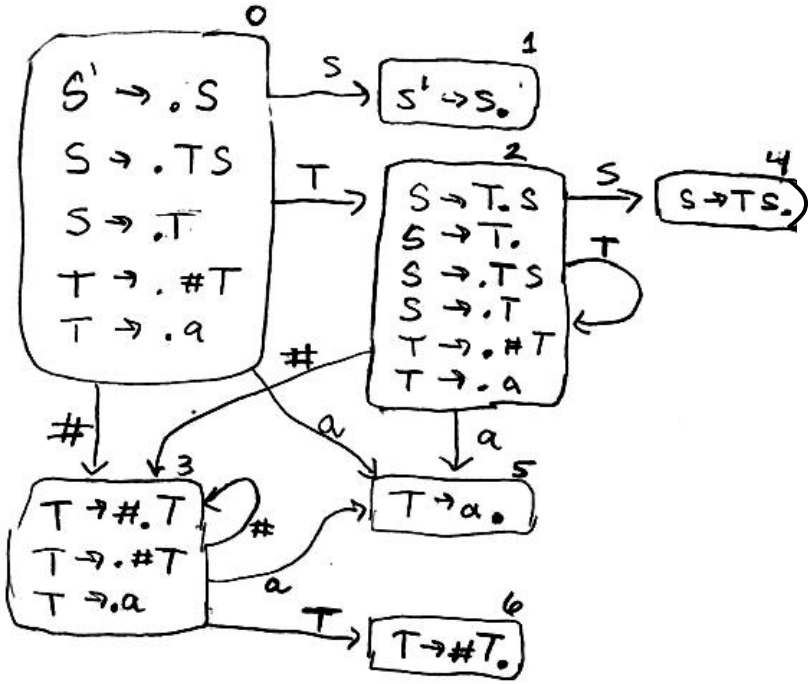
Vi kan fra S generere en-eller-flere 'a'-er hvor hver a har null eller flere # foran seg.

2c

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

Eksamen 2006, oppgave 2

2d LR(0)-DFA'en blir som følger:



S → T S
S → T
T → # T
T → a

Eksamen 2006, oppgave 2

2e

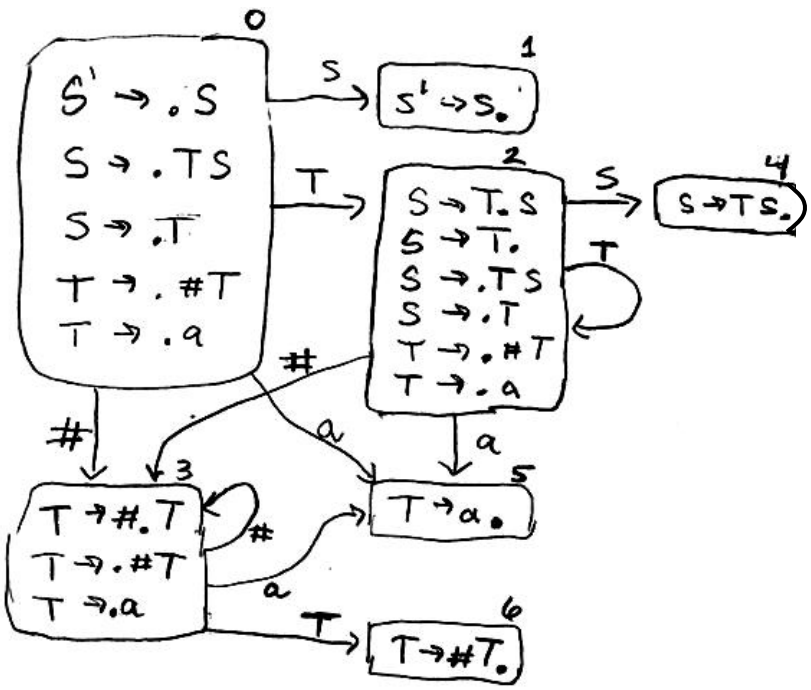
Den er ikke LR(0) fordi det ikke er entydig hvilken regel vi skal bruke etter f.eks å ha lest '#' (vi må se på nest symbol for å se hvilken produksjon vi skal nytte – kommer det en ny # eller 'a'?) Det er altså shift/red konflikt i tilstand 2.
Den er SLR(1) fordi den skift-reduser konflikten vi har i tilstand 2 løses ved at Followmengden til S ikke inneholder # eller a (dvs. har vi # eller a som neste symbol, benyttes T-> .#T med overgang til 3 eller 5, mens er det $ gjør vi reduksjon med S -> T og går tilbake til 0 og så 1 og accept).
Siden den er SLR(1), er den også LALR(1) og LR(1). Den er følgelig også entydig.

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)		

(Merk at i tilstand 2 har vi både skift og reduksjon – basert på look-ahead-symbolet)

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



Eksamen 2006, oppgave 2

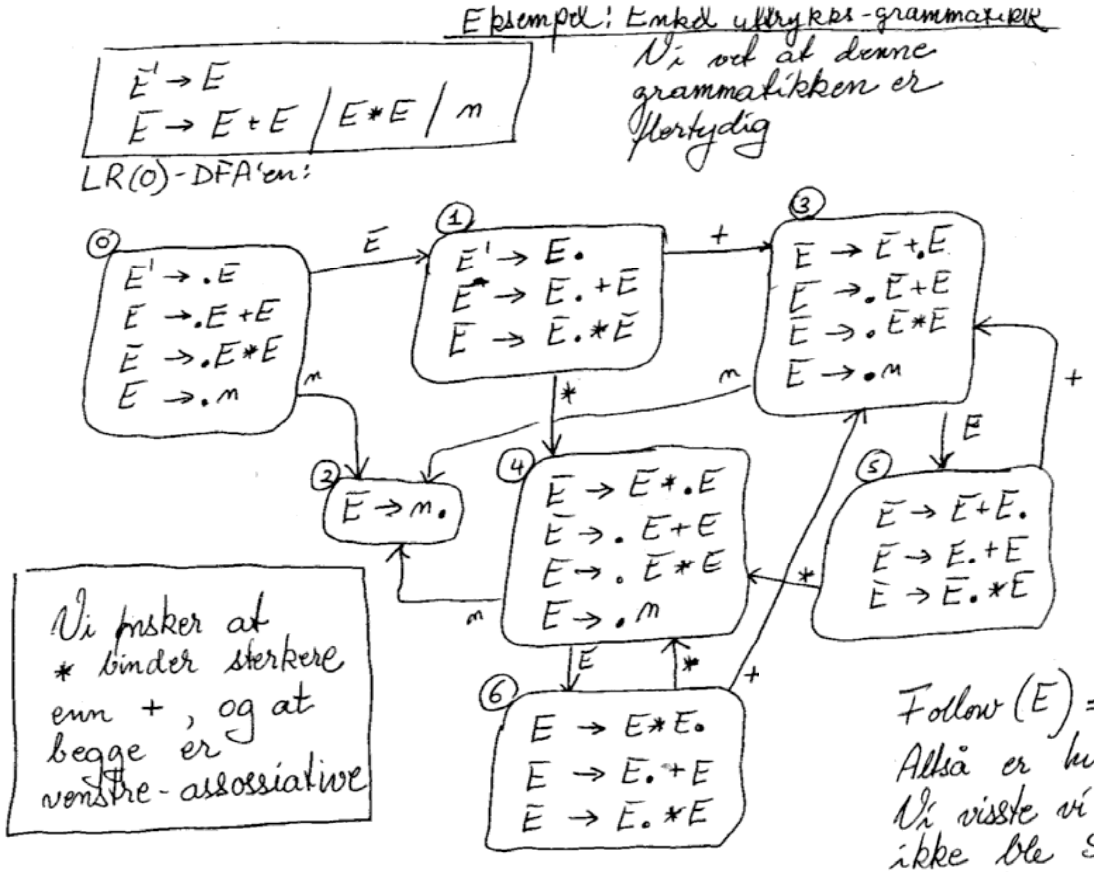
2g

	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	

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



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!

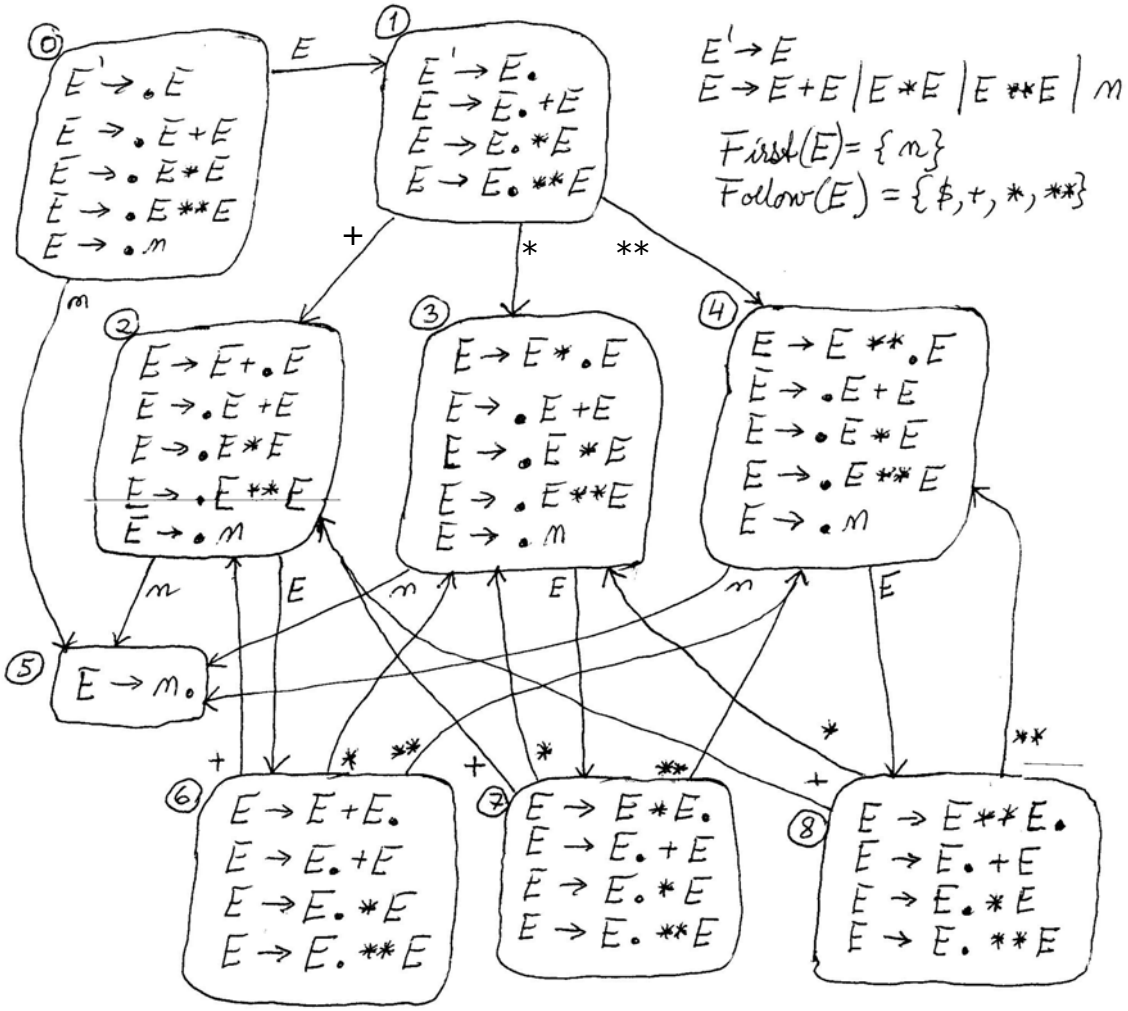
Fordel ved flertydige grammatikker:
De er som regel enklere å sette opp, se f.eks. til venstre her, og tidligere grammatikker for if-setningen

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

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

	m	+	*	$	E
0	s2				1
1		s3	s4	accept	
2		r(E→m)	r(E→m)		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

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

* skift 3

** skift 4

Tilstand 7

\+ red(E -> E*E)

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

** skift 4

Tilstand 8

\+ skift 2

* skift 3

** skift 4 (høyre-ass.)

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