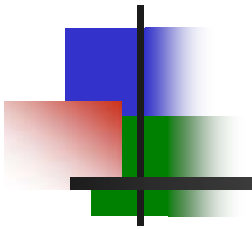


Kap. 5: Oppgaver m.m.

(Noen lysark fra tidligere er gjentatt her)

Stein Krogdahl,

Ifi, UiO

8. Mars 2007

Typisk Yacc-produsert parseringstabell

(merk påfyll av ekstra reduksjoner, som en plass-optimalisering i Yacc)

Grammatikk: command → exp

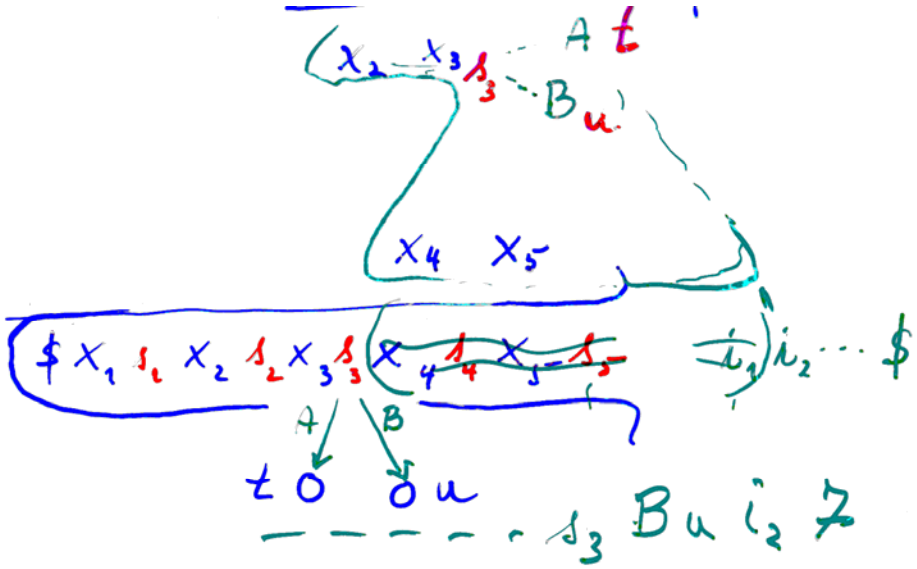
exp → exp + term | exp – term | term

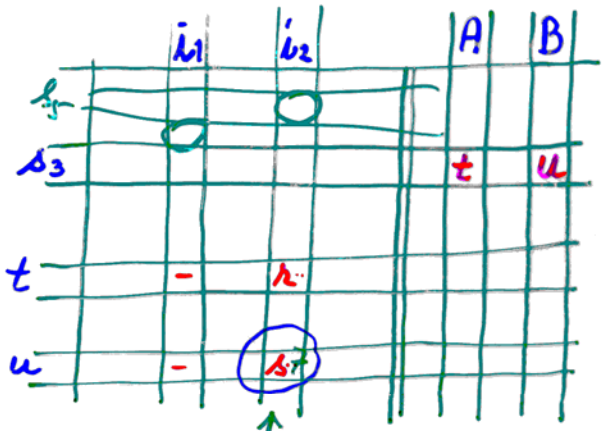
term → term * factor | factor

factor → NUMBER | (exp)

State	Input							Goto			
	NUMBER	(	+	-	*	)	$	*command*	*exp*	*term*	*factor*
0	s5	s6						1	2	3	4
1							accept				
2	r1	r1	s7	s8	r1	r1	r1				
3	r4	r4	r4	r4	s9	r4	r4				
4	r6	r6	r6	r6	r6	r6	r6				
5	r7	r7	r7	r7	r7	r7	r7				
6	s5	s6							10	3	4
7	s5	s6								11	4
8	s5	s6								12	4
9	s5	s6									13
10			s7	s8		s14					
11	r2	r2	r2	r2	s9	r2	r2				
12	r3	r3	r3	r3	s9	r3	r3				
13	r5	r5	r5	r5	r5	r5	r5				
14	r8	r8	r8	r8	r8	r8	r8				

Panic-mode for LR-parsering
(det eneste vi skal se på)





Velger til slutt å pushe på B, som etter å ha fjernet i_1 gir tilstand u

1. Pop states from the parsing stack until a state is found with nonempty Goto entries.
2. If there is a legal action on the current input token from one of the Goto states, push that state onto the stack and restart the parse. If there are several such states, prefer a shift to a reduce. Among the reduce actions, prefer one whose associated nonterminal is least general.
3. If there is no legal action on the current input token from one of the Goto states, advance the input until there is a legal action or the end of the input is reached.

i_1

s_3 (t og u)

} Ta vekk én og én input, og gjenta 2

Eksempel: if-setning er mindre generell enn setning

5.45

Panic-mode for LR-parsering kan gå i evig løkke

Vi bruker følgende grammatikk:

command → exp
exp → exp + term | exp – term | term
*term → term * factor | factor*
factor → NUMBER | (exp)

samt parseringstabellen som Yacc produserte for denne (tidligere lysark)

Parsering med feil input ”(n n)”:

$ 0	(n n) $
$ 0 (6	n n) $
$ 0 (6 n 5	n) $
$ 0 (6 F 4	n) $
$ 0 (6 T 3	n) $
$ 0 (6 E 10	n) $
$ 0 (6 F 4	n) $

... gjentar seg selv

Feil, siden [10, n] i parserings-tabellen er tom. Videre:

- 10 har ingen goto, så E 10 poppes av
- 6 har goto for E T F
- ville gå til tilstand 10 3 4
- ved input n gir dette - r4 r6

(ingen skift, dessverre!)

- Av T og F velger vi F, som er den minst generelle, og pusher derfor på F 4

Men da er vi tilbake til en tilstand vi har vært i (uten å ha lest noe input), og ting vil da bare gjenta seg selv.

Mulig løsning (meget løslig):

- Hold greie på om du kommer tilbake til samme tilstand, og gjør noe spesielt:
- Ta da mer bort fra stakken, og forsøk igjen
- Kanskje: *Forlange* en skift-mulighet for å sette i gang igjen

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.

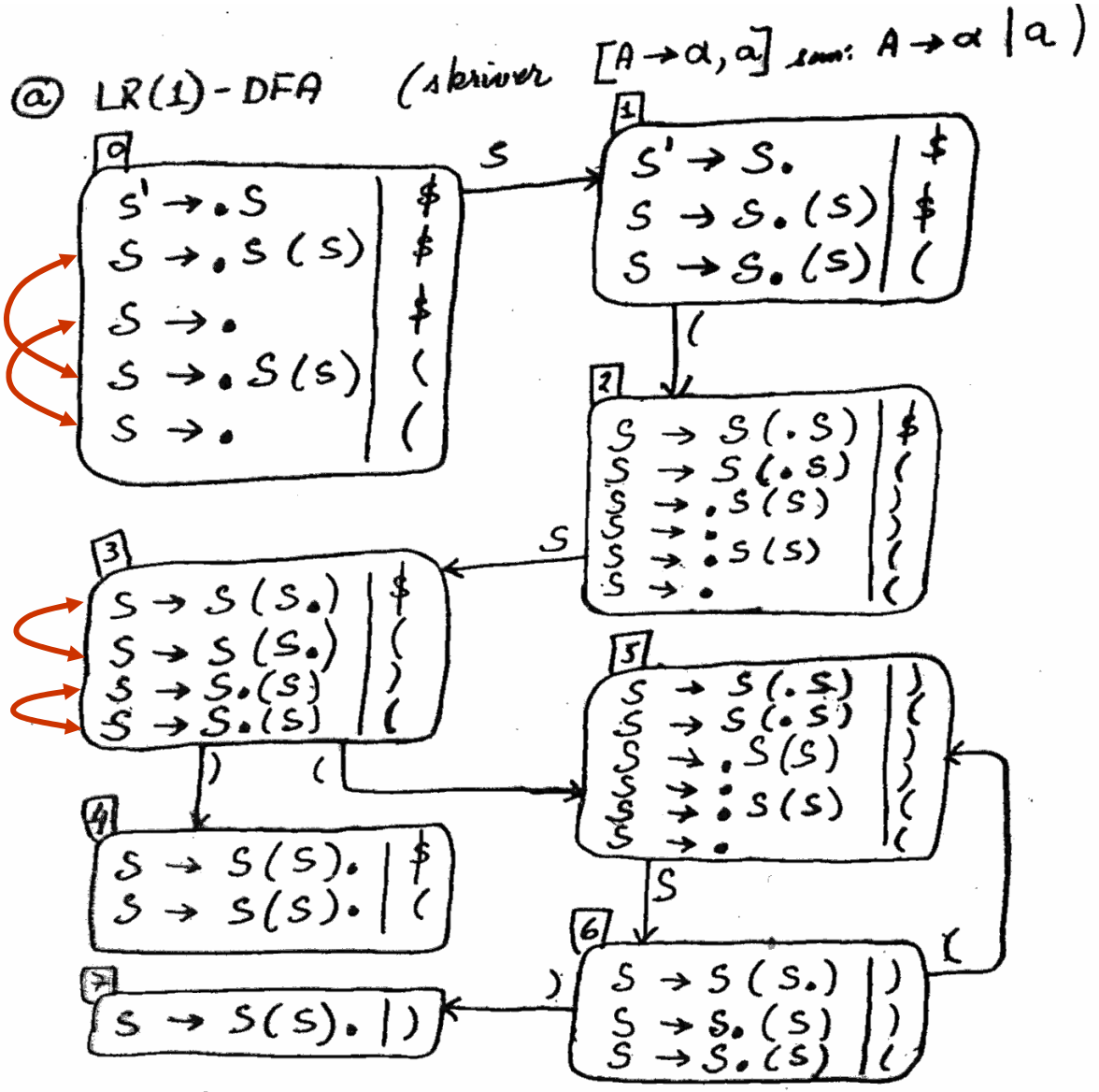
Oppgave 5.4 (a)

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

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

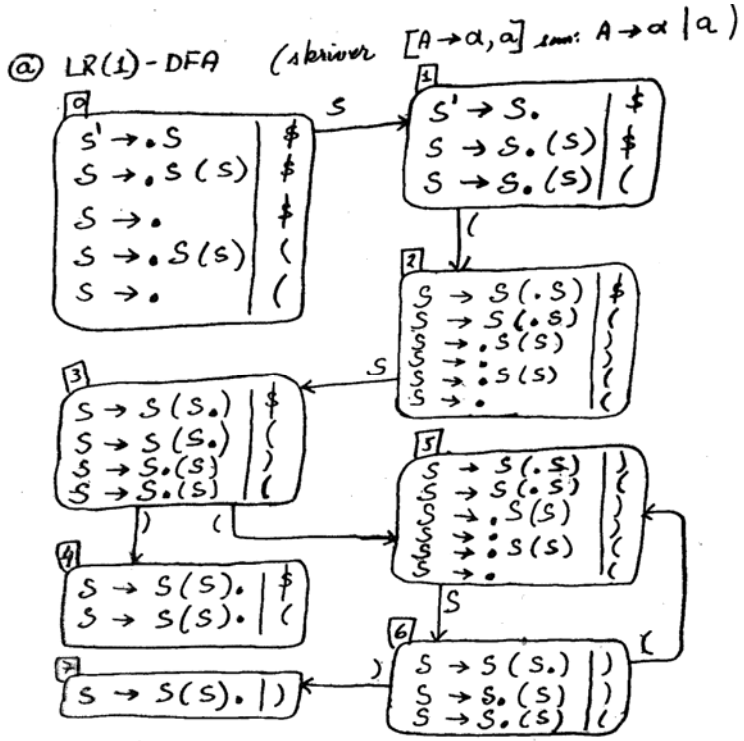
I LALR(1)-DFA’er vil man vanligvis skrive disse to som én linje, slik:

... S → S • (S) ...	) (



Oppgave 5.4 (a og b)

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

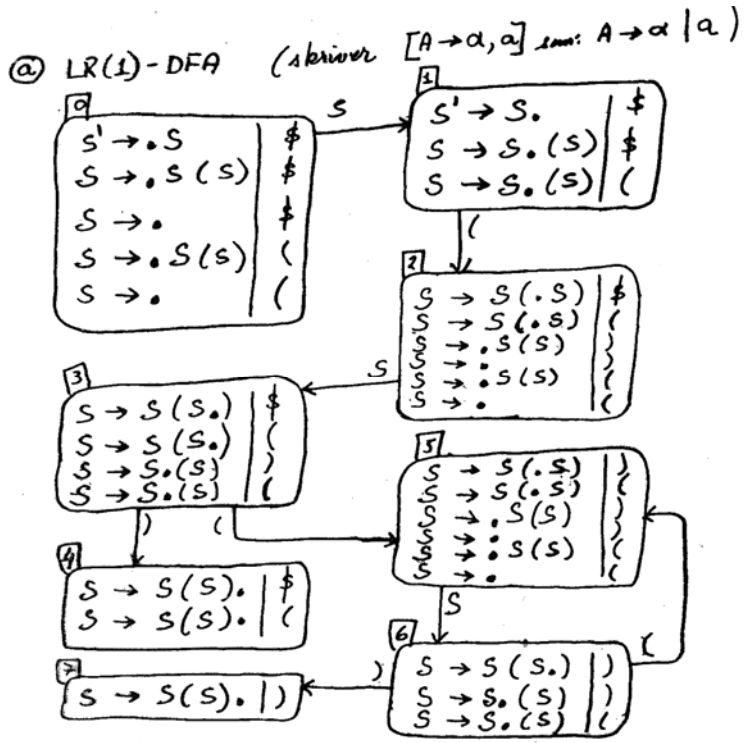


4)

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

Oppgave 5.4 (a og b)

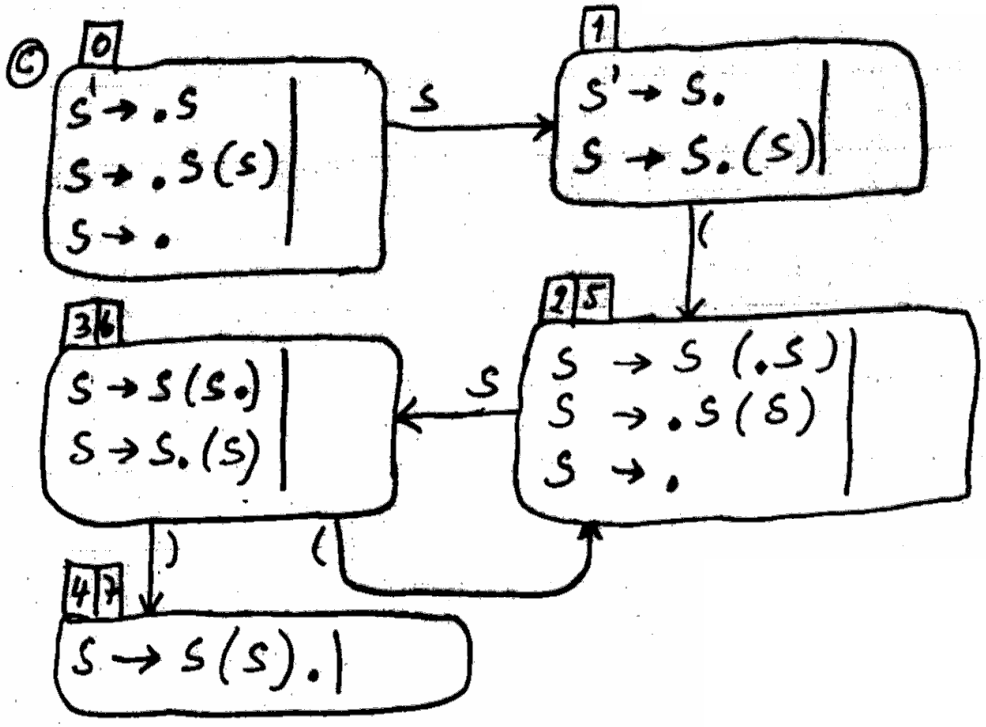
$$S \rightarrow S\ (\ S\ )\ |\ \varepsilon$$



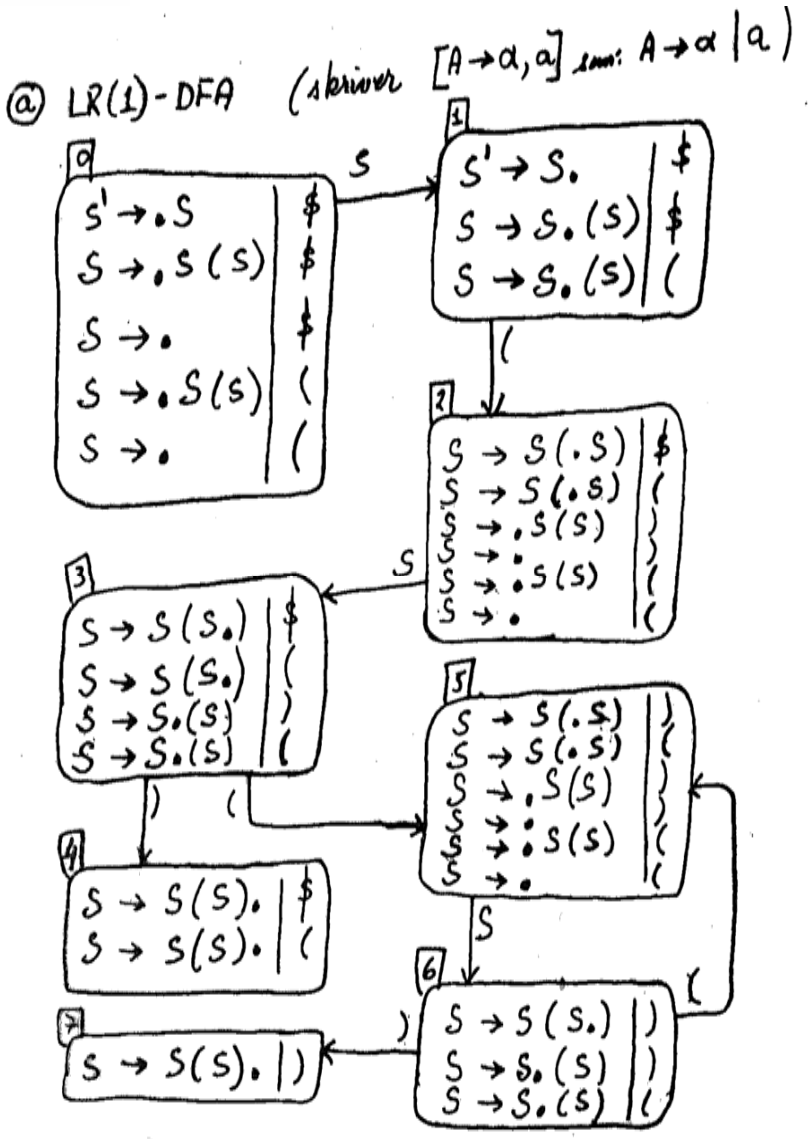
b)

	(	)	$	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))		

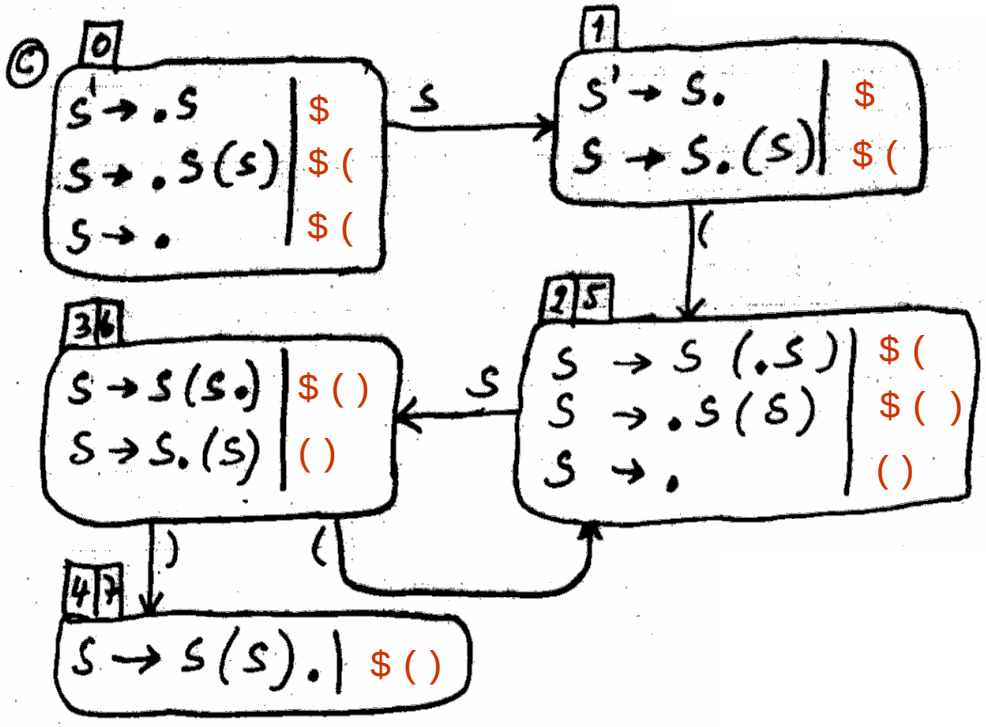
Oppgave 5.4 -c



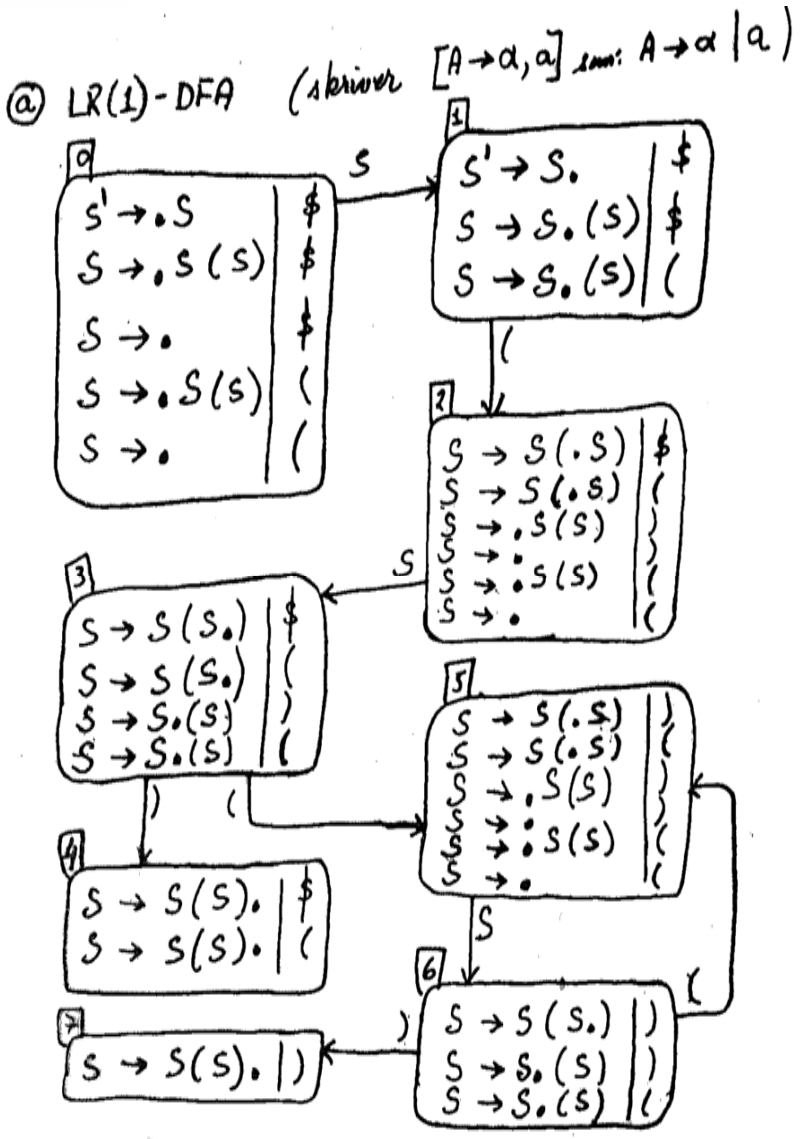
$S \rightarrow S\ (\ S\ )\ |\ \varepsilon$



Oppgave 5.4 -c



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

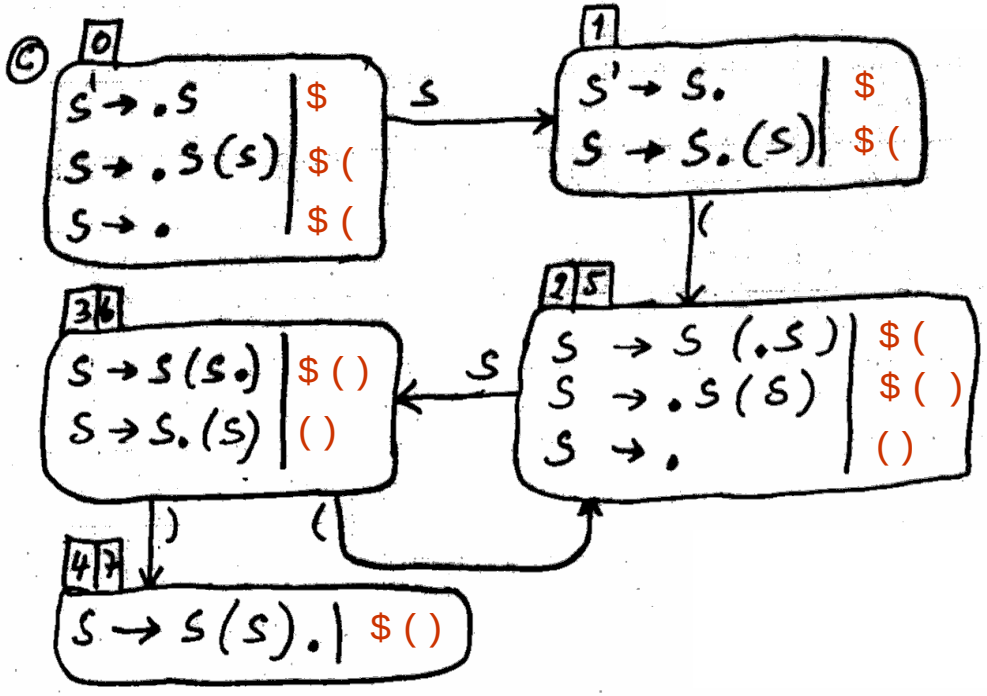


Follow(S) = { (,), $ }

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

(Men det får ikke betydning for entydigheten av parserings-tabellen, grammatikken er også SLR(1))

Oppgave 5.4 - d og e



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

Deloppgave (e):

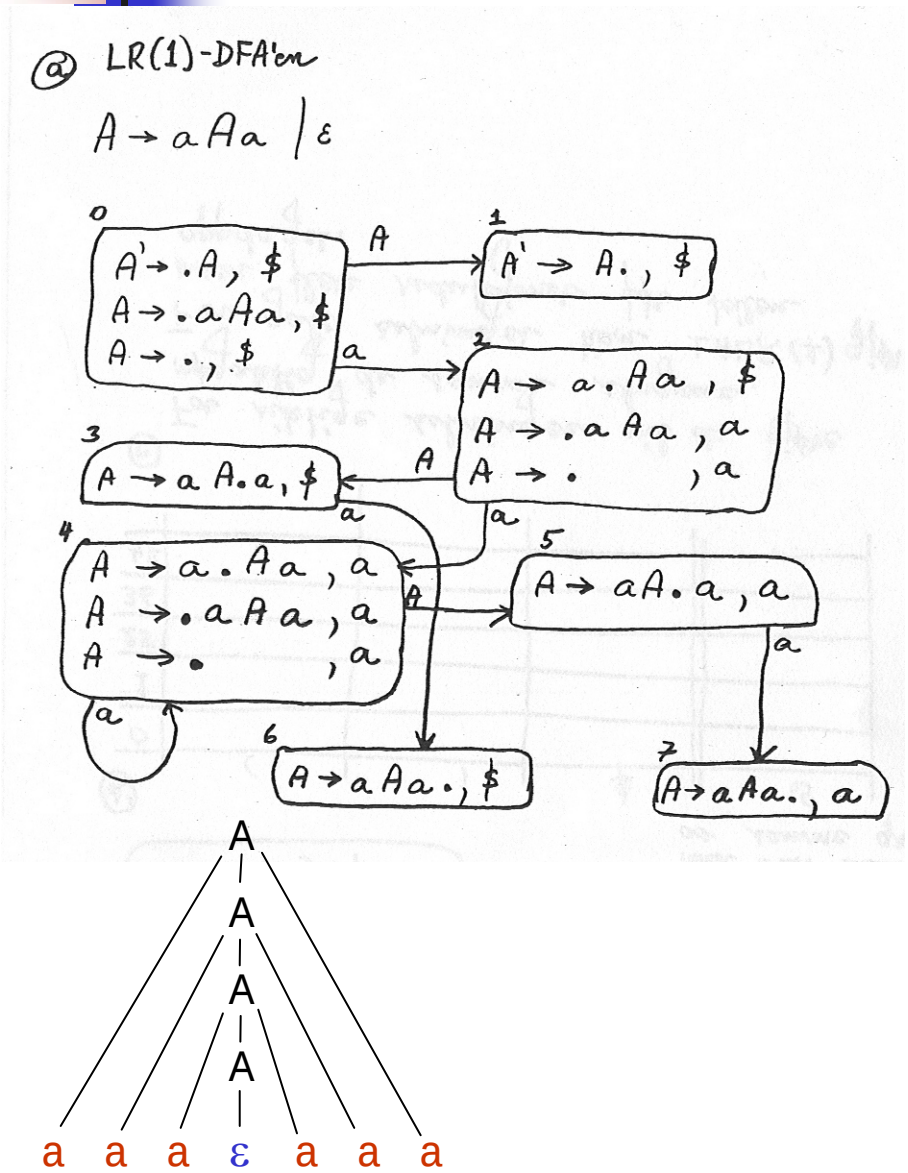
For riktige setninger vil LALR(1)-tabellen og LR(1)-tabellen gjøre nøyaktig de samme stegene.

For gale setninger kan LALR(1)-tabellen gjøre noen flere reduksjoner 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.

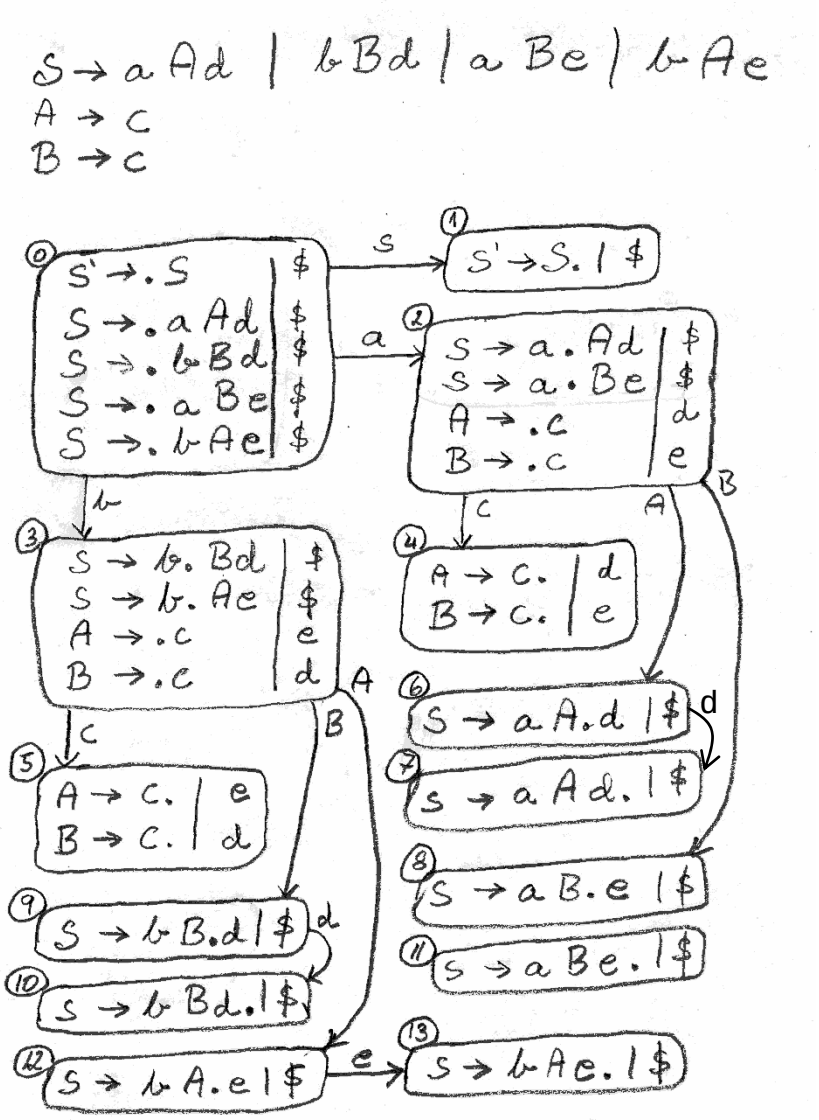
Andre grammatikker som gir de samme setninger, og som helt kurant er SLR(1)

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 der LALR(1)-DFA'en fikk en skift/red.-konflikt?

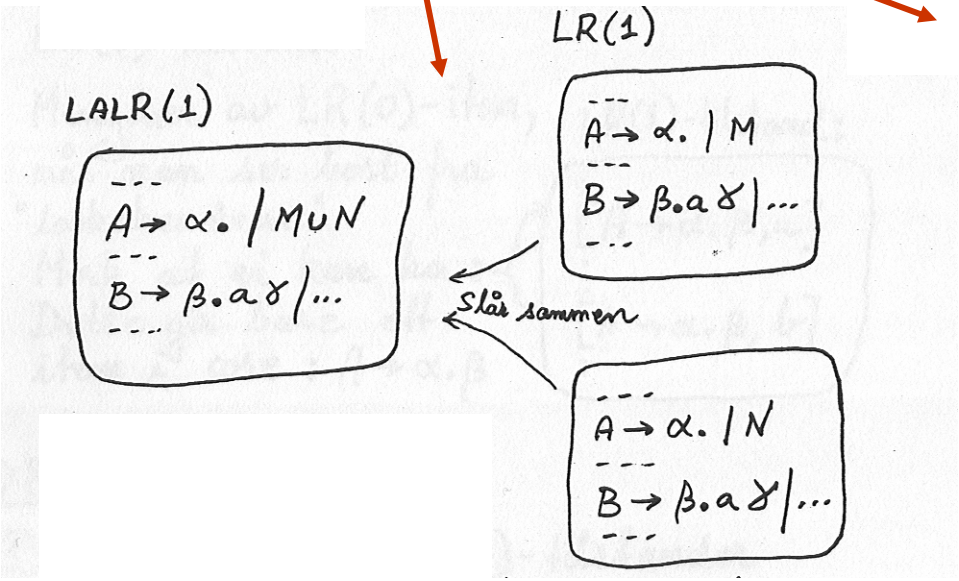
NEI, se oppg. 5.13 neste lysark

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)

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

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.



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.