4.8

LEXP → ATOM | LIST

ATOM → num | id

LIST → (LEXP-SEQ)

LEXP-SEQ → LEXPSEQ LEXP | LEXP

a) Grammar after removal of left recursion

LEXP → ATOM | LIST

ATOM → num | id

LIST → (LEXP-SEQ)

LEXP-SEQ → LEXP LEXP-SEQ'

LEXP-SEQ' → LEXP LEXP-SEQ' | ∈

b) FIRST + FOLLOW sets of non-terminals

	FIRST	FOLLOW
LEXP	{num, id, (}	{$,), num, id, (}
ATOM	{num, id}	{$,), num, id, (}
LIST	{(}	{$,), num, id, (}
LEXP-SEQ	{num, id, (}	{)}
LEXP-SEQ'	{∈, num, id, (}	{)}

c) Resulting grammar is LL(1).

LEXP → ATOM | LIST FIRST(ATOM) ∩ FIRST(LIST) = ∅

ATOM → num | id FIRST(num) ∩ FIRST(id) = ∅

LEXP-SEQ' → LEXP LEXP-SEQ' | ∈ FIRST(LEXP) ∩ FOLLOW(LEXP-SEQ') = ∅

⇒ the grammar is LL(1).

	num	id	(	)	$
LEXP	LEXP → ATOM	LEXP → ATOM	LEXP → LIST		
ATOM	ATOM → num	ATOM → id			
LIST			LIST → (LEXP-SEQ)		
LEXP-SEQ	LEXP-SEQ → LEXP LEXP-SEQ′	LEXP-SEQ → LEXP LEXP-SEQ′	LEXP-SEQ → LEXP LEXP-SEQ′		
LEXP-SEQ′	LEXP-SEQ′ → LEXP LEXP-SEQ′	LEXP-SEQ′ → LEXP LEXP-SEQ′	LEXP-SEQ′ → LEXP LEXP-SEQ′	LEXP-SEQ′ → ∈	

Stack	Input
$ LEXP	((id) num)$
$ LIST	"
$) LEXP-SEQ (	((id) num)$
$) LEXP-SEQ	(id) num)$
$) LEXP-SEQ′ LEXP	"
$) LEXP-SEQ′ LIST	(id) num)$

Stack	Input
$) LEXP-SEQ′) LEXP-SEQ (	(id) num)$
$) LEXP-SEQ′) LEXP-SEQ	id) num)$
$) LEXP-SEQ′) LEXP-SEQ′ LEXP	id) num)$
$) LEXP-SEQ′) LEXP-SEQ′ ATOM	id) num)$
$) LEXP-SEQ′) LEXP-SEQ′ id	id) num)$
$) LEXP-SEQ′) LEXP-SEQ′	) num)$
$) LEXP-SEQ′)	) num)$
	num)$

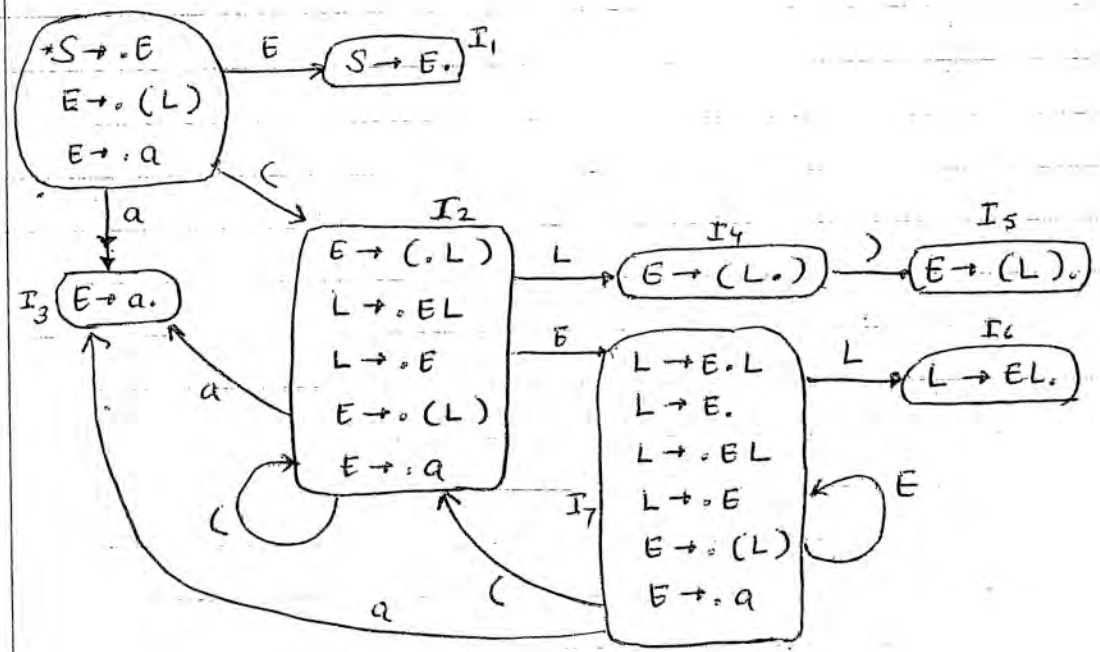
Stack	Input
$) LEXP-SEQ′ ATOM	num)$
$) LEXP-SEQ′ num	num)$
$) LEXP-SEQ′	)$
$)	)$
$	$

success.

SLR(1) parsing

$S \to E$
$E \to (L) \mid a$
$L \to EL \mid E$

FOLLOW(E) = { $,), a, (}
FOLLOW(L) = {) }



	a	(	)	$	E	L
I_0	s, I_3	s, I_2	—	—	I_1	—
I_1	—	—	—	accept	—	—
I_2	s, I_3	s, I_2	—	—	I_7	I_4
I_3	r, E→a	r, E→a	r, E→a	r, E→a	—	—
I_4	—	—	s, I_5	—	—	—
I_5	r, E→(L)	r, E→(L)	r, E→(L)	r, E→(L)	—	—
I_6	—	—	r, L→EL	—	—	—
I_7	s, I_3	s, I_2	r, L→E	—	I_7	I_6

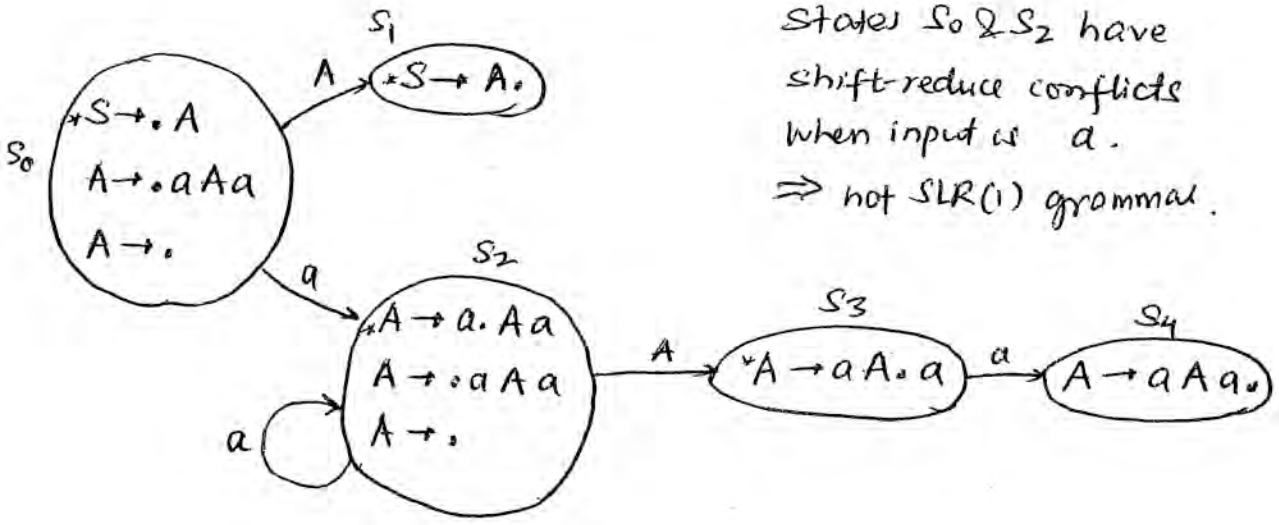
parse input (a) using the above tables.

SLR(1) & LR(1) parsing

$S \to A$

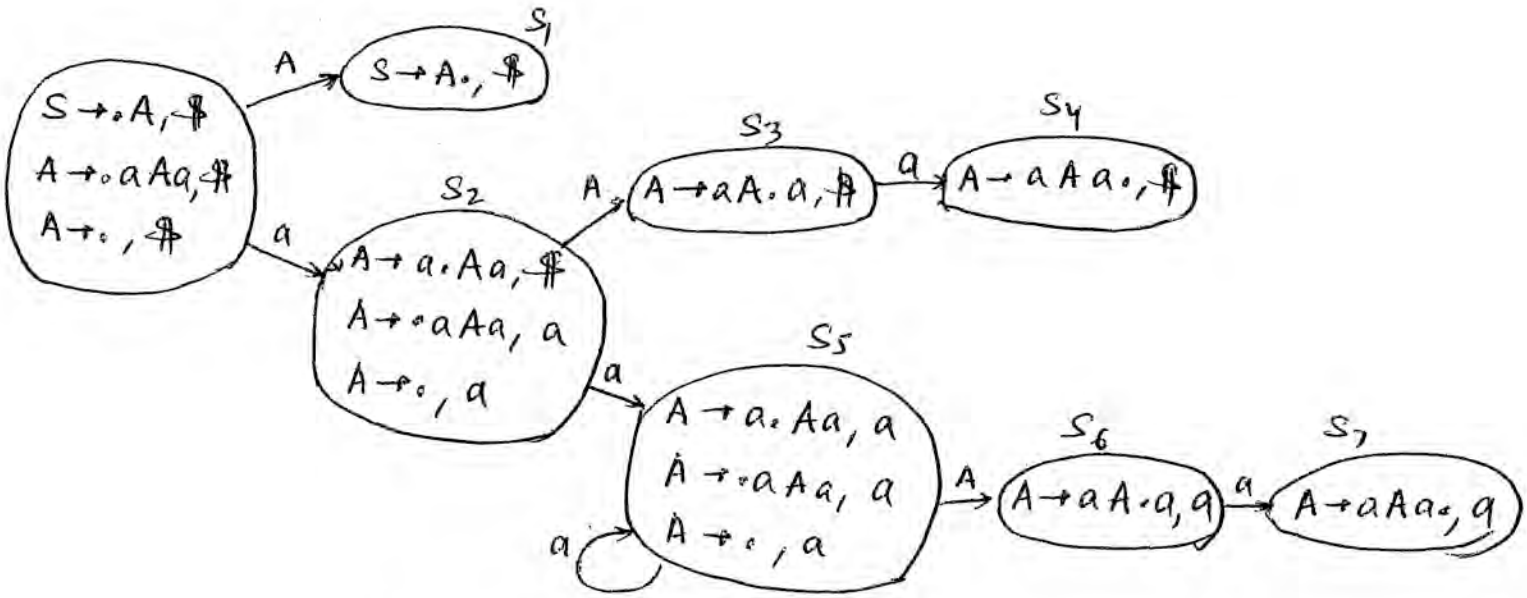
$A \to aAa \mid \varepsilon$

FOLLOW(A) = {$, a}



States S_0 & S_2 have shift-reduce conflicts when input is a.

⇒ not SLR(1) grammar.



States S_2 & S_5 have shift-reduce conflicts when input is a.

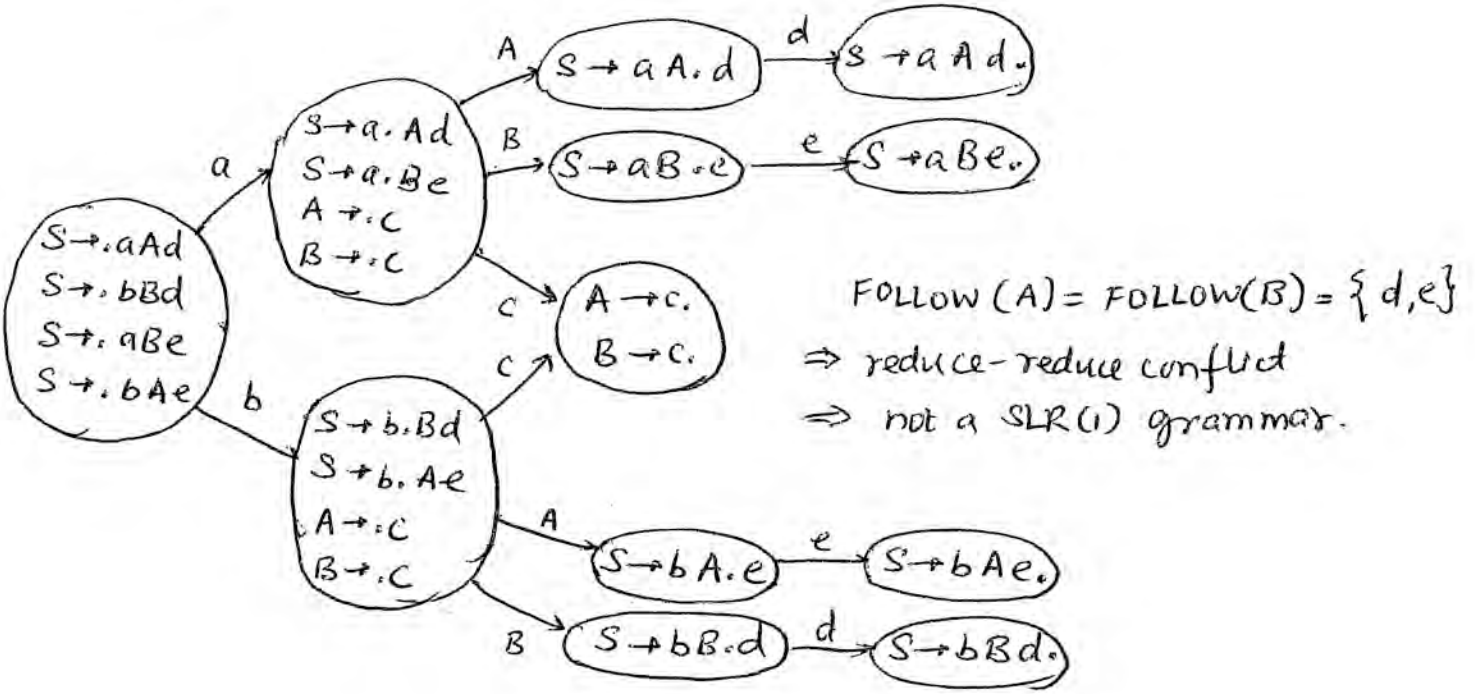
⇒ not LR(1) grammar

SLR(1), LALR(1) & LR(1) parsing

$S \to aAd \mid bBd \mid aBe \mid bAe$

$A \to c$

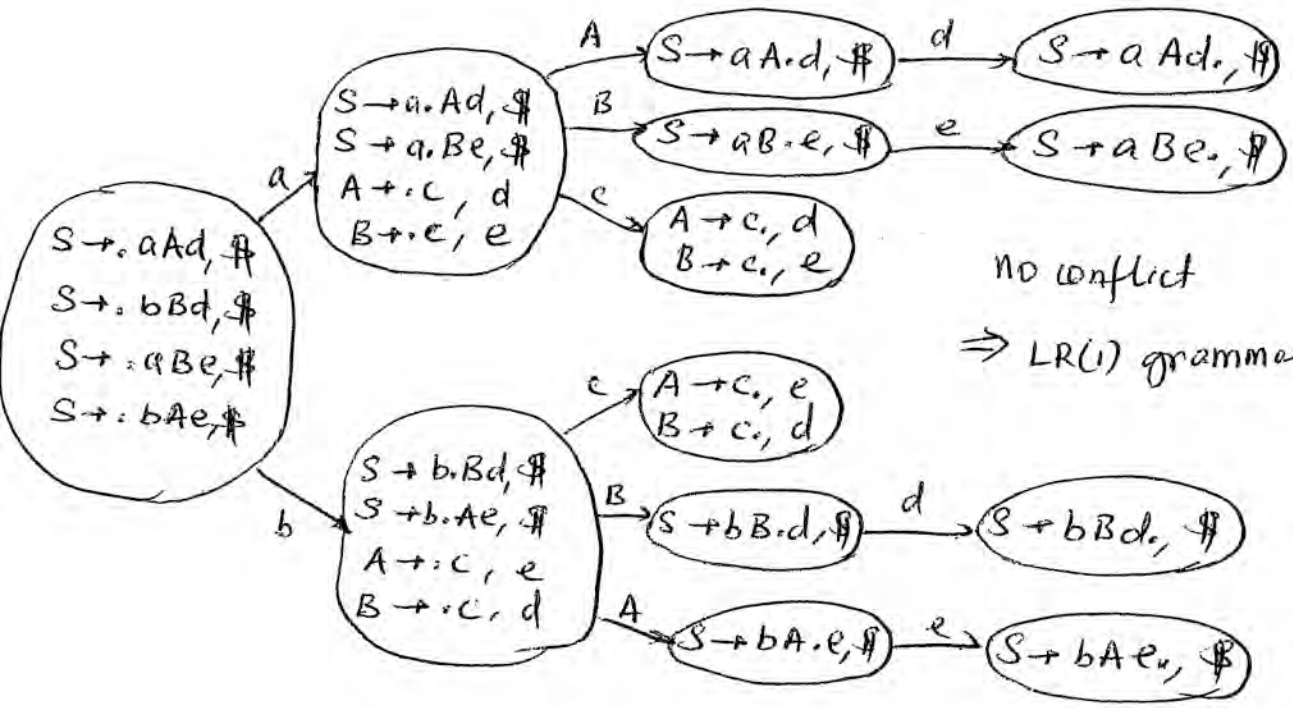
$B \to c$



FOLLOW(A) = FOLLOW(B) = {d, e}

⇒ reduce-reduce conflict

⇒ not a SLR(1) grammar.



no conflict

⇒ LR(1) grammar.

LALR(1) ??

A → c., d
B → c., e

A → c., e
B → c., d

merge

A → c., d/e
B → c., d/e

— reduce-reduce conflict

⇒ not LALR(1).

$S' \to S$
$S \to CC$
$C \to cC \mid d$