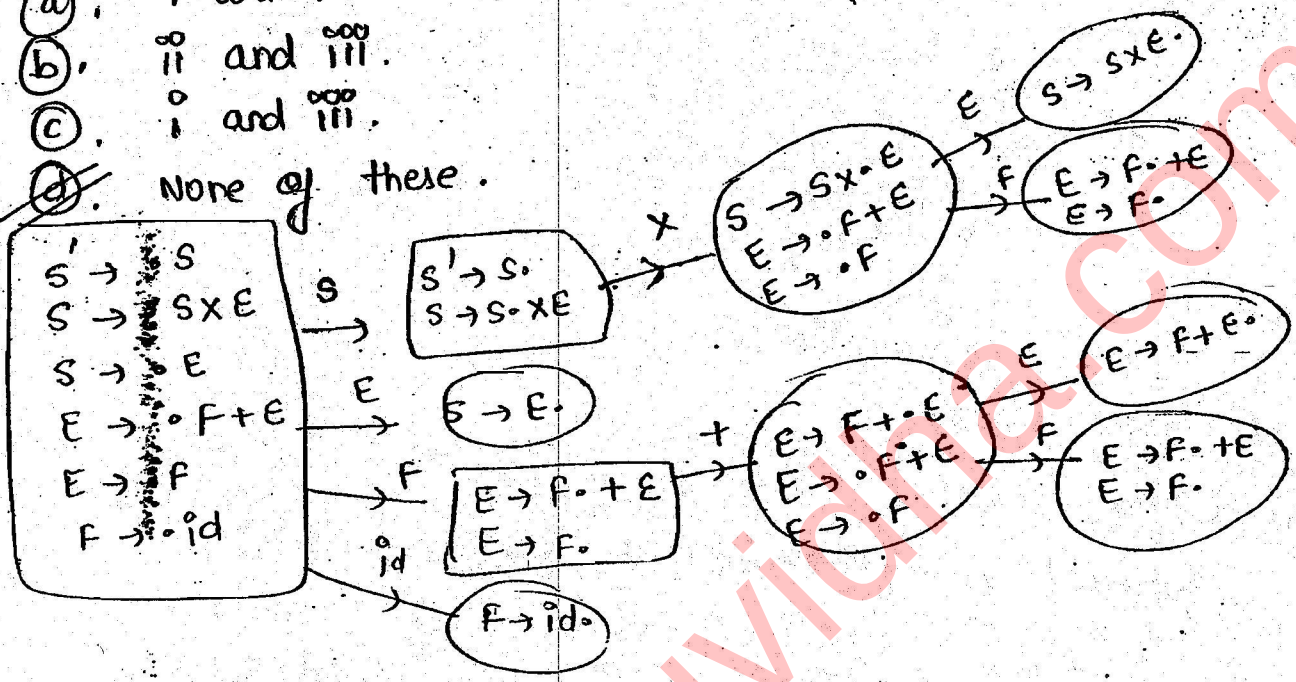
Given the items above which 2 of them will appear in the same set in the canonical sets of items for the grammar.

(a). i and ii.
(b). ii and iii.
(c). i and iii.
(d). None of these.



(d).
No, none of the above.

Canonical LR' (CLR)

CLR is the most powerful technique among all parsing techniques. It can parse any arbitrary context free grammar.

The principle drawback of this method is it required lot of hardwork.

CLR parsing table construction depends on canonical sets of LR(1) items.

A LR(1) item is basically LR(0) item with lookahead symbols. The general form of LR(1) item is.

$$\underbrace{A \to X \cdot YZ}_{LR(0)}, \underbrace{\{a, b\}}_{LookAhead}$$

The construction of LR(1) sets of item depends on 3 procedures

1. Augmented Grammar
2. closure function :- → If $A \to \alpha \cdot B\beta, \{a,b\}$ is in I_i and $B \to \alpha\gamma$ is B production then add $B \to \cdot\gamma, FIRST(\beta, \{a,b\})$ to I_i ;
3. Goto function.

SLR depends on LR(0)
CLR depends on LR(1)
↳ and LR(1) = LR(0) + lookahead symbol

Shortcut :- If β is ε then add same lookahead as above direct

$\frac{Y_1}{\beta, \{a,b\}} \quad \frac{Y_2}{}$

↓ aise consider kara hai.

Rule :- (for closure :-)

$$FT(\beta\{a,b\}) = \{a,b\} \text{ if } \beta = \varepsilon.$$

$$= FIRST(\beta\{a,b\})$$

e.g., $FT(\underset{Y_1}{\beta}\ \underset{Y_2}{\{a,b\}}) \rightarrow$

Goto function :- If $A \to \alpha \cdot X\beta, \{a,b\}$ is in I_i, then

$GOTO(I_i, X) = A \to \alpha X \cdot \beta, \{a,b\}$.

↓ ye as it is copy kar do then for further cal closure.

Q. Consider canonical sets of LR(1) set of item for the following grammar G – :

~~S → AB~~

Note – Construction of LR(1) sets of item in general begin with

$S' \rightarrow .S, \{\$\}$

$S \rightarrow AB$
$A \rightarrow Ba \mid b$
$B \rightarrow a$

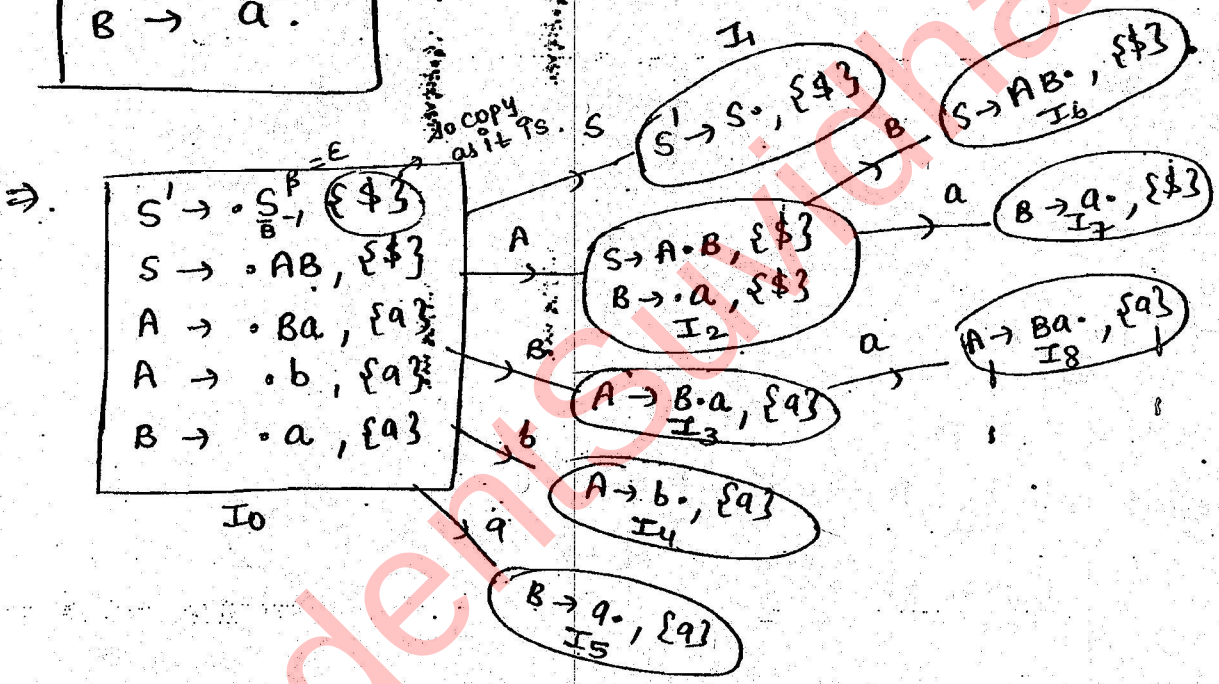


Table –

	ACTION			GOTO		
@	a	b	$	S	A	B
0	S5	S4		1	2	3
1			accept			
2	S7					6
3	S8					
4	r by A→b					
5	r by B→a					
6			r by S→AB			
7			r by B→a			
8	r by A→Ba					
9						

yaha par reduce ~~kar~~ hone par, lookahead symbol K neeche likh do, follow nikalne ki gnut nai hoti.

If CLR parsing donot have multiple entries then grammar is called LR(1) or CLR(1) grammar.

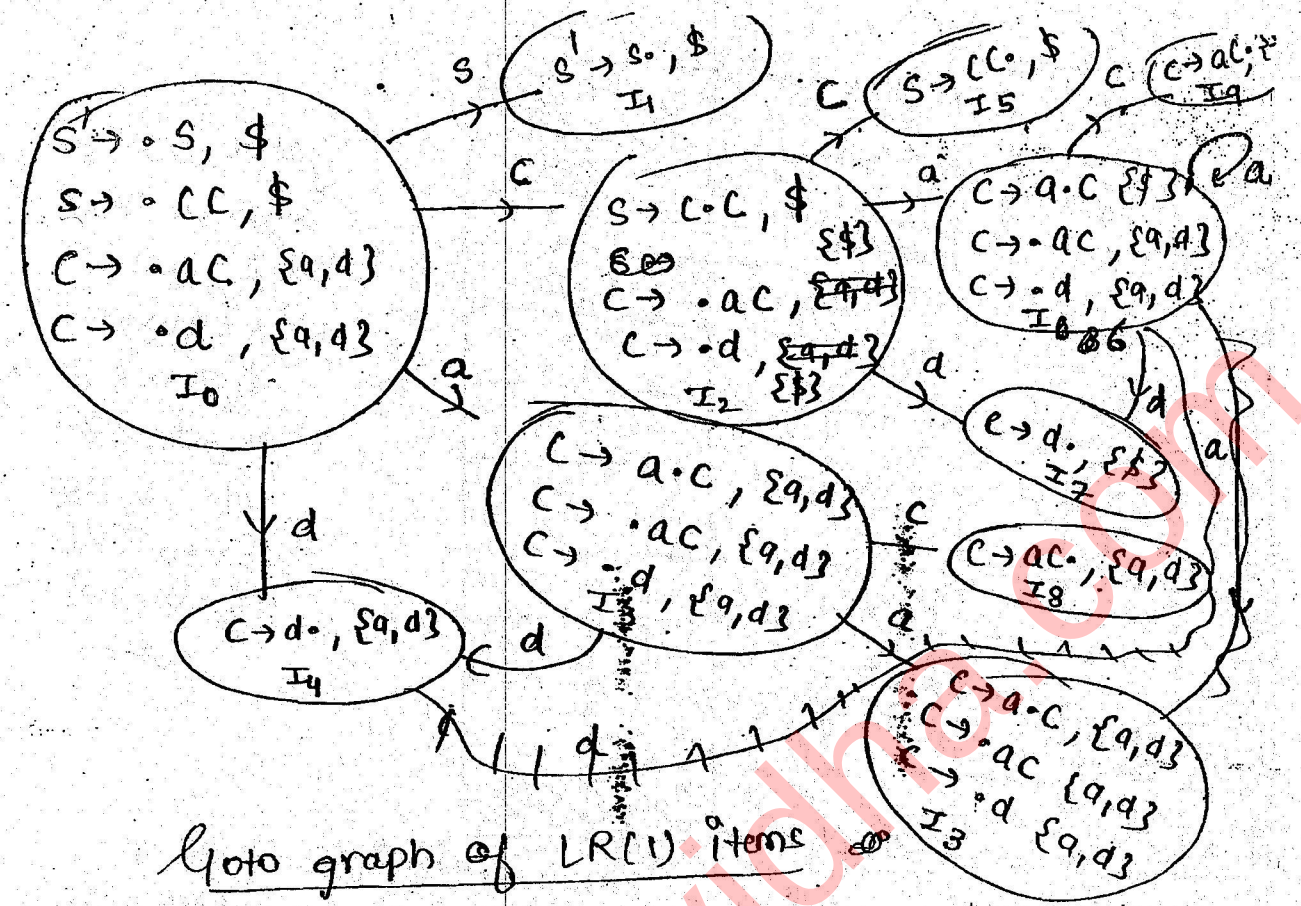
LR(1) or CLR(1) grammar :-

A grammar G1 is said to be CLR(1) if CLR(1) of parsing table do not contain multiple defin entries.

Q. find LR(1) or not ?.

$S \rightarrow CC$

$C \rightarrow aC \mid d$.



Goto graph of LR(1) items

CLR Table

	a	d	$	S	C
0	S_3	S_4		1	2
1			ACCEPT		
2	S_6	S_7			5
3	S_3	S_4			8
4	r by C→d	r by C→d			
5			r by S→CC		
6	S_6	S_7			9
7			r by C→d		
8	r by C→aC	r by C→aC			
9			r by C→aC		

Q1 Even CLR parser can not contain parse all grammar,

Ans If CLR grammar is LL ambiguous then CLR also fails.

No, parser can passe ambiguous grammar.

CLR occupies more space.
↳ so LALR comes.

LOOKAHEAD LR (LALR)

(1) In LALR parsing table construction, we construct a GOTO graph for canonical sets of LR(1) items as we have done in CLR technique.

(2) Now, identify sets of items which have having common first component and differentiating only in second component.

(3) We merge those states / rows in CLR parsing table to obtain LALR(1) parsing table.

for eg., in the GOTO graph of the grammar

$S \to CC$
$C \to ac \mid d$

LR(1) sets of items, the states I_3 and I_6 have the common first component differentiating in second component Hence, they merge it as I_{36}.

Similarly, I_4 & I_7 = I_{47}
I_8 & I_9 = I_{89}.

LALR table for above grammar –

	a	d	$	S	C
0	S36	S47		1	2
1			accept		
2	S36	S47			5
36	S36	S47			89
47	r by C→d	r by C→d	r by C→d		
5			r by S→cc		
89	r by C→ac	r by C→ac	r by C→ac		

LALR(1) parsing table

* LALR(1) grammar –

A grammar G, is said to be LALR(1) if LALR(1) parsing table do not contains multiple entries.

Even though we can't have ~~more~~ multiple entries into CLR table, but we can have multiple entries in LALR table due to merging ~~Error~~, so CLR is more powerful.

~~* LALR and~~ Every LALR(1) is also CLR(1).

every LR(0) is also SLR(1).

Reverse may not true.

* Main Points :-

(1). The parsing table sizes of SLR and LALR techniques are always same for any arbitrary context free grammar

(2). CLR parsing table size is always larger than SLR / LALR table size. (same bhi ho sakte hai) but good relation is greater

(3). In LALR, SLR tables shift entries are always identical. → kuki dono K liya rule same hi hai.

(4). GOTO entries of both LALR, SLR are alwa identical.

(5). Reduce entries of the SLR and LALR tables may be different [may or may not be identical]. ↳ [kuki nikalne K rule alag hai]

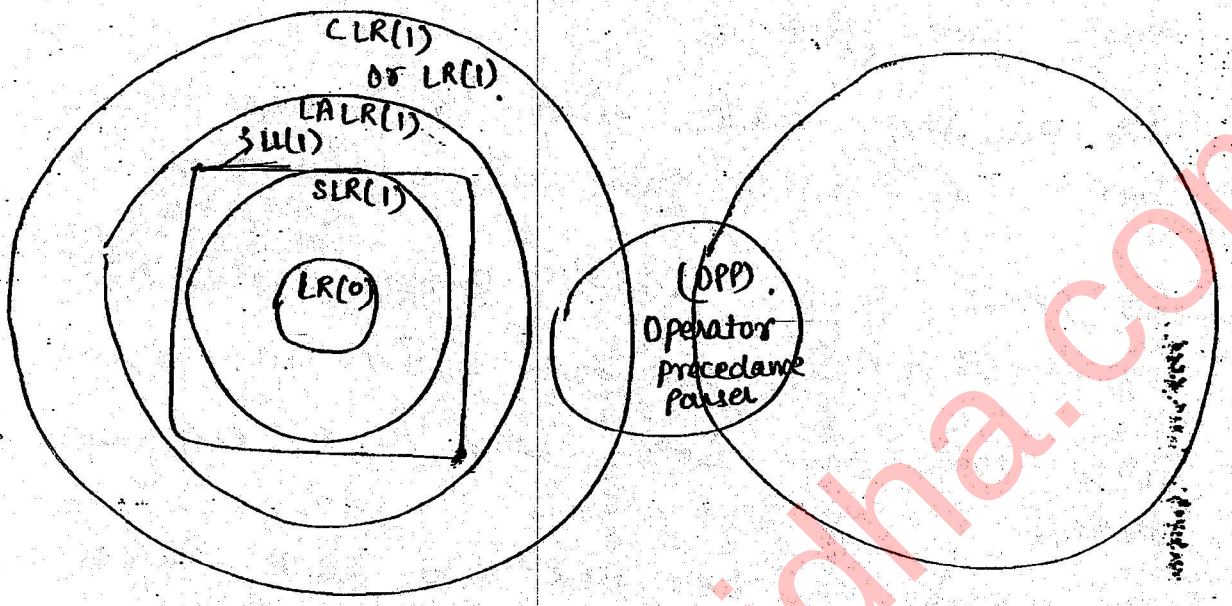
(6). Error entries of the both the tables may be different. ↳ kuki Reduce entries change ho sakti hai.

* agar sari language K beech check karna ho tho bs LR(1) goto graph bnao, bs fir reduce vale rule alag kaik dekho.

LL(1) is subse of LALR(1).

Relationship, among the grammar - :

unambigious ambigious.



$$LR(0) \subseteq SLR(1) \subseteq LALR(1) \subseteq CLR(1)$$
$$LL(1) \subseteq LALR(1) \subseteq CLR(1)$$

LR(0), SLR(1) and LL(1) mai koi direct relation nai hai.

* RR conflict in LR(1) item.

$A \to \alpha.$ {a, b}
$B \to \beta.$ {a, d}

@ → dono ki intersection ϕ nai hai tho error hogi.

* SR conflict in LR(1) item.

$A \to \alpha.a\beta$, {a, b}
$B \to \gamma.$ {a, b}

dono ki intersection ϕ nai hai tho ki error hogi.

every time construct CLR karo.

Q. The Grammar $S \to aSa \mid bS \mid c$ is

Ans
(a). LL(1) but not LR(1)
(b). LR(1) but not LL(1)
(c). Both LL(1) and LR(1).
(d). Neither LL(1) nor LR(1).

Sol. This is LL(1). as by checking all possible combination of ~~FIRST(S)~~. ~~for~~ ~~aa~~ first of each production of S.

and since it is LL(1) so definately LR(1).

If above given language is changed as -:

$S \to aSa \mid bS \mid c \mid \epsilon$

So, not LL(1).

as it contains multiple entries.

	a	b	c	ε
S	S→aSa S→ε	S→bS	S→c	S→ε

→ also not LR(1)

as follow(S) = $\{\$\} \cup \{a\}$

$\Rightarrow \{\$\}, \{a\}$

~~If is unambiguous grammar, so definately LR(1)~~

Q. Consider the grammar $S \to (S) \mid a$.
Let no. of States in SLR(1), LR(1) and LALR parser for the grammar be n_1, n_2 & n_3 respectively. Which of the following relations hold good?

(a). $n_1 < n_2 < n_3$
(b). $n_1 = n_3 \leq n_2$
(c). $n_1 = n_2 = n_3$
(d). $n_1 > n_2 > n_3$.

Q. Assume that SLR parser for G_1 has n_1 states and LALR parser for G_1 has n_2 states the relationship b/w n_1 and n_2 is -

(a) n_1 is necessarily less than n_2.

(b) n_1 is necessarily equal to n_2.

(c) n_1 is necessarily greater than n_2.

(d) none of these.

Q. Consider the following 2 sets of LR(1) items of LR(1) grammar. [2013]

$X \to c \cdot X, c/d$	$X \to c \cdot X, \$$
$X \to \cdot cX, c/d$	$X \to \cdot cX, \$$
$X \to \cdot d, c/d$	$X \to \cdot d, \$$

Which of the following statement is related to merging of 2 sets in the corresponding parses is/are false?

(a)

S1. Can't be merged since look ahead symbol are different.

S2. Can be merged but result will in SR conflict

S3. Can be merged but result in RR conflict.

S4 Can not be merged since goto in c will Leads to the diff. Sets.

(a) I only

(b) II only

(c) I and IV only

(d) I, II, III, IV only.

Since all statements are wrong.

Q. Among Simple LR (SLR), Canonical LR (CLR), Lookahead LR (LALR)
e2015. Which of the following pair identifies the method which is very easy to implement and the method which is more powerful in order

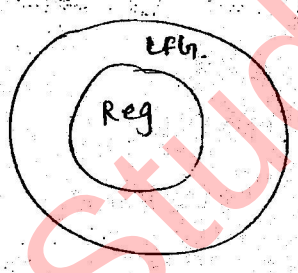
(a) SLR, LALR
(b) CLR, LALR
(c) SLR, CLR
(d) LALR, CLR

Q. Consider the following 2 statements :
S1. Every Regular grammar is a LL(1) grammar
S2. Every Regular set has a LR(1) grammar.
Which of the following statement is True ?

(a). Both (1) and (2)
(b). (1) is true (2) is false
(c). (1) is false (2) is true
(d). Both (1) and (2) are false.

Regular language
$A \to w \mid wB$
(or)
$S \to a \mid as$

Do not LL(1) because left factoring is possibility in Regular Grammar.



Q.

~~CLR is~~ Powerful

CLR > LALR > SLR > LR(0)

If LR(K), K is not mention, by default it is 1.