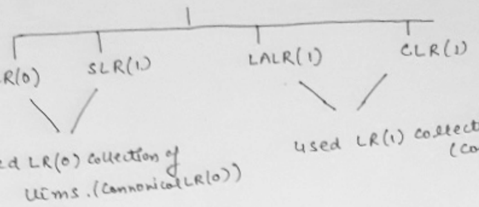
LR parsing Table consists of two part :- ACTION and GOTO

LR parser (Fig: 1)



used LR(0) collection of items. (Canonical LR(0))

used LR(1) collection of items (Canonical LR(1))

Steps for constructing LR Table :

(1) Create an Augmented grammar G_1' for a given context free grammar G_1.

(2) Create canonical collection of item using GOTO and closure operations.

(3) Design a parsing table using canonical collection set.

Step 1 (Explanation) : create an Augmect grammar G_1'

If G_1 is a grammar with start symbol S then G_1' be Augmented grammar for G_1, it G_1 with a new start symbol S' and production $S' \to S$

Ex 1 : G_1: $S \to AB$
$A \to a$
$B \to b$

→ Augmented Grammar G_1'

G_1': $S' \to S$
$S \to AB$
$A \to a$
$B \to b$

Ex2:

G: $S \to AA \mid aA \mid b$ → Augmented Grammar G' → G': $S' \to S$, $S \to AA$, $A \to aA \mid b$

Ex3:

$E \to E+T \mid T$
$T \to T*F \mid F$
$F \to (E) \mid id$

→ Augmented Grammar G' →

G': $E' \to E$, $E \to E+T \mid T$, $T \to T*F \mid F$, $F \to (E) \mid id$

Step 2: (Explanation): create a canonical collection of item (First we Read LR(0) collection of items which is Needed in LR(0) and SLR(1) parser Fig:1)

LR(0) item of a Grammar G is a production of G with a 'DOT' at some position of body

Thus $A \to XYZ$ yields the four item

$A \to .XYZ$
$A \to X.YZ$
$A \to XY.Z$
$A \to XYZ.$

} in the Right hand side DOT is moving from left to Right

If $A \to \varepsilon$ (A produce epsilon is a production) then we generate only item in LR(0) collection of item.

$A \to \varepsilon$ so $\boxed{A \to .}$ is a production in LR(0) collection of item.

$A \to \varepsilon$ $\boxed{\begin{matrix} A \to .\varepsilon \\ A \to \varepsilon. \end{matrix}}$ → This is Incorrect ε Mean empty and DOT is not Moved in an empty ~~str~~ Production so. $A \to .$ is a production in LR(0) collection of item.

Closure of item set

If I is a set of items for grammar G, then closure(I) is the set of items constructed from I by two Rules :-

(1) Initially, add every item in I to closure(I).

(2) If $A \to \alpha . B\beta$ is in closure(I) and $B \to \gamma$ is a production (Gamma), then add the item $B \to .\gamma$ to closure(I) if it is not already there. Apply this Rule until no more new items can be added to closure(I).

Example Consider a grammar G

$S \to AA$

$A \to aA \mid b$

Find the closure of each item

Soln Augmet Grammar G' : $S' \to S$

$S \to AA$

$A \to aA \mid b$

C = Closure of $(S' \to .S)$

I_0 : First item

$S' \to .S$
$S \to .AA$
$A \to .aA \mid .b$

(Dot is Before S so add product of S with Dot on left hand side)

(Dot is Before A so add productions of A with Dot on left hand side)

GOTO function :

If I is a set of items & X is a grammar symbol GOTO(I,X) is defined as closure of the set of all $[A \to \alpha X \cdot \beta]$ such that $[A \to \alpha \cdot X \beta]$ is in I. The GOTO function is used to defined transition in the LR(0) automaton for grammar

Ex: I_0: $A \to \alpha \cdot X \beta$ $\xrightarrow{goto(I_0, X)}$ $A \to \alpha X \cdot \beta$

Ex: I_2: $A \to \cdot aA$ $\xrightarrow{goto(I_2, a)}$ $A \to a \cdot A$

Ex I_3: $A \to \cdot b$ $\xrightarrow{goto(I_3, b)}$ $A \to b \cdot$

Question G_1: $S \to AA$
$A \to aA$
$A \to b$

Find a LR(0) collection of item

→ Find Augmented Grammar G_1':
$S' \to S$
$S \to AA$
$A \to aA$
$A \to b$

→ LR(0) Collection (Method 1 to Find LR(0) collection)

C = Closure ($S' \to \cdot S$)

I_0 : $S' \to \cdot S$
$S \to \cdot AA$
$A \to \cdot aA$
$A \to \cdot b$

goto GOTO (I_0, S) = Closure ($S' \to \cdot S$)

$I_1 \Rightarrow S' \to S \cdot$

GOTO (I_0, A)
= Closure ($S \to A \cdot A$)

I_2: $S \to A \cdot A$
$A \to \cdot aA$
$A \to \cdot b$

DOT is Before A
so add production of A
with DOT on left hand side.

GOTO(I_0, a) ⇒ Closure (A → a.A)

I_3: A → a.A
A → .aA
A → .b

→ DOT is Before A so add production of A with DOT on left hand side

GOTO(I_0, b) ⇒ Closure (A → b.)

I_4: A → b.

→ No production is Need to add Becoz Nothing is present on left hand side of DOT.

GOTO(I_2, A) ⇒ Closure (S → AA.)

I_5: S → AA.

GOTO(I_2, a) ⇒ Closure (A → a.A)

I_3: A → a.A
A → .aA
A → .b

→ DOT is Before A so add production of A with DOT on left hand side

Here we write I_3 Becoz same production with DOT is present in the item I_3 so New item none is not given.

GOTO(I_2, b) ⇒ Closure (A → b.)

I_4: A → b.

GOTO(I_3, A) ⇒ Closure (A → aA.)

I_6: A → aA.

GOTO(I_3, a) ⇒ Closure (A → a.A)

I_3: A → a.A
A → .aA
A → .b

→ LR(0) collection (Method 2 to Find LR(0) collection)

G^1: $S' \to S$

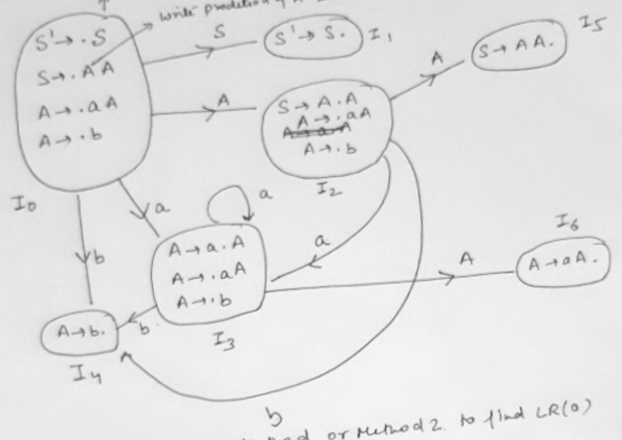
$S \to AA$

$A \to aA$

$A \to b$

Closure of $(S' \to .S)$ dot is Before S so write production of S with Dot on left hand side

write production of A with Dot on left hand side



✱ So this is Automaton Type Method or Method 2. to find LR(0) collection of item

✱ $I_0 \xrightarrow{S}$ Mean GOTO function - GOTO (I_0, S)

$I_0, I_1, I_2, I_3, I_4, I_5, I_6 \Rightarrow$ Collection of item set

✱

LR(0) parsing Table

1. Total No. of items in the previous question is ($I_0, I_1, I_2, I_3, I_4, I_5, I_6$) so that No. of state in LR(0) parser is 0 to 6
2. There are two parts in the Table — ACTION (used for terminal) — GOTO (used for variable)
3. $ is also a part of Terminal

	ACTION			GOTO	
	a	b	$	S	A
0	S3	S4		1	2
1			Accept		
2	S3	S4			5
3	S3	S4			6
4	r3	r3	r3		
5	r1	r1	r1		
6	r2	r2	r2		

→ In I_0 state on 'a' I/P we go into I_3 so we write S_3, S_3 mean shift on state 3
→ In I_0 state on 'b' I/P we go in I_4 so we write S_4, S_4 means shift on state 4
→ In I_0 state on S we go into I_1 (1) and on A we go into I_2 (2) means, number
→ In I_1 state we always have the production in which DOT is at the almost right on the start symbol $S' \to S.$ so we write Accept Action on I_1, $
start symbol
→ To do Reduce entry we Find the state in which DOT is at the almost right. So $I_1 (S' \to S.)$ $I_5 (S \to AA.)$ and $I_6 (A \to aA.)$ so I_1 is for Accept entry.

I_5, I_4, I_6 are the state for which we do Reduce entry.

→ So Arrange the production with serial No

1. $S \to AA$
2. $A \to aA$
3. $A \to b$

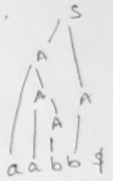
(Note: Augmented Grammer is Not a part of original Production)

→ In state I_5 ($S \to AA.$ So production No-1) we do Reduce entry r_1 in state 5 (I_5) to all terminal. This entry is also called Blind entry, Bcoz we are NOT seeing any terminal.

→ In state I_6 ($A \to aA.$ So production-2) we do Reduce entry r_2 in state 6 to all ~~Production~~ terminals.

→ In state I_4 ($A \to b.$ So production-3) we do Reduce entry r_3 in state 5 to all ~~Product~~ terminals.

PARSING OF STRING BY PARSETABLE :



I/P string : a a b b $ ($ end of string)

0	a	3	a	3	b	4	A	6	A	6	A	7	5	4	A	8	$	1

→ STACK GROWS

Start with state 0 (initial state) & I/P string left to Right

$0 \xrightarrow{a} S3$ (Shift a)

$3 \xrightarrow{a} S3$

$3 \xrightarrow{b} S4$

$1 \xrightarrow{\$}$ Accpt

$4 \to b$ (r_3) so production-3 ($A \to b$)

Right side length of that production is One. Double the length & pop the elem from the stack & push A on the stack

1. $S \to AA$
2. $A \to aA$
3. $A \to b$

Program to Solve the Question of LR(0) Parser

(1) Find Augmented Grammer

(2) Find LR(0) collection of AItems by Method 1 or Method 2.

(3) Construct LR(0) parsing Table (No. of Items are state)
with ACTION & GOTO

(4) Parse the I/P string by the help of Parsing Table.