10/2 UNIT-2 ROLE OF PARSER

INTRO:

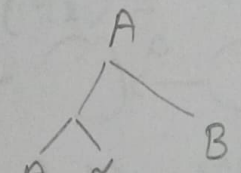
* PARSER → Syntax analysis of 2nd stage.
* grammar is ultimate output of parse tree
* Drawing parse tree leads to grammar

$A \longrightarrow aB \mid , \mid * \mid +$ // output of parse tree.

(Non-terminal) (terminal | non-terminal)

Elimination of left recursion:

$A \rightarrow A\alpha / B$



B remains same, left side increasing from top to bottom
↳ left recursion

[α → Combination of NT & T]

To Eliminate left recursion:

if left & right side having the same non-terminal Symbol → left recursion.

To overcome this.

$A \rightarrow \beta A'$

$A' \rightarrow \alpha A' \mid \varepsilon$

PROBLEM:

1. Eliminate left recursion from the following grammar.

$S \rightarrow (L) / a$

$L \rightarrow L, S / S$

$S \rightarrow (L) / a$ (in this first production NT in LHS and NT in RHS differs)

So, no left recursion needed.

$L \rightarrow \underset{A}{L}, \underset{\alpha}{S} / \underset{\beta}{S}$ (Here left recursion occurs)

$L \rightarrow SL'$

$L' \rightarrow ,SL' / \varepsilon$

2. Eliminate left recursion for the following.

$A \rightarrow Ac / Aad / bd / \varepsilon$

[2 left recursion]

$A \rightarrow Ac$ (A, α)

$A \rightarrow Aad$ (A, α)

$A \rightarrow A\alpha / A\alpha / B / B$

$A \rightarrow bdA' / \varepsilon A'$

$A' \rightarrow cA' / adA' / \varepsilon$

Elimination of left factoring.

$A \to \alpha\beta_1 \mid \alpha\beta_2 \mid \gamma_1 \mid \gamma_2$ [if no element is present replace by ε]

$A \to \alpha A' \mid \gamma_1 \mid \gamma_2$

$A' \to \beta_1 \mid \beta_2 \mid \varepsilon$

Problem:

1. eliminate the left factoring from the given grammar $S \to \underset{\alpha}{iEts} \mid \underset{\alpha\ \beta_1}{iEtSes} \mid \underset{\gamma_1}{a}$

$A \to \alpha\beta_1 \mid \alpha\beta_2 \mid \gamma_1 \mid \gamma_2$

$\alpha \to$ compare 1st & 2nd production

no β in 1st production

$S \to iEtSS' \mid a$ | $A \to \alpha A' \mid \gamma_1 \mid \gamma_2$

$S' \to$ ~~es~~ $\mid \varepsilon$ | $A' \to \beta_1 \mid \beta_2$

$\varepsilon \mid es$ | $\because$ no β_1 ; we replace as ε

2. $A \to \underset{\alpha\ \beta_1}{aAB} \mid \underset{\alpha\ \beta_2}{aA} \mid \underset{\alpha}{a}$

$A \to aA'$

$A' \to AB \mid A \mid \varepsilon$

2. $B \to \underset{\alpha\ \beta_1}{bB} \mid \underset{\alpha}{b}$

$B \to bB'$

$B' \to B \mid \varepsilon$

22/2 FIRST:

→ This is the set of all terminals that may begin in right-hand size of any right Sentential form.

→ first appearing terminal in the production.

First(x) : Rules to find out first(x)

→ if x is a terminal then first(x) = {x}

→ if x is a non-terminal and x → ε, then first(x) = ε

→ if x is a non-terminal, Defined by x → a or x → a

then first(x) = {a}

→ if x is a non-terminal, such that x → AB then first(x) = first(A) if A ↛* ε

→ Sum rule: if A → ε, first(A) ∪ first(B)

→ if x is a non-terminal & A → Y then first(x) = first(y)

PROBLEM:

Find first of the following production:

S → ε/a/b

S → ε
S → a
S → b

→ first(S) = {ε, a, b}

first we are going to find first(S)

→ $S \to aS|bs|\varepsilon$

$S \to \underline{a}s|\underline{b}s|\underline{\varepsilon} \Rightarrow first(s) = \{a, b, \varepsilon\}$

24/2 find the 1st function for following production:

(i) $S \to aS$
$S \to bS$
$S \to \varepsilon$

first(s) → terminals present

Sol $first(S) = \{a, b, \varepsilon\}$

(ii) $S \to aSb$
$S \to bSa$
$S \to \varepsilon$

[Any terminal symbol appearing at 1st]

$first(S) = \{a, b, \varepsilon\}$

(iii) $S \to aA|bB$
$A \to aA|\varepsilon$
$B \to bB|d$

$first(S) = \{a, b\}$
$first(A) = \{a, \varepsilon\}$; $first(B) = \{b, d\}$

(iv) $A \to A(A)|\varepsilon$

first is N.T, need to eliminate A when you substitute ε in A, then it will be as

A
↓
ε(A) → A will be vanished.

$first(A) = \{\epsilon, (\}$ → open parenthesis "("

(v) $S \to ABC$
$A \to a|\varepsilon$
$B \to b|\varepsilon$
$C \to c|\varepsilon$

$first(S) = \{first(A)\}$
$first(A) = \{a, \varepsilon, first(B)\}$
$first(B) = \{a, \epsilon, b, \varepsilon, first(c)\}$
$= \{a, \varepsilon, b, \varepsilon, c, \varepsilon\}$
$first(S) = \{a, b, \varepsilon, c\}$

first(A) = {a, ε} first(C) = {c, ε}

first(B) = {b, ε}

27/2 To find the first of the following productions

(i) $E \rightarrow TE'$

$E' \rightarrow +TE' | \varepsilon$ → terminal

$T \rightarrow FT'$

$T' \rightarrow *FT' | \varepsilon$

$F \rightarrow (E) | id$

First(E) = { First(T) }

= { First(F) }

First(E) = { (, id } → open Bracket.

First(E') = { +, ε }

First(T) = { First(F) }

= { (, id }

First(T') = { *, ε }

First(F) = { (, id }

(ii) $S \to (L) \mid a$

$L \to SL'$

$L' \to ,SL' \mid \varepsilon$

$First(S) = \{(, a\}$

$First(L) = \{First(S)\} = \{(, a\}$

$First(L') = \{ ,, \varepsilon\}$ → TERMINAL

(iii) $S \to AA$

$A \to aA \mid b$

$First(S) = \{First(A)\}$

$= \{a, b\}$

$First(A) = \{a, b\}$

(iv) $S \to ABC$

$A \to a \mid \varepsilon$

$B \to b \mid \varepsilon$

$C \to c \mid \varepsilon$

$First(S) = \{First(A)\}$

$= \{a, First(B)\}$

$= \{a, b, First(c)\}$

$= \{a, b, c\}$

[if we put ε, 0 will be negligible $S \to BC$]

$First(A) = \{a, \varepsilon\}$ $First(c) = \{c, \epsilon\}$

$First(B) = \{b, \varepsilon\}$

(v) $S \to aAb \mid Ba$ $B \to dB \mid \varepsilon$

$A \to bA \mid aB \mid \varepsilon$

$first(S) = \{a, first(B)\} = \{a, d, a\}$

$first(A) = \{b, a, \varepsilon\}$

$first(B) = \{d, \varepsilon\}$

FOLLOW(A):

is the set of All terminals that may follow to the right of A, in any right sentential form. [right side next to non-terminal]

Rules to find out Follow (A):

* if A is the start symbol of the grammar. then follow of A => follow(A) = {$}
* if $S \to \alpha A$ or $\alpha A \beta$, where $\beta \xrightarrow{*} \varepsilon$ then follow(A) = follow(S)
* if $S \to \alpha A \beta$, where $\beta \xrightarrow{*} \varepsilon$ (not leads too) then Follow(A) = First(B)
* if $S \to A$, then follow(A) = follow(S)

1|3|23 Find follow function for the following productio [values next to S]

(i) $S \to a|b|\varepsilon$ follow(S) = {$}

(ii) $S \to Sa|Sb|\varepsilon$ follow(S) = {a, b, $}

(iii) $S \to aSb|\varepsilon$ follow(S) = {b, $}

(iv) $S \to aSb|bSa|\varepsilon$ follow(S) = {b, a, $}

(v) $S \to aA|b$ follow(S) = {$}

$A \to aAb|\varepsilon$ follow(A) = {b, follow(S)}

= {b, $}

vi) $S \rightarrow aAb \mid Ba$ Since there is no $ in values it will be '$'
$A \rightarrow Aa \mid bB \mid b$
$B \rightarrow Be \mid d$

follow (S) = {$}
follow (A) = {b, a} $\Rightarrow$ {follow (S), a}
follow (B) = {a, e, follow (A)} = {a, e, b, $}

6/3 LL(1) Grammar or Top down Parser [Left to Left grammar]

Step 1: Push Start Symbol into the Stack
Step 2: Compare the top most Symbol of the Stack with look ahead Symbol
Step 3: If Matching occurs then pop off the the grammar & Symbol ans. Increment the Input pointer.

Example:
$S \rightarrow aAb \mid bB$
$A \rightarrow aA \mid b$
$B \rightarrow bB \mid a$
$w = aaabb$

PARSE TABLE

	a	b	$
S	S → aAb	S → bB	
A	A → aA	A → b	
B	B → a	B → bB	

STACK	INPUT STRING	ACTION
$S	aaabb$	push S → aAb
$bAa	aaabb$	pop
$bA	aabb$	push A → aA
$bAa	aabb$	pop
$bA	abb$	push A → aA
$bAa	abb$	pop
$bA	bb$	push A → b
$bb	bb$	pop.
$b	b$	pop.
$	$	parsing is Successful

LL(1) ALGORITHM:

Let x is the top most Symbol of the stack & 'a' is the look ahead Symbol.

1. if x = a = $ then parsing is Successful
2. If x = a ≠ $ then pop off ans increment the I/P pointer
3. If x ≠ a ≠ $ and M[x, a] contain the production x → u v w then replace 'x' by u v w in the reverse order.

7/3 Construction of LL(1) Parse table (or) Predictive Parse table

Procedure to construct LL(1) Predictive Parsing table:

→ for every production of the form $A \to \alpha$ repeat the following rules

(i) add $A \to \alpha$ in $M[A, a]$ (row, col), for every terminal 'a' in first[α]

(ii) if first[α] contains ε, then Add $A \to \alpha$ in $M[A, b]$ for every terminal 'b' in follow(A)

Q: Construct LL(1) predictive parse table for the following production:

$S \to \underline{a}A \mid \underline{b}B$ — first(S) = {a, b}

$A \to \underline{a}A \mid \underline{b}$ — first(a) = {a, b}

$B \to \underline{b}A \mid \underline{c} \mid \underline{\varepsilon}$ — first(B) = {b, c, ε}

	a	b	c	$
S	S → aA	S → bB	ERROR	ERROR
A	A → aA	A → b	ERROR	ERROR
B	ERROR	B → bA	B → c	B → ε

A → ε comes from B, need to find the follow(B). After B no production, then use from where the B is derived.

follow(B) = follow(S) as S is a start Symbol then follow(S) = $

[Row filled by N.T, Column by T & $]

[non-production box = ERROR]

Construct LL(1) Predictive-parse table.

$S \rightarrow aA \mid BaA$

$A \rightarrow aA \mid \varepsilon$

$B \rightarrow bB \mid \varepsilon$

To find first function of all non-terminals

[Since B is a N.T. find first(B)]

First(S) = {a, First(B) = b, ε}

[ε will be negligible, so a]

= {a, b, a}

First(A) = {a, ε}

First(B) = {b, ε}

Now, construct parse tree.

	a	b	$
S	$S \rightarrow aA$ $S \rightarrow BaA$	$S \rightarrow BaA$	
A	$A \rightarrow aA$		$A \rightarrow \varepsilon$
B	$B \rightarrow \varepsilon$	$B \rightarrow bB$	

$\varepsilon \rightarrow$ follow(whose it belongs)

We must see what is there after A

Since no symbol we must find for follow(S)

Since S is the start symbol it is $

follow(A) = follow(S) = $

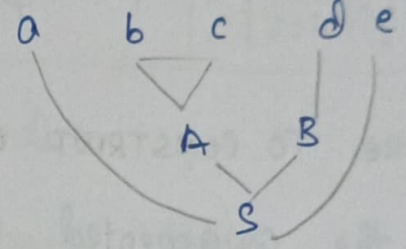
follow B = {a}

10/s Bottom up Parser (or) Shift Reduce Parser.

The process of constructing the parse tree starting with the I/p string & getting the start symbol S of the grammar is called as Bottom up parser.

Ex:
S → aABe
A → b/bc
B → d

Ex: Input String



HANDLE:

The Sequence of grammar Symbol that Matches the R.H.S of any production is called Handle.

Ex: S → aABe
A → b/bec
B → d.

grammar Symbol matches with the I/p string

4. Construct Shift Reduce Parsing table from the grammar

Ex: S → AA
A → aA/b

Input String: abab$

(i) Shift 1st Symbol from I/P.
(ii) if it Doesn't matches with I/p string then shift again
if it matches with grammar, perform Reduce

STACK	INPUT STRING	ACTION
$	abab$	Shift
$a	bab$	Shift
$ab	ab$	Reduce (A → b)
$aA (A→aA/b)	ab$	Reduce (A → aA)
$A (A→aA) [if not matches then shift]	ab$	Shift
$Aa	b$	Shift
$Aab	$	Reduce (A → b)

$AaA (A→aA/b)	$	Reduce (A→aA)
$AA (A→aA)	$	Reduce (S→AA)
$S [START SYMBOL]	$	Accept

8/3 PROCEDURE TO CONSTRUCT LR PARSING TABLE:

→ Obtain the augmented grammar.
→ Construct the Cannonical Collection of LR(0) items
→ Draw the DFA
→ Construct the parse table.

AUGMENTED GRAMMAR:

The grammar which is obtained from adding one or more production is called Augmented grammar.

Ex: $S \to AB$, $A \to a$, $B \to b$ ⟹ $S' \to S$, $S \to AB$, $A \to a$, $B \to b$

LR(0) item or LR(K) item:

Any production which has · anywhere on R.H.S of the production is called LR(0) item.

Ex:

$A \to xyz$
$A \to .xyz$
$A \to x.yz$
$A \to xy.z$ } Shift operation
$A \to xyz.$ ⟶ Reduce.

PROBLEM 1: Construct LR parse table.

Ex → $A \to aA \to r_1$
$A \to b \to r_2$

Step 1: Augmented grammar:

$A' \to A$ → One more production in Start Symbol
$A \to aA$
$A \to b$

Step 2: I_0: closure $(A' \to \cdot A)$ — R.H.S ADD . in the starting symbol. if it is a N.T need to expand all the productions of non-Terminal

I_0:
$A' \to .A$
$A \to .aA$
$A \to .b$

$\Rightarrow$ goto $(I_0, A) = I_0$. (move the .)

I_1: $A' \to A.$ (shifting the .)

$\Rightarrow$ goto $(I_0, a) = I_2$

$\Rightarrow I_2$:
$A \to a.A$
$A \to .aA$
$A \to .b$

After . if it is a N.T then expand all terminals.

goto $(I_0, b) = I_3$ [Reaches the final position then stop]

I_3 : $A \rightarrow b.$

goto $(I_2, A) = I_4$

I_4 : $A \rightarrow aA.$ NEW STATE

goto $(I_2, a) = I_2$ → NT Expand.

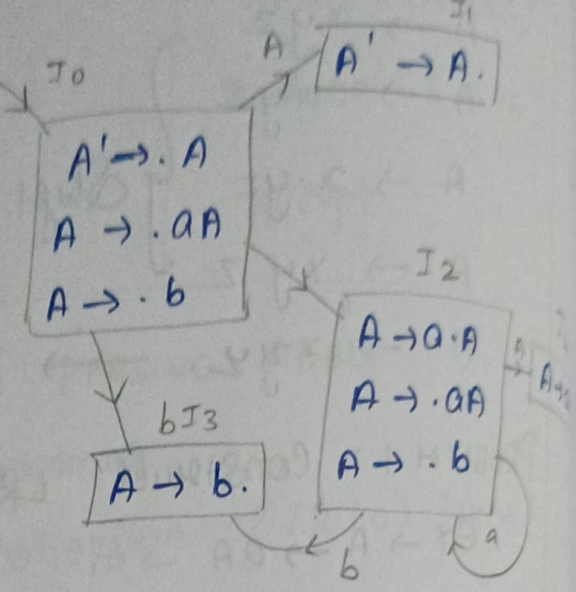
$A \rightarrow a.A$
$A \rightarrow .aA$
$A \rightarrow .b$

SAME

goto (I_2, b)

$A \rightarrow b.$: I_3 SAME

DRAW THE DFA:



BASED ON DFA CONSTRUCT LR PARSE TABLE

	a	b	$	A
I_0	S_2	S_3		1
I_1	REACHES THE FINAL POSITION ACCEPT	ACCEPT	ACCEPT	
I_2	S_2	S_3		4
I_3	γ_2	γ_2	γ_2	
I_4	γ_1	γ_1	γ_1	

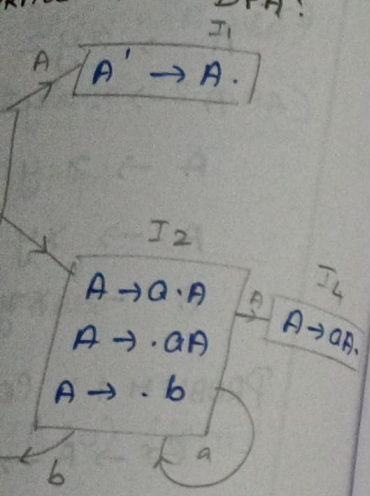
All symbols.

Non-Terminal just mentions the State

TERMINAL $I_2 \rightarrow S_2$ (Shift op) ; $I_3 \rightarrow S_3$

if it is not a augmented then mention (Reduced Pos.NO) => g_1, g_2, g_3

RAW THE DFA:



STRUCT LR PARSE TABLE

Non-Terminal just mention the State

=> g_{12}, g_{13}, g_{12}

Construct SLR Parse Table

SLR PARSE:

Parse table construction procedure of SLR(1) is similar to LR(0) one difference.

Whenever there is a final item then reduce the entry under the following symbol of L.H.S variable.

Then the table is SLR (Table)

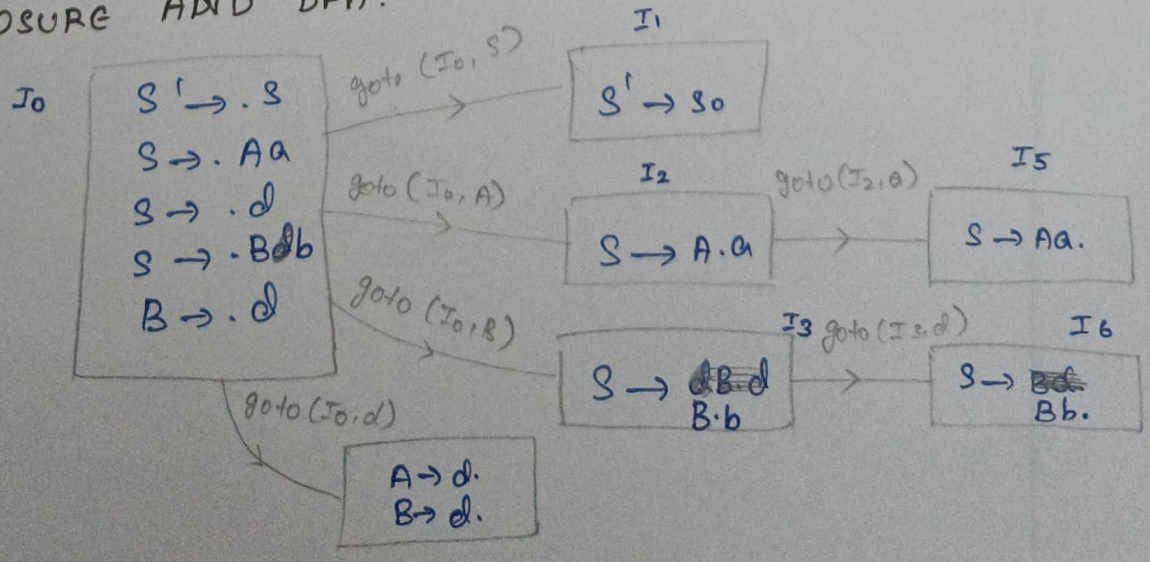
S → Aa
S → Bb
A → d
B → d

Augmented GRAMMAR:

S' → S
S → Aa
S → Bb
A → d
B → d

CLOSURE & GOTO & DFA

CLOSURE AND DFA:



Find item States:

Start Symbol

$S' \to S$.

$I_1 : S' \to S.$ [Accept]

$I_4 : A \to d.$ [r3]
$B \to d.$ [r4]

$I_5 : S \to Aa.$ [r1]

$I_6 : S \to Bb.$ [r2]

Reduce operation for SLR:

1. (I_4) $A \to d. \to r3$
   $B \to d. \to r4$
   follow (A) = {a}
   follow (B) = {b}
2. (I_5) $S \to Aa. \to r1$
   follow (S) = $
3. (I_6) $S \to Bb. \to r2$
   follow (S) = $

	a	b	d	$	S	A	B
→ I_0			S_4	ε	1	2	3
I_1	Accept	Accept	Accept	Accept	Accept	Accept	Accept
I_2	S_5						
I_3		S_6					
I_4	R_3	R_4					
I_5				R_1			
I_6				R_2			

CONSTRUCT CANONICAL LR PARSING TABLE [CLR]

$S \to CC$
$C \to aC$
$C \to b$

Augmented grammar

$S' \to S$
$S \to CC$
$C \to aC$
$C \to b$

I_0:
$S' \to .S, \$$
$S \to .CC, \$$
$C \to .aC, a/b$
$C \to .b, a/b$

→ Starting Symbol, lookahead $, First($) = $
First $
First C $
$negligible, first(C) = a/b

(I_0, S): I_1
$S' \to S. \$$

(I_0, C): I_2
$S \to C.C, \$$
$C \to .aC, \$$
$C \to .b, \$$
first($) = $
First $

(I_0, a): I_3
$C \to a.C, a/b$
$C \to .aC, a/b$
$C \to .b, a/b$
→ Since N.T, Expand the productions of C
First (a, b)
first(a, b) = a/b

(I_0, b): I_4
$C \to b., a/b$

(I_6, b): I_7
$C \to b., \$$

(I_2, C): I_5
$S \to CC. \$$

(I_2, a): I_6
$C \to a.C, \$$
$C \to .aC, \$$
$C \to .b, \$$
First(

(I_2, b): I_7
$C \to b. \$$

(I_3, C): I_8
$C \to aC.a/b$

(I_3, a): I_3
$C \to a.C, a/b$
$C \to .aC, a/b$
$C \to .b, a/b$
First (a/b

(I_3, b): I_4
$C \to b., a/b$

(I_6, C): I_9
$C \to aC. \$$

(I_6, a): I_6
$C \to a.C, \$$
$C \to .aC, \$$
$C \to .b, \$$
First($)

	a	b	$	S	C
I_0	S_3	S_4		1	2
I_1	Accept	Accept	Accept	Accept	Accept
I_2	S_6	S_7			5
I_3	S_3	S_4			8
I_4	γ_3	γ_3			
I_5			γ_1		
I_6	S_6	S_7			
I_7			γ_3		
I_8	γ_2	γ_2	γ_2		
I_9			γ_2		