Top-Down Parsing

6.11.2023

* Construction of Non-Recursive Predictive Parser

Accepts grammer of the form of LL(1)

i/p :- LL(1) grammar

1st L : stands for Left to Right scanning of input

2nd L : stands for Leftmost Derivation

1 : one lookahead symbol

* A G is LL(1) if and only if whenever $A \to \alpha | \beta$ are two different productions of G, the following conditions hold:

1. For no terminal 'a', do both α and β derive strings beginning with a
2. Almost one of α and β can derive the empty string.
3. If $\beta \overset{*}{\Rightarrow} \varepsilon_0$ (0 or more derivations) then α doesnt derive string beginning with a terminal in FOLLOW(A) ie v, a !

* FIRST(X)

1. If X is a terminal then First(X) = {X}
2. If X is a non-terminal and $X \to Y_1 Y_2 \ldots Y_n$ then place 'a' in FIRST(X) for a in FIRST(Y_1) and ε_0 in all of FIRST(Y_1) ... FIRST(Y_n) i.e $Y_1 \ldots Y_n \overset{*}{\Rightarrow} \varepsilon_0$.
   If FIRST(Y_n) doesn't derive ε_0, then don't add ε_0 in FIRST(X).
3. If $X \to \varepsilon_0$ is a production, then add ε_0 to FIRST(X)

* FOLLOW(A) [A → non-terminal]

1. Place $ in FOLLOW(S) if S is the start symbol and $ is the end of input string.
2. If $A \to \alpha B \beta$, then everything in FIRST(β) except ε_0 is in FOLLOW(B).
3. If $A \to \alpha B$ or $A \to \alpha B \beta$, where FIRST(β) contains ε_0, then everything in FOLLOW(A) is in FOLLOW(B)

Eg: $S \rightarrow 0S1 \mid 01$

Left Factoring,

$S \rightarrow 0S'$

$S' \rightarrow S1 \mid 1$

Parsing,

	First	Follow
S	0	{$, 1}
S'	0, 1	{1, $}

First(S') = First(S)

$S \rightarrow 0S'$ $\quad \alpha B$ (R.S)

$S' \rightarrow S1$ $\quad \alpha B\beta$ $\quad Fi(B) = 1$ Lyf(S)

$S' \rightarrow 1$

---

* Problems:

$\varepsilon \rightarrow T-$ $\quad E \rightarrow T+TE'$

1) $E \rightarrow TE'$

$E' \rightarrow +TE' \mid \varepsilon$

$T \rightarrow FT'$

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

$F \rightarrow (E) \mid id$

ELR $\Leftarrow$

$E \rightarrow E+T \mid T$

$T \rightarrow T*F \mid F$

$F \rightarrow (E) \mid id$

*

Non-Terminals	First	Follow
E	(, id	$,)
E'	+, ε	$,)
T	(, id	$,), +
T'	*, ε	+,), $
F	(, id	*, +,), $

$Fi(E) = Fi(T) = Fi(F)$

* Applying Rules,

① $F(E) \Rightarrow \$$

② $F_i(E') - \varepsilon_0 \Rightarrow F(T)$ → $E \to TE'$ (α B B, α B)

③ $F(E) \Rightarrow F(T)$

④ $F(E) \Rightarrow F(E')$

⑤ $F(E') \Rightarrow F(T)$

⑥ $F_i(T') - \varepsilon_0 \Rightarrow F(F)$

⑦ $F(T) \Rightarrow F(F)$

⑧ $F(T) \Rightarrow F(T')$

⑨ $F(T') \Rightarrow F(F)$

⑩ $F(E) \Rightarrow)$ $F \to (E)$ (α B B)

---

2)

$S \to (S)X \mid aX$

$X \to +SX \mid SX \mid *X \mid \varepsilon_0$

Non-terminals	First	Follow
S	(, a	$,), +, (, a, *, $
X	+, (, a, *, ε_0	$,), +, (, a, *, $

R3: $S \to (S)X$ (α = (S), B = X)

R2: α B, B

Rules:

① $F(S) \Rightarrow \$$

② $F_i(X) - \varepsilon_0 \Rightarrow F(S)$

③ $F(S) \Rightarrow F(X)$

$F(X) \Rightarrow F(S)$

$S \to aX$ (α B)

R3: α B

$S \to +SX$ (α B B)

R2;

⇒ Everything in $F_i(X)$ except ε_0 in $F(S)$

Parse Table
M[A, a] → terminals
↓ Non-terminals

* Construction of Predictive Parsing Table

Algorithm

i/p - G
o/p - Parsing Table M

Method:-

① For each production $A \to \alpha$ of the G, do the foll:-

1. For each terminal a in FIRST(α), add $A \to \alpha$ to M[A, a]
2. If ε is in FIRST(α), then for each terminal b in FOLLOW(A), add $A \to \alpha$ to M[A, b]. If ε is in FIRST(α) & \$ in FOLLOW(A), add $A \to \alpha$ to M[A, \$] as well.
3. If there is no production in M[A, a], then set M[A, a] to error.

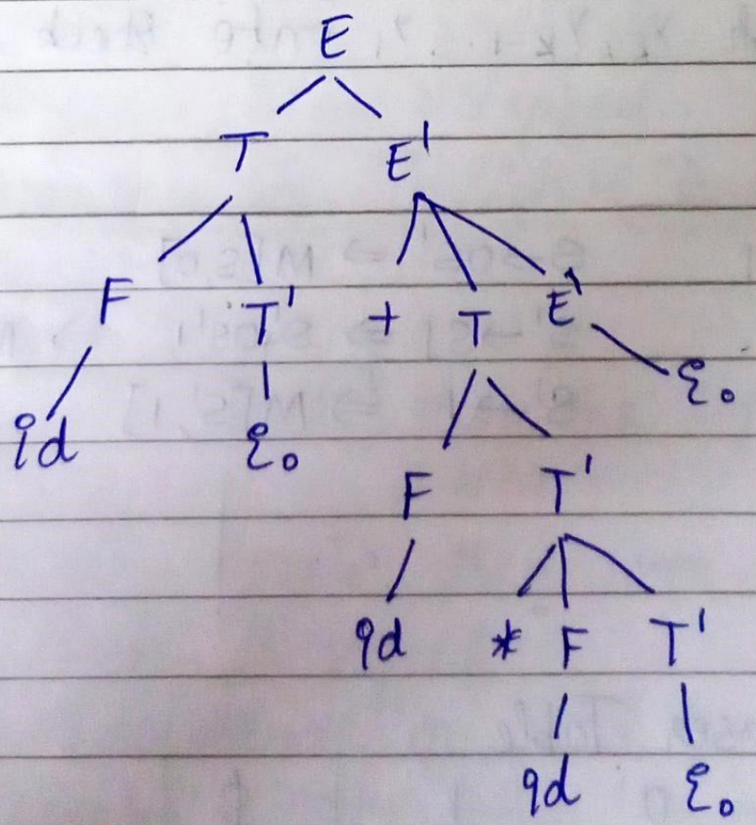
G:- Example from Previous Pg - From First-Follow table

Table driven Predictive Parser

M[A,b] Terminals / Non-terminals	+	*	(	)	id	\$ (right end marker)
E			$E \to TE'$		$E \to TE'$	
E'	$E' \to +TE'$			$E' \to \varepsilon$		$E' \to \varepsilon$
T			$T \to FT'$		$T \to FT'$	
T'	$T' \to \varepsilon$	$T' \to *FT'$		$T' \to \varepsilon$		$T' \to \varepsilon$
F			$F \to (E)$		$F \to id$	

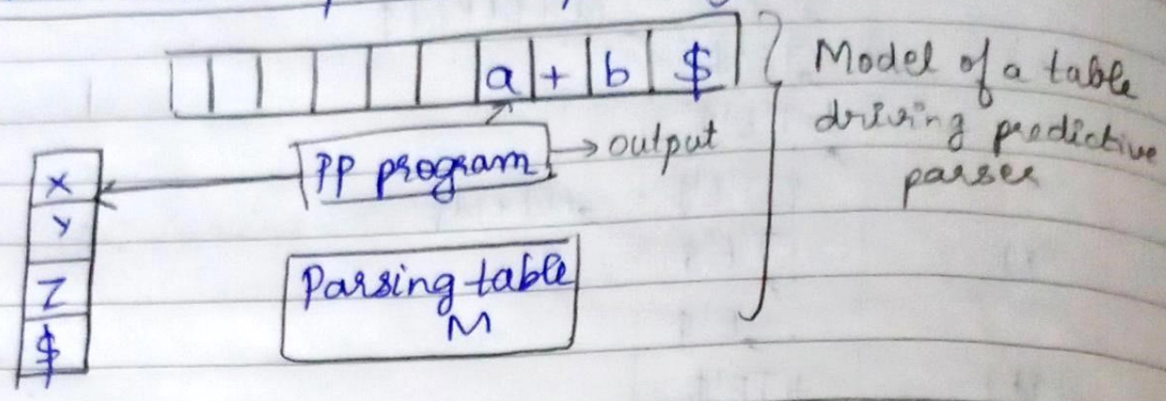
String	Stack	Input Symbol
	$E\$$	$id + id * id \$$
	$TE'\$$	
	$FT'E'\$$	
	$idT'E'\$$	$+id * id\$$
id	$T'E'\$$	
	$E'\$$	
id +	$+TE'\$$	
	$TE'\$$	$id * id\$$
	$FT'E'\$$	
id + id	$idT'E'\$$	
	$T'E'\$$	$*id\$$
	$*FT'E'\$$	
id + id *	$FT'E'\$$	$id\$$
	$idT'E'\$$	$id\$$
id + id * id	$T'E'\$$	$\$$
	$E'\$$	$\$$
	$\$$	$\$$

accepted



16.11.2023

* Non-recursive predictive parsing:



Predictive Parsing Algorithm:

Set i/p to point → 1st symbol of the string
Set x to the top stack symbol
while (x ≠ $) {
    // Stack not empty
    if (x is a) Pop the stack & Advance i/p;
    else if (x is terminal) error();
    else if (M[x,a] is an error) error();
    else if (M[x,a] = x → $Y_1, Y_2 \ldots Y_K$) {
        o/p → the production $x \to Y_1, Y_2 \ldots Y_K$,
        Pop the stack;
        Push $Y_K, Y_{K-1} \ldots Y_1$ onto stack;
    }

Eg: $S \to 0S1 \mid 01$

$S \to 0S'$
$S' \to S1 \mid 1$

$S \to 0S' \Rightarrow M[S,0]$
$S' \to S1 \Rightarrow S1 0S'1 \Rightarrow M[S',0]$
$S' \to 1 \Rightarrow M[S',1]$

* Predictive Parser Table

NT \ T	0	1	$
S	S→0S'		
S'	S'→S1	S'→1	

a+b a+*b
a*b

* Parsing process

i/p: 0011$

Match	Stack	i/p string	
	S$	0011$	M[S,0]
	0S'$	0011$	
0	S'$	011$	Pop & advance
	S1$	011$	M[S',0]
	0S'1$	11$	M[S,0]
00	S'1$	11$	Pop & advance
	11$	11$	M[S',1]
001			Pop & advance
	1$	1$	Pop & advance
0011	$	$	accepted

* But 00011 is not accepted.

* Error Recovery in Predictive Parsing:

Panic Mode Recovery

1) If M[A,a] = blank, then the i/p symbol 'a' is skipped.
2) If M[A,a] = synch, then non-terminal in top of stack is popped.
3) If token / terminal on stack top doesn't match the i/p symbol, pop token / terminal from stack.

* Synch - Synchronising set of token

Note:- If no entry in M[A,a] where a ∈ Follow(A), then add synch to M[A,a].

* If 1st contains ε, compute follow

Eg:
E → TE'
E' → +TE' | ε
T → FT'
T' → *FT' | ε
F → (E) | id

	Follow
E	$,)
E'	$,)
T	+, $,)
T'	+, $,)
F	*, +, $,)

NT \ T	id	+	*	(	)	$
E	E→TE'			E→TE'	synch	synch
E'		E'→+TE'			E'→ε	E'→ε
T	T→FT'	synch		T→FT'	synch	synch
T'		T'→ε	T'→*FT'		T'→ε	T'→ε
F	F→id	synch	synch	F→(E)	synch	synch

i/p :-)id * + id $

Stack	i/p string	
E$	)id*+id$	[skip ')']
TE'$		
FT'E'$		
idT'E'$		Pop & advance
T'E'$	*+id$	
*FT'E'$		Pop & advance
FT'E'$	+id$	
T'E'$	+id$	Skip (synch)
E'$	+id$	
+TE'$	+id$	Pop & advance
TE'$	id$	
FT'E'$	id$	
idT'E'$	id$	Pop & advance
T'E'$	$	Match T'→ε
E'$	$	Match E'→ε
$	$	) → Error

* Bottom-Up Parsing

* Bottom-Up parser is also called shift-reduce parser.
* Basic operations in Shift reduce parsing:
  1. Shift
  2. Reduce
  3. Accept
  4. Error

* LR Parser → LALR(1) [Current]

$S \rightarrow 0S1 \mid 01$

Identify the handles for the string 000111 while doing shift Reduce Parsing

Handle	Stacke	Input
	$	000111$
	$0	00111$
	$00	0111$
	$000	111$
01	$0001	11$
	$00S	-"-
0S1	$00S1	1$
	$0S	-"-
0S1	$0S1	$
	$S	$
		accepted

* Simple LR parser

Augmented G:

$S' \rightarrow S$

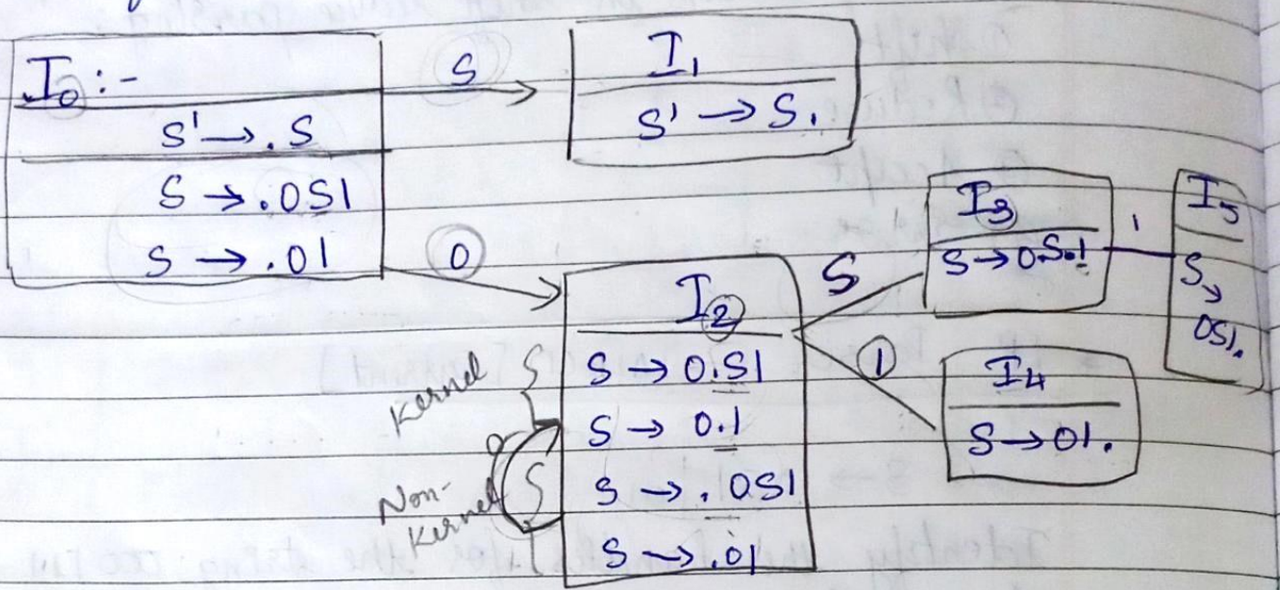
$S \rightarrow 0S1$

$S \rightarrow 01$

* SLR → LR(0)

$S \to \overset{①}{0}S1 \mid \overset{②}{01}$

* Set of items



Goto :- $I_i, S = I_j \Rightarrow I_0, S = I_1 \Rightarrow 0, S = 1$

(S) $I_i, a = I_j \Rightarrow$ Shift a and move to j $\Rightarrow S_j$ (Shift actions)

States	Action			Goto
	0	1	$	(S)
0	S_2			1 $[I_0, S = I_1]$
1			accept	
2	S_2	S_4		3 $[I_2 S = I_3]$
3		S_5		
4		r_2	r_2	
5		r_1	r_1	

(S) → 0S1.

↓

follow (S) → 1, $

① ② ③

* Eg: $S \to 0S0 \mid 1S1 \mid 10$

Augmented G:-

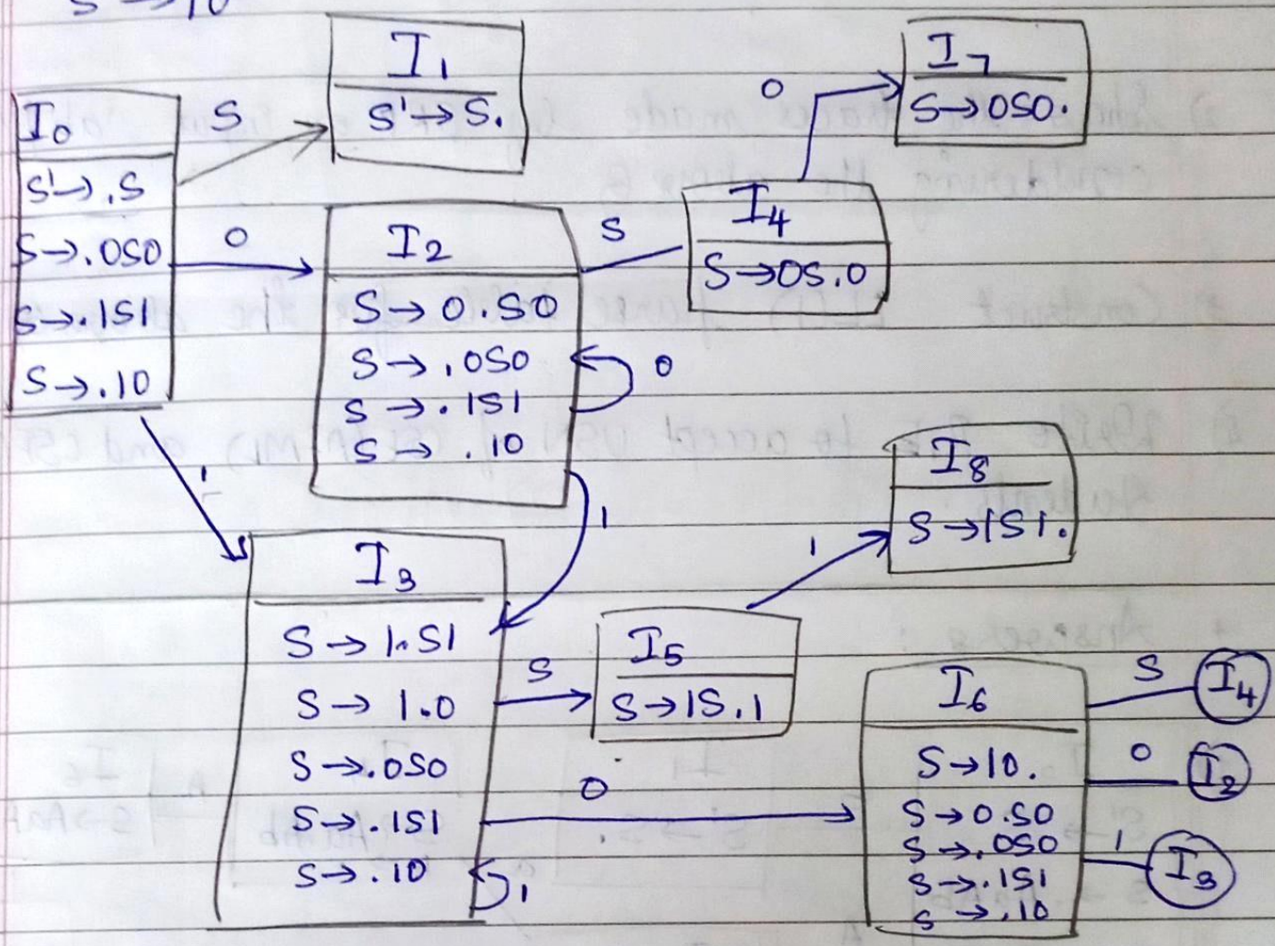
$S' \to S$

$S \to 0S0$

$S \to 1S1$

$S \to 10$

Follow(S) = $, 0, 1



States	Action			Goto
	0	1	$	(S)
0	S_2	S_3		1
1			accept	
2	S_2	S_3		4
3	S_6	S_3		5
4	S_7			
5		S_8		
6	S_2/R_3	S_3/R_3	R_3	4
7	R_1	R_1	R_1	
8	R_2	R_2	R_2	

23.11.2023

* Problems:

1) Construct LR(0) parse table for
   S → AaAb | BbBa
   A → ε ①
   B → ε ④

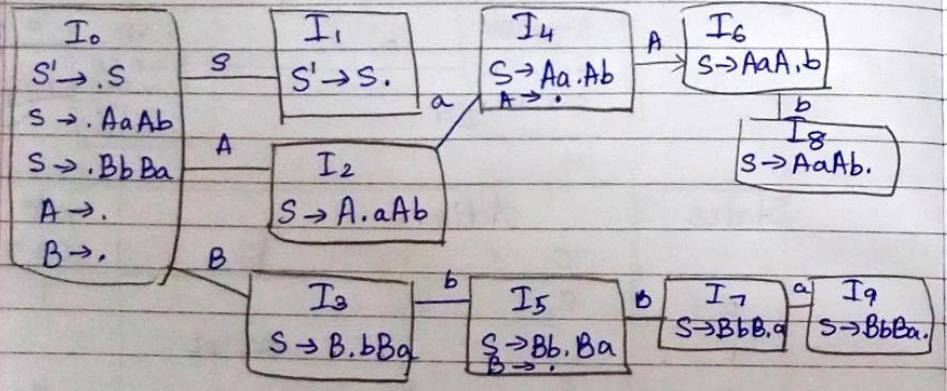
2) Show the traces made by SRP on input 'ab$' considering the above G.

3) Construct LL(1) parse table for the above G.

4) Write R.E to accept USN of CSE(AIML) and CSE(CY) students.

* Answers:

1)



Σ = {a, b}

Follow(A) = {a, b, $} ③ A→. in I0

For 0, a → r3

0, b → r3

F(B) = {a, b, $} ④ B→. in I0

0, a & 0, b → r4

P.T.O →

States	Action			Goto		
	a	b	$	S	A	B
0	r3/r4	r3/r4		1	2	3
1			accept			
2	S4					
3		S5				
4	r3	r3			6	
5	r4	r4				7
6		S8				
7	S9					
8			r1			
9			r2			

2) S → AaAb ① | BbBa ② } No LR & LF
   A → ε ③
   B → ε ④

Stack	Input	Action
$	ab$	Reduce A → ε
$Aa	b$	Shift
$AaA	b$	Reduce A → ε
$AaAb	$	Shift
$S	$	Reduce S → AaAb
$S	$	accepted

3) LL(1) parse table S →

NT \ T	a	b	$
S	S→AaAb	S→BbBa	
A	A→ε	A→ε	
B	B→ε	B→ε	

NT	First	Follow
S	a, b, ε	$
A	ε	a, b
B	ε	a, b

If first contains ε, compute follow(A)

* Check each & every G, S → AaAb → first ⇒ A → ε ⇒ a (first ⇒ a) → [S, a] → write this

Parsing

Stack	Input	
$S\$$	$ab\$$	
$AaAb\$$	$ab\$$	$M[A,a] \Rightarrow A \to \varepsilon$
$aAb\$$	$ab\$$	Pop & advance
$Ab\$$	$b\$$	$M[A,b] \Rightarrow A \to \varepsilon$
$b\$$	$b\$$	Pop & advance
$\$$	$\$$	accept

4)

$$([1MS][21/22/23][CI/CY][0\text{-}4][0\text{-}9][0\text{-}9])^*$$