Date ___/___/_____

Saathi

Bottom up Parsing LR

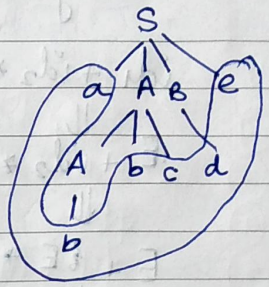
L → Left to right scan
R → rightmost derivation in Reverse order.

Given a grammar S→aABe A → Abc | b
B → d

w → abbcde
First check Right-hand side of the production

~~S → aABe~~
~~A~~
e ~~Abc~~ ~~b~~

w ⇒ abbcde
⇒ aAbcde
⇒ aAde
⇒ aABe
⇒ S



Rightmost derivation →
S ⇒rm aABe ⇒rm aAde ⇒rm aAbcde ⇒rm abbcde

* Handle:
A Handle of a string is a substring that matches the right side of a production and who's reduction to the nonterminal on the left side of the production that represents one step along the reverse of a rightmost derivation

S → aABe
A → Abc | b
B → d

abbcde
aAbcde A → b this b is a handle.
aAbcde A → Abc → this is the handle
aAde B → d → this substring is the handle.

Date ___/___/_____

S $S \rightarrow aABe$

this substring is the handle.

* Consider the following grammar:

$E \rightarrow E+E$
$E \rightarrow E \times E$
$E \rightarrow (E)$
$E \rightarrow id$

String is $id_1 + id_2 * id_3$

$id_1 + id_2 * id_3$
$\Downarrow$
$E + id_2 * id_3$ $\quad E \rightarrow id_1$ → id_1 is a handle
$\Downarrow$
$E + E * id_3$ $\quad E \rightarrow id_2$ → id_2 is a handle
$\Downarrow$
$E + E * E$ $\quad E \rightarrow id_3$ → id_3
$\Downarrow$
$E + E$ $\quad E \rightarrow E \times E$ → handle
$\Downarrow$
E $\quad E \rightarrow E+E$ → handle

$E \overset{rm}{\Rightarrow} E+E \overset{rm}{\Rightarrow} E+E*E$
$\overset{rm}{\Rightarrow}$
$id_1+id_2*id_3$

Right Sentential Form	Handle	Reducing Production
$id_1+id_2*id_3$	id_1	$E \rightarrow id_1$
$E+id_2*id_3$	id_2	$E \rightarrow id_2$
$E+E*id_3$	id_3	$E \rightarrow id_3$
$E+E*E$	$E*E$	$E \rightarrow E*E$
$E+E$	$E+E$	$E \rightarrow E+E$
E	$\$$	$\$$

Saathi

Date __/__/____

30/08/2022

① Expression → expression + term
Expression → expression * term
expression → term

A → β A'

expr → term expr'
expr' → +term expr' | -term expr' | ∈

α A α A'

S → ACB | CbB | Da
A → da | BC
B → g | ∈
C → h | ∈

fi(S) = {d, h, b, a, g, ∈}	fo(S) → {$}
fi(A) = {d, g, h, ∈}	fo(A) → {h, g, $, $}
fi(B) = {g, ∈}	fo(B) → {b, a, $, g, h}
fi(C) = {h, ∈}	fo(C) → {b, g, $, h}

FOLLOW(B) = FOLLOW(S) ∪ FIRST(a) ∪ FIRST(c) ∪ FOLLOW(A) = {$, a, g, h}

FOLLOW(c) = FIRST(b) ∪ FOLLOW(A) ∪ FIRST(B) ∪ FOLLOW(S)

---

* Stack Implementation of Shift-Reduce Parsing (Bottom-up)

	Stack	Input
Initial Config	$	w$
Final	$E	$

The primary operation of the parser are shift & reduce, there are actually 4 possible actions a shift reduce parser can make: shift, reduce, accept, error.

① In a shift action, the next symbol is shifted onto the top of the stack.
② In a reduce action, the parser knows the right end of the handle each at the top of the stack

Saathi

Date ___/___/_____

(3) It must then locate the left end of the handle within the stack and reside with what non-terminal to replace the handle.

(4) In an accept action the parser announces successful completion of parsing.

(5) In an error action, the parser discovers that a syntax error, and calls an error recovery routine.

	Stack	Input	Action
(1)	$	$id_1 + id_2 * id_3 $	Shift
(2)	$id_1	$+id_2 * id_3 $	Reduce by E→id
(3)	$ E	$+id_2 * id_3 $	Shift
(4)	$E+	$id_2 * id_3 $	Shift
(5)	$E + id_2	$* id_3 $	Reduce by E→id
(6)	$E + E	$* id_3 $	Reduce by E→E+E
(7)	$ E	$* id_3 $	Shift
(8)	$ E *	$id_3 $	Shift
(9)	$E * id_3	$	Reduce E→id
(10)	$E * E	$	Reduce E→E*E
(11)	$ E	$	Accept

Operator Precedence Parsing

Operator Grammar: "A grammar have the operator property that no production on the right side has two adjacent non-terminal.

Eg: $E \to E+E \mid E-E \mid E \times E \mid E/E \mid E \uparrow E \mid (E) \mid id$.

not an eg: $EAE \mid (E) \mid id$

$A \to + \mid - \mid * \mid / \mid \uparrow$

Date ___/___/_____ Saathi

In precedence parsing we define three disjoint precedence relation $<\cdot, \doteq, \cdot>$ between pair of terminals. This precedence relation ship guide the selection of handle and have the following meaning. Eg. $a <\cdot b$

✗ $F \to \epsilon$
✗ $F \to AA$

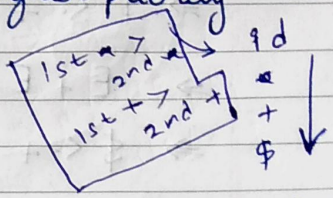
Relation	Meaning
$a <\cdot b$	a" yields precedence to" b
$a \doteq b$	a" has same precedence over" b
$a \cdot> b$	a" takes precedence over" b

Operator precedence Relation matrix:

$E \to E+E \mid E*E \mid id$
$W = id_1 + id_2 * id_3$

	id	+	*	$
id	·	·>	·>	·>
+	<·	·>	<·	·>
*	<·	·>	·>	·>
$	<·	<·	<·	·

Highest priority → id



When we are given a string, insert $ on both ends of the string.

$W = \$id + id * id \$$

$W = \$ <\cdot id \cdot> + <\cdot id \cdot> * <\cdot id \cdot> \$$

Then find the handle using the steps:

Step 1: Scan the string from the left end until the first $\cdot>$ is encountered.

Step 2: Then scan backward (to the left) over any $\doteq$ until $a <\cdot$ is encountered.

Step 3: The handle contains everything to the left to the first $\cdot>$ and to the right of the $<\cdot$ encountered in Step 2, including any intervening surrounding non-terminal.

Saathi

Date

L to R

$ <· id ·> + <· id ·> * <· id ·> $

Step1

R to L

handle is the id

→ Reduce the first id using non-terminal

E + id * id reduce E → id

→ $ <· + <· id ·> * <· id ·> $

handle = * id

⇒ E + E * id Reduce E → id

⇒ $ <· + <· * <· id ·> $

handle

⇒ E + E * E Reduce E → id

⇒ $ <· + <· * ·> $

⇒ E + E Reduce by E → E * E

⇒ $ <· + ·> $

⇒ E Reduce by E → E + E

⇒ $ $

precedence of $f(id) > g(id)$

5/09/22

	id	+	*	(	)	$
f(a)	5	3				
g(b)	4	2				

a ·> b f(a) > g(b)

a ·= b f(a) = g(b)

a <· b f(a) < g(b)

Construct precedence function

Input → operator precedence matrix

Output → precedence function representing the input matrix.

* **Step 1:** Create Symbol f(a) and g(a) for each 'a' ie a terminal or end marker

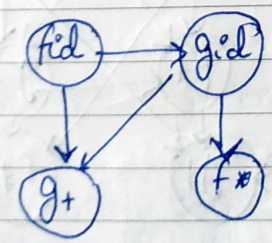
* **Step 2:** Partition the created symbol as many groups as possible in such a way that if a = b then f(a) and g(b) are in the same group.

* **Step 3:** Create a directed graph, whose nodes are the groups found in Step two, for any a and b, if a <· b place an edge from the group of g(b) to the group of f(a). If a ·> b place an edge from the group of ~~f(a)~~ fa to that of ~~g(b)~~ gb

if (a ·> b) ~~g(a)~~ ~~f(b)~~ $g_a \rightarrow f_b$

if (a <· b) g(a) ~~f(b)~~ $g_a \leftarrow f_b$

* **Step 4:** If the graph constructed in Step 3 has a cycle then no precedence function exist. If there are no cycle, let f(a) be the length of the longest path beginning at the group of fa and let g(a) be the length of the longest path from the group ga.



f(id) = 2

g(id) = 1

g(+) = 0

f(*) = 0

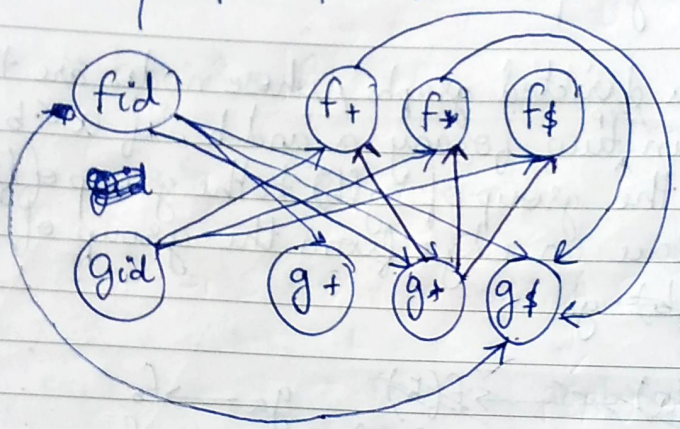
Saathi

Date

Q) operator precedence function for the precedence matrix:

	id	+	*	$
id	·	·>	·>	·>
+	<·	·>	<·	·>
*	<·	·>	·>	·>
$	<·	<·	<·	·

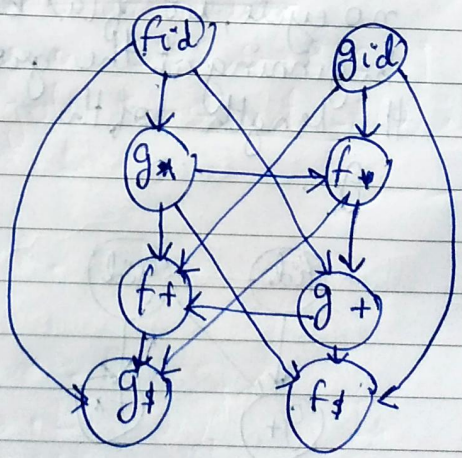
(g) ↑ (f) →



X

f(id) = 3
f(+) = 2
f(*) = 4
f($) =

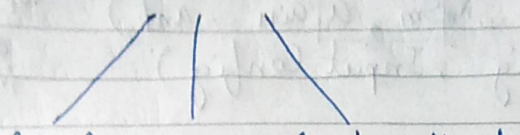
	id	+	*	$
f	5	1	3	0
g	4	2	4	0

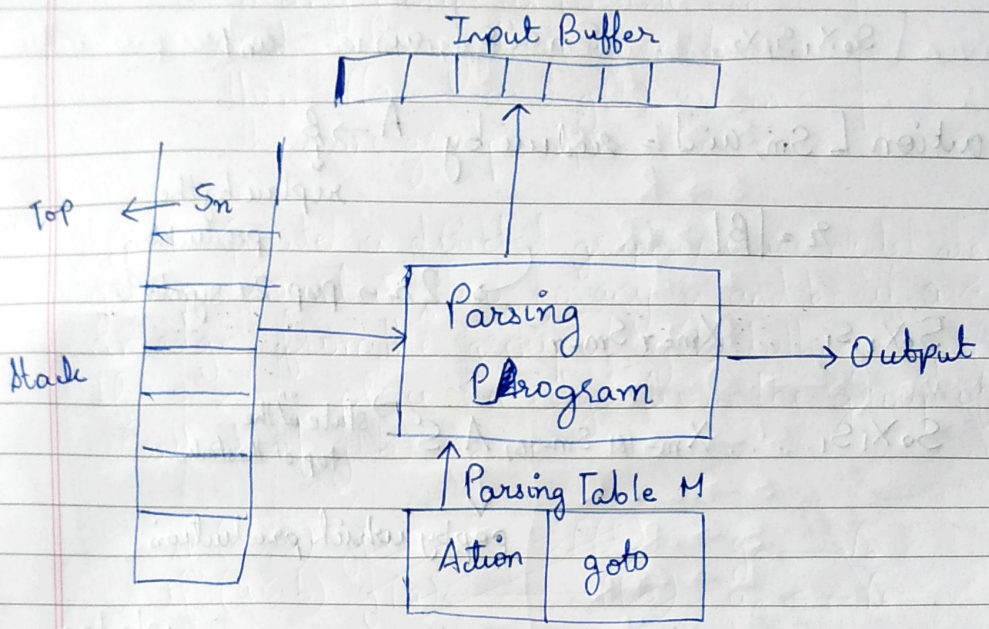


Date ___/___/_____

Saathi

LR Parser





S_i → State

X_i → grammar symbol

$S_0 X_1 S_1 X_2 S_2 X_3 \cdots X_n S_n$ → Stack Configuration

Top of the stack is always S_n or a state if not then error

Action → terminal
Goto → non-terminal / only

r1 → reduce by production no. 1
s6 → shift the handle to 6 (State)

Date ___/___/___

$(S_0 X_1 S_1 X_2 \ldots X_m S_m, a_i a_{i+1} \ldots a_n \$)$

(Stack Config , Input Config)

shift, reduce, accept, error — depends on this top of stack and input symbol.

Shift → action $[S_m, a_i]$ = shift S

$S_0 X_1 S_1 X_2 \ldots S_m a_i S, a_{i+1} \ldots a_n \$$

Reduce → action $[S_m, a_i]$ = reduce by $A \to \beta$

replace by the production

$r = |\beta|$

↳ $2r$ → pop $2r$ symbol

$S_0 X_1 S_1 \ldots X_{m-r} S_{m-r}$

$S_0 X_1 S_1 \ldots X_{m-|\beta|} S_{m-|\beta|} A S$ ← state of the top of the stack.

↓

pop by which production.

13/9/22

SLR parsing table

If G is a grammar, ~~table construction~~

1) Augmented grammar → {S'→S, S→B, C→a}
2) Using closure and goto construct LR(0) item.

$A \to .XYZ$

$A \to X.YZ$

$A \to XY.Z$

$A \to XYZ.$

Date ___/___/_____

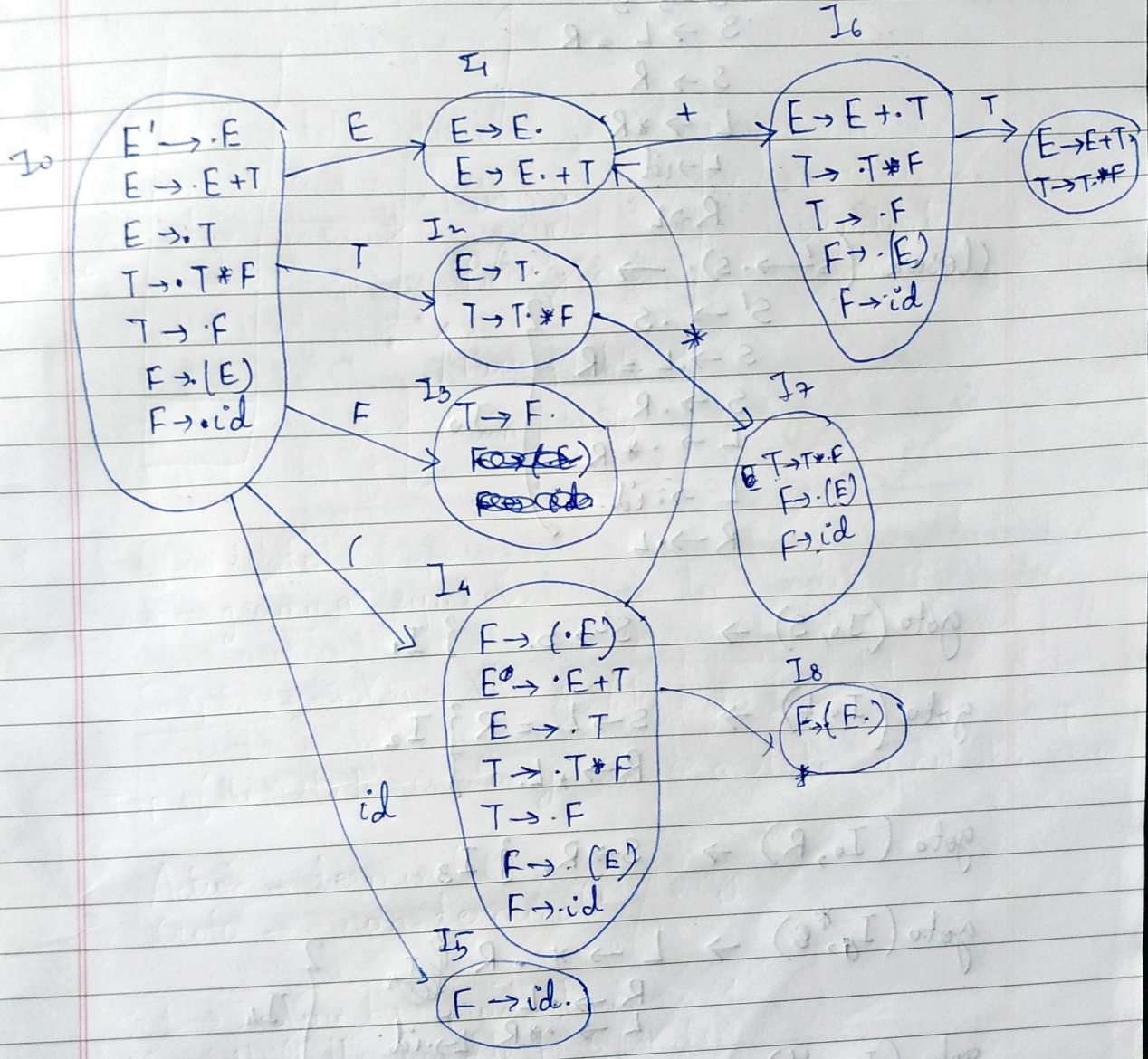
Saathi

Closure Operation

Augmented Grammar

$E' \to E$
$E \to E + T$
$E \to T$
$T \to T * F$
$T \to F$
$F \to (E)$
$F \to id$

$\longrightarrow$ Grammar G'



Date __/__/____

FOLLOW(F) =

19/09/2022

(1) Every SLR(1) 'G' is unambiguous but there are many unambiguous grammar that are not SLR(1).

Ans. Sol. Given an unambiguous grammar, whose productions are:

$S' \to S$
$S \to L = R$
$S \to R$
$L \to *R$
$L \to id$
$R \to L$

Closure $(S' \to .S)$: $\to$

$S' \to .S$
$S \to .L = R$
$S \to .R$
$L \to .*R$
$L \to .id$
$R \to .L$

} I_0

goto $(I_0, S) \to S' \to S.$ } I_1

goto $(I_0, L) \to S \to L. = R$
$R \to L.$ } I_2

goto $(I_0, R) \to S \to R.$ } I_3

goto $(I_0, *) \to L \to *.R$
$R \to .L$
$L \to .*R$
$L \to .id$ } I_4

goto $(I_0, id) \to L \to id.$ } I_5

goto $(I_2, =) \to S \to L = .R$
$R \to .L$
$L \to .*R$
$L \to .id$ } I_6

goto (I_4, R) → $L \to *R. \} I_7$

goto (I_4, L) → $R \to L. \} I_8$

goto $(I_4, *)$ → $L \to *.R \} I_4$

goto (I_4, id) → I_5

goto (I_6, R) → $S \to L = R. \} I_9$

~~goto $(I_6$~~ goto $(I_6, *) \Rightarrow I_4$

goto $(I_6, id_2) \Rightarrow I_5$

FOLLOW(R) = FOLLOW(S) ∪ FOLLOW(L)

FOLLOW(R) = {$, =}

FOLLOW(L) = {=, $}

goto$(I_2, =) = I_6$

So for the item in column =, we see that both s6 and r5 is present. So this grammar is not SLR(1).

In canonical LR we add a terminal or endmarker at the end of each production to remove the disadv. obtained in SLR, So it is known as LR(1).

LR(1) →

$A \to \alpha . B\beta, a$ → endmarker

$B \to .\delta, b$ where $b = FIRST(\beta a)$

closure operation

Goto Operation:

$I_i: A \to \alpha . X \beta, a$

goto(I_i, X)

$I_j: A \to \alpha X . \beta, a$