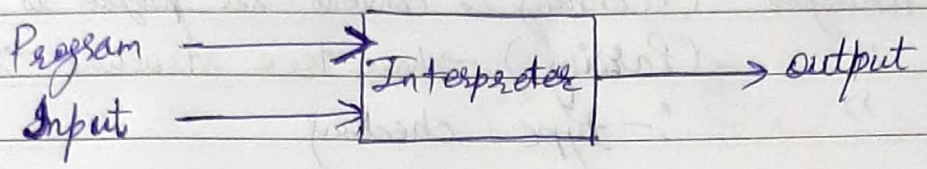
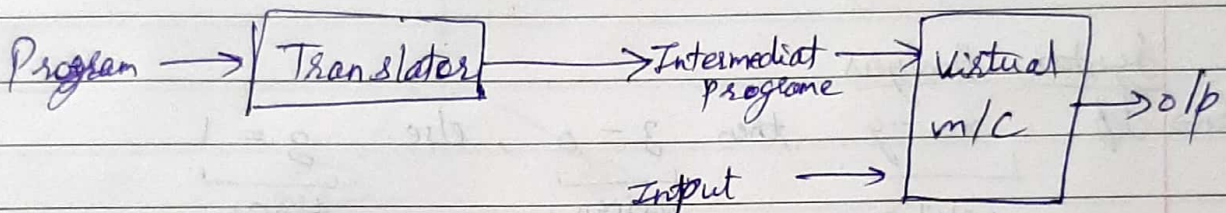
PAGE NO.
DATE / /

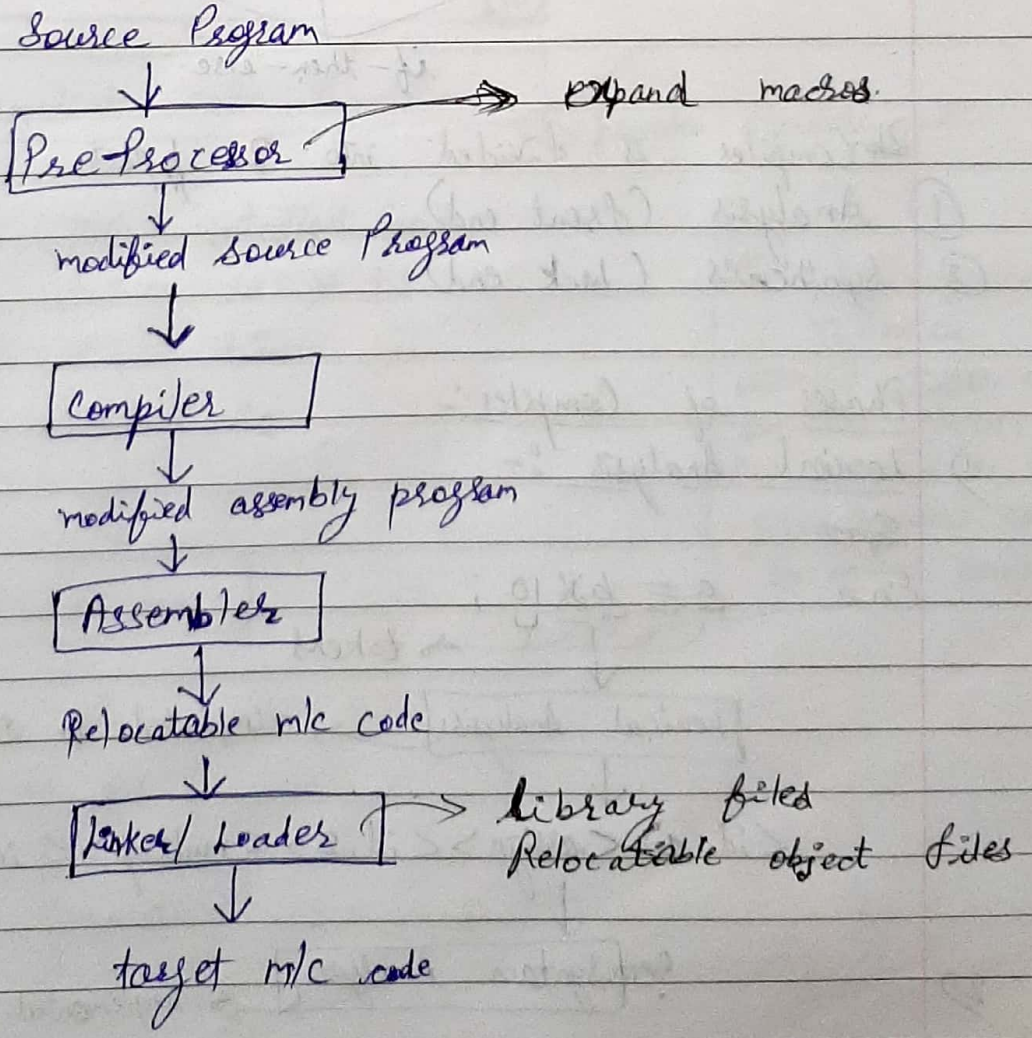
Compiler :→ Convert given file (program) to executable file/code



Hybrid Compiler :



A language Processing System :



First compiler is designed for FORTRAN-1.

Phases for any compiler or FORTRAN

Analysis
- (i) Lexical Analysis (scanning) :- converts src program to tokens
- (ii) Syntax " (Parsing) → have tree structure → PDA
- (iii) Semantic " :- type checking

synthesis
- (iv) Optimisation
- (v) Code generation.

Syntax Analysis

Eg. If x==y then z=0 , else z=1

relation, assign, assign

Predicate, then statement, else stat

if-then-else

Compiler is divided into 2 parts

(1) Analysis (front end)

(2) Synthesis (back end)

Phases of Compiler :-

1) Lexical Analysis :-

Eg:

a = b * 10 ;

→ tokens

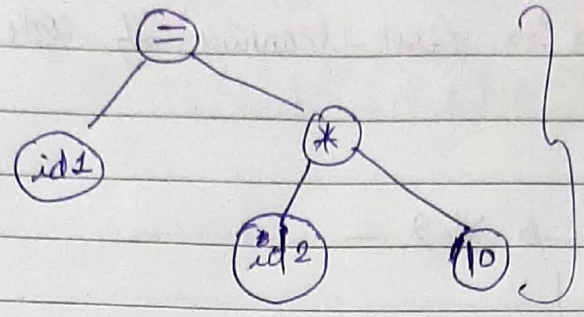
Lexical Analysis → implented by DFA (Deterministic finite automata)

<id, 1> <assign> <id, 2> <multop> <iconst, 10> → tokens

2) Syntax Analyser → implemented by pda

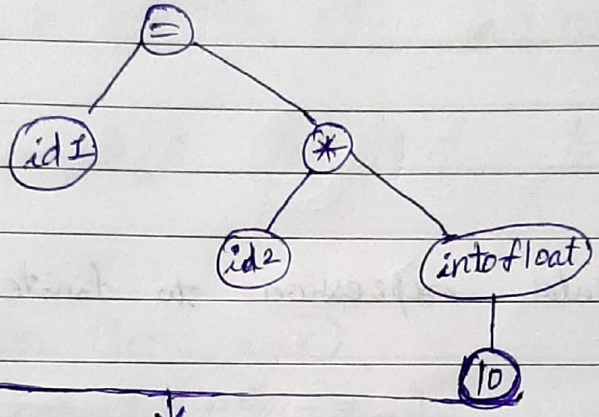
PAGE NO.
DATE / /

2(CFG→PT)



Syntax tree
(derivation tree in CFG).

3) Semantic Analysis



4) Intermediate code Generation → provides standard code

```
t1 = intofloat (10)
t2 = id2 * t1
t3 = t2
id1 = t3
```

Ex:- m languages
n m/cs.
without Inter. code
we need m×n compiler
But intermediate code
give standard code
↳ m+n compiler

5) Optimisation

```
t1 = id2 * 10.0
id1 = t1
```

6) Code Generation

↓

m/c code

1) Lexical Analysis :- first scanning of code then Lexical Analysis

① Tokens

a = b * 2

↓

<id, 1> <assign> <id, 2> <multiplication> <num, 2>

attribute value — token

tokens

② Lexemes

③ Pattern :- Regular expression to finite automata

Lexical Errors :-

↳ Error - Recovery mechanism → inserting character

Error may be type of

fro in place of for → fro i = 0 to n

identifier

<id, 1> <id, 2>

wrong pattern (no error)

(i) int roll = 5

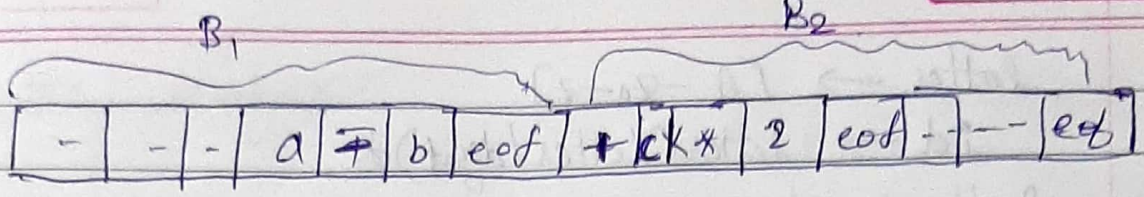
no pattern available

Input Buffering :- use to speed up the reading

Ens roll no. = b * c * 2

lexical beginning pointer

lexical forwarding pointer



Regular Expression :-

$a* = \{\phi, a, aa, \dots\}$

$a+ = \{a, aa, aaa, \dots\}$

$a? = \{\phi, a\} \equiv a/\epsilon$ ⇒ a or ∈

$[abc] = a \mid b \mid c$ (or)

A/B/C/..../Z ≡ A-Z

C - identifier

letter (letter/digit)

where letter → [A-Za-z_]

digit → [0-9]

Ex:- grammer → if-then-else

stmt → if expr then stmt /

if expr then stmt else stmt /

∈

expr → term relop term /

term

(relop → Relation operator)

term → id / number

pattern for tokens

digit → [0-9]

digits → [digits]+

number → digits (.digits)? (E[+-]? digits)?

(E → exponential)

Ex: number = 34, 57.38, 27.5E2

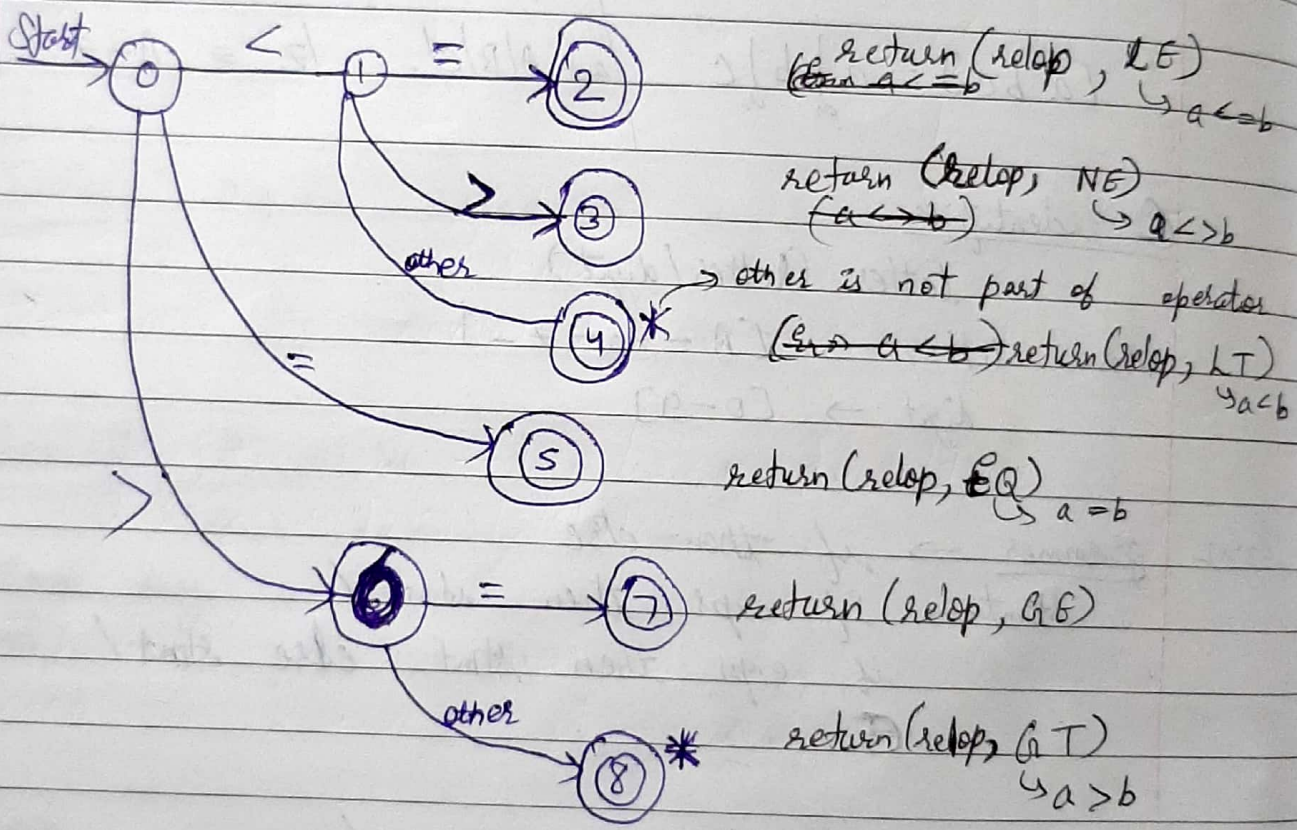
letter → [A-Za-z]

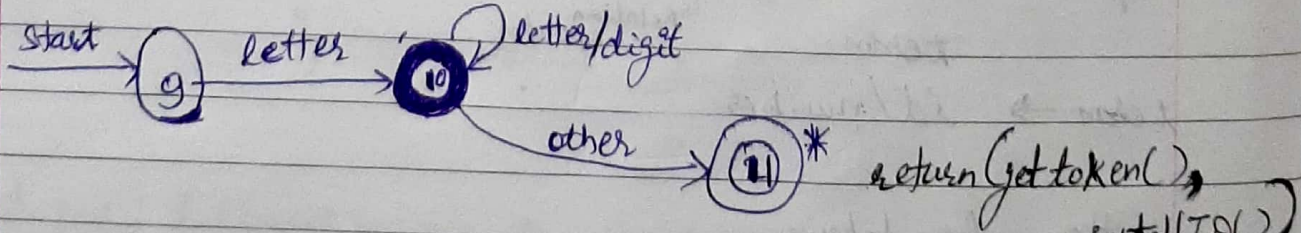
id → letter(letter/digit)*

- if → if
- else → else
- then → then

relop → <|>|<=|>=|=|<> (not equal)

Transition Diagram:-



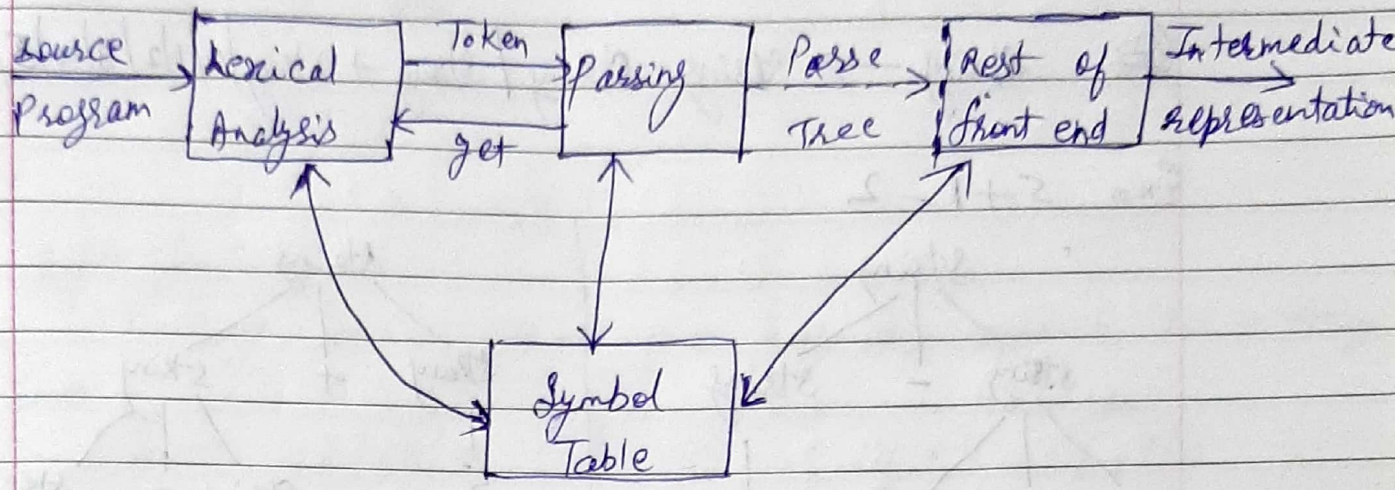


gettoken() → return id no.

installID() → pointer to that id no.

PAGE NO.
DATE / /

(2) Syntax Analysis



1) Top - Down Passing
2) Bottom - Up Parsing

Type of Errors :-

(1) Lexical Errors :- No pattern matching.
(2) Syntax Errors :- curly ({}) bracket, switch case etc.
(3) Semantics " :- type is not matching (int ≠ char).
(4) Logical " :-

Error Recovery Strategies :-

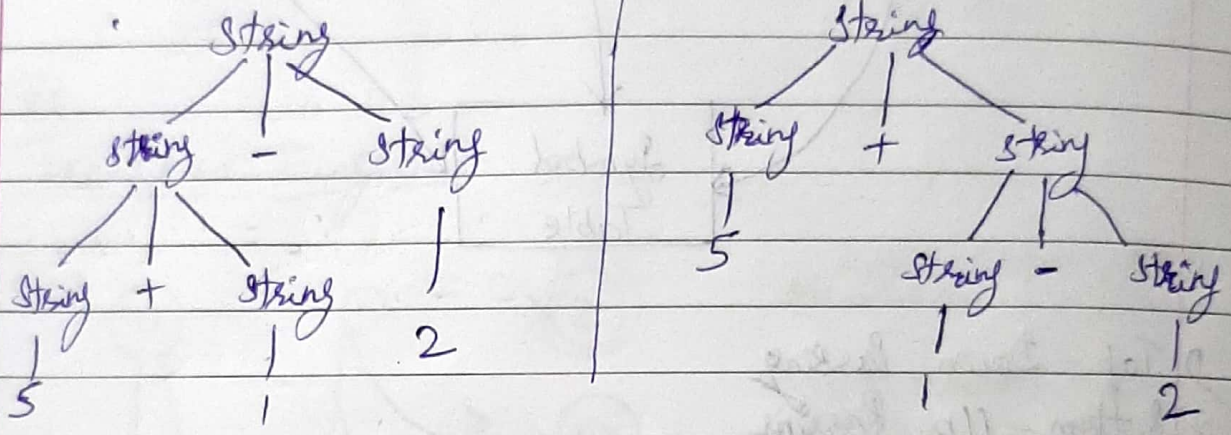
(1) Panic-Mode Recovery
(2) Phrase-level Recovery.
(3) Error Production :-
(4) Global Correction

Grammer:

$G = (V, T, P, S)$

Eg: String → String − String | String + string | 0|1|2|3|4|5|6|7|8|9

Eg: 5 + 1 − 2



→ Ambiguished grammer

→ need to convert to non-ambiguished grammer

Ex,

String → String − digit / string + digit / digit

digit → 0|1|2|3|4|5|6|7|8|9.

→ Top-down approach uses left-most derivative

→ need to remove left-recursion.

Eg.

if $A \rightarrow A\alpha BB$

$\Downarrow$

$A \rightarrow BB$

$B \rightarrow \alpha B / \epsilon$

Eg: Remove left-recursion from

$E \rightarrow E + T / T$

$T \rightarrow T * F / F$

$F \rightarrow (E) / id.$

PAGE NO.
DATE / /

$E \rightarrow T\ B$

$B \rightarrow +T\ B / \epsilon$

$T \rightarrow F \cdot C$

$C \rightarrow *F\ C / \epsilon$

$F \rightarrow (E) / id$

Left factoring :-

↳ Have more than one way to approch for a string

Ex.

$A \rightarrow aB / aA / a$

$B \rightarrow b$

↓ Left factoring

$A \rightarrow aC$

$C \rightarrow B / A / \epsilon$

$B \rightarrow b$

Ex.

$S \rightarrow iEtS / iEtes / a$

$E \rightarrow b$

↓

$S \rightarrow iEtSA / a$

$A \rightarrow eS / \epsilon$

$E \rightarrow b$

★ Top-Down Parsing :- (uses ~~Left-most~~ derivation)

Recusive-Desent Passing :-

```
void A( ){
  choose an A-production, A → X1 X2 ---- Xk ;
  for (i = 1 to k) {
      if (Xi is a non terminal)
          call procedure Xi( );
      else if (Xi equals the current i/p symbol 'a')
          advance the i/p to the next symbol;
      else
          /* an error has occured */ ;
```

first( ) & follow( )

Eg:

$S \rightarrow aAb/b$

$A \rightarrow bB$

$B \rightarrow a$

$first(S) = first(aAb) \cup first(b)$

$= \{a\} \cup \{b\}$

$= \{a, b\}$.

$first(A) = \{b\}$

$first(B) = \{a\}$.

* First(X)

1. If X is a terminal, then $first(X) = \{X\}$.
2. If X is non-terminal, and $X \rightarrow Y_1 Y_2 \dots Y_k$

Eg:- If $first(Y_1)$ does not derive ϵ, then $first(X) = first(Y_1)$

But if $first(Y_1)$ derives ϵ, then

$first(X) = first(Y_1) \cup first(Y_2)$

and so on.

3. If $X \rightarrow \epsilon$ is a production then add ϵ to first(X)

Q. (1) Find first( ) for given grammer.

$E \rightarrow TE'$

$E' \rightarrow +TE'/\epsilon$

$T \rightarrow FT'$

$T' \rightarrow *FT'/\epsilon$

$F \rightarrow (E)/id$

PAGE NO.
DATE / /

First(E) = {(, id}.
first(E') = {+, ∈}
First(T) = {(, id}
first(T') = {*, ∈}
first(F) = {(, id}

* follow( ) ⇒ for acceptance of string
(1.) Place $ in follow(S) where S is start symbol
(2) If A → αBβ, then
FOLLOW(B) = FIRST(β) − {∈}.
(3) If A → αBβ and FIRST(β) contain ∈,
then FOLLOW(B) = FOLLOW(A).

Ex⇒ follow for previous question
FOLLOW(E) = {$} for E → TE'
FOLLOW(T) = FIRST(E') − {∈} ∪ follow(E)
= FIRST({+, ∈}) − {∈} ∪ {$}
= {+} ∪ {$}
= {+, $} — ①

FOLLOW(E') = FOLLOW(E) = {$} — ②

for E' → +TE'
FOLLOW(T) = FIRST(E') − {∈} ∪ FOLLOW(E)
= {+} ∪ {$}
= {+, $} — ③

for $T \to FT'$

$FOLLOW(F) = FIRST(T') - \{\epsilon\} \cup FOLLOW(T)$

$= \{*\} \cup \{+, \$\}$

$= \{*, +, \$\}$ ---(4)

$FOLLOW(T') = FOLLOW(T) = \{+, \$\}$ ---(5)

for $T' \to *FT'$

$FOLLOW(F) = FIRST(T'') - \{\epsilon\} \cup FOLLOW(T')$

$= \{*\} \cup \{+, \$\}$

$= \{*, +, \$\}$ ---(6)

for $F \to (E)$

$FOLLOW(E) = First())$

$= \{)\}$ ---(7)

for eq. (2) & (4)

$FOLLOW(E) = \{), \$\}$.

Iterate till Follow of start symbol is fixed.

Here Starting symbol E has updated at last so, need to iterate whole production with updated value of FOLLOW(E).

for $E \to TE'$

$FOLLOW(T) = First(E') - \{\epsilon\} \cup FOLLOW(E)$

$= \{+\} \cup \{), \$\}$

$= \{+,), \$\}$

$FOLLOW(E') = FOLLOW(E) = \{), \$\}$.

PAGE NO.
DATE / /

(Q2) Find FIRST() & FOLLOW() for given grammer

$S \rightarrow ACB/CbB/Ba$

$A \rightarrow da/BC$

$B \rightarrow g/\epsilon$

$C \rightarrow h/\epsilon$

Ans) FIRST(C) = {h, ϵ}

FIRST(B) = {g, ϵ}

FIRST(A) = {d} ∪ FIRST(B) = {d} ∪ {g, ϵ} = {d, g, ϵ}

FIRST(S) = FIRST(A) ∪ FIRST(C) ∪ FIRST(B)

= {d, g, ϵ} ∪ {h, ϵ} ∪ {g, ϵ}

= {d, g, h, ϵ}.

FIRST(A) = {d} ∪ FIRST(B) − {ϵ} ∪ FIRST(C)

= {d, g, h, ϵ}.

FIRST(S) =

= {d, g, h, ϵ}.

follow(S) = $

follow(A) = first(C) − {ϵ} ∪ first(B) − {ϵ} ∪ follow(S)

PAGE NO.
DATE / /

LL(1) grammer :-

L → left-to-right
L → Leftmost derivation
1 → one i/p symbol of lookahead.

Condition :-

A → α/β

1) for terminal 'a', both α & β should not derive strings begining with 'a'.

2) Atmost one of α & β can derive empty string
↳ first(α) & first(β) are disjoint set.

3) If $\beta \overset{*}{\Rightarrow} \epsilon$, then α does not derive any string begining with a terminal in a terminal in follow(A)
Likewise, if $\alpha \overset{*}{\Rightarrow} \epsilon$

H.W. Check is ①-② is LL(1) grammer or not ?

Parsing Table →

Eg.

$E \rightarrow TE'$
$E' \rightarrow +TE'/\epsilon$
$T \rightarrow FT'$
$T' \rightarrow *FT'/\epsilon$
$F \rightarrow (E)/id$

So,

first(E) = {(, id}	FOLLOW(E) = {), $}
first(E') = {+, ε}	(E') = {), $}
FIRST(T) = {(, id}	(T) = {+,), $}
FIRST(T') = {*, ε}	(T') = {+,), $}
FIRST(F) = {(, id}.	(F) = {+, *,), $}.

PAGE NO.
DATE / /

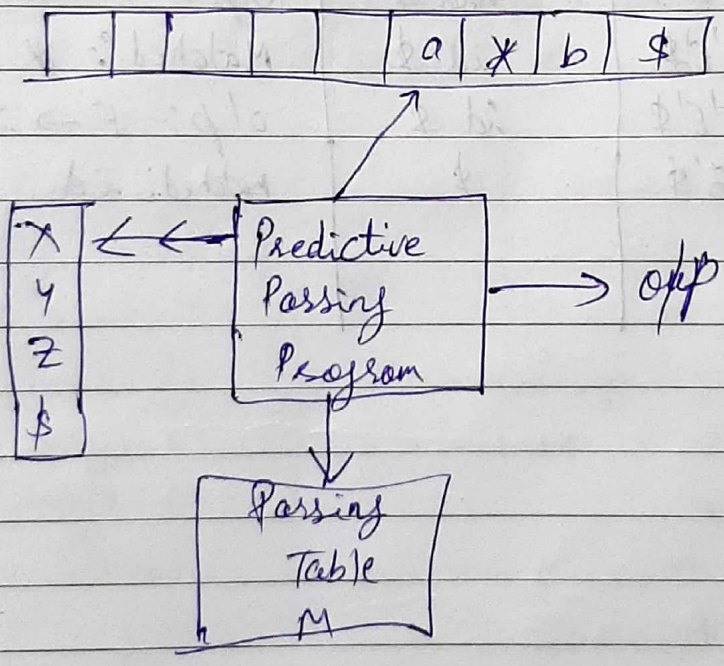
(1) first see the FIRST(E) of product ~~X → TE~~ E → TE'. for all FIRST(E) value, insert the entry in table. if E → ∈ then see FOLLOW of (E), do same for FOLLOW(E) values.

Ex.

Non-Terminal	id	+	*	(	)	$
E	E → TE'			E → TE'		
E'		E' → +TE'			E' → ~~∈~~	E' → ~~∈~~
T	T → FT'			T → FT'		
T'		T' → ∈	T' → *FT'		T' → ∈	T' → ∈
F	F → id			F → (E)		

Ex.

Non-recursive Predictive Passing :-



Self study → Error Recovery → pg → 128 → Ullman

PAGE NO.
DATE / /

~~matched~~ String :- id + id * id

Match	Stack	Input	Output/Action.
	$E\$$	$id+id*id\$$	~~o/p → E → TE'~~
	$TE'\$$	,,	o/p :- $E \to TE'$
	$FT'E'\$$	,,	$T \to FT'$
	$idT'E'\$$	$id+id*id\$$	~~o/p~~ :- $F \to id$
id	$T'E'\$$	$+id*id\$$	Matched :- id
id	$E'\$$	$+id*id\$$	o/p :- $T' \to \epsilon$
id+	$+T'E'\$$	$+id*id\$$	~~matched~~ o/p :- $E' \to +TE'$
id+	$TE'\$$	$id*id\$$	Matched :- +
,,	$FT'E'\$$	$id*id\$$	o/p :- $T \to FT'$
	$idT'E'\$$	$id*id\$$	o/p :- $F \to id$
id+id	$T'E'\$$	$*id\$$	Matched :- id
,,	$*FT'E'\$$	$*id\$$	o/p :- $T' \to *FT'$
id+id*	$FT'E'\$$	$id\$$	Matched :- *
id+id*	$idT'E'\$$	$id\$$	o/p :- $F \to id$
id+id*id	$T'E'\$$	$\$$	matched : id.
id+id*id			

PAGE NO.
DATE / /

☆ Bottom-Up Parsing :- Right-most derivatives.
↳ Left-to-right scanning.

Let the grammer be

$E \to E+T$

$E \to T$

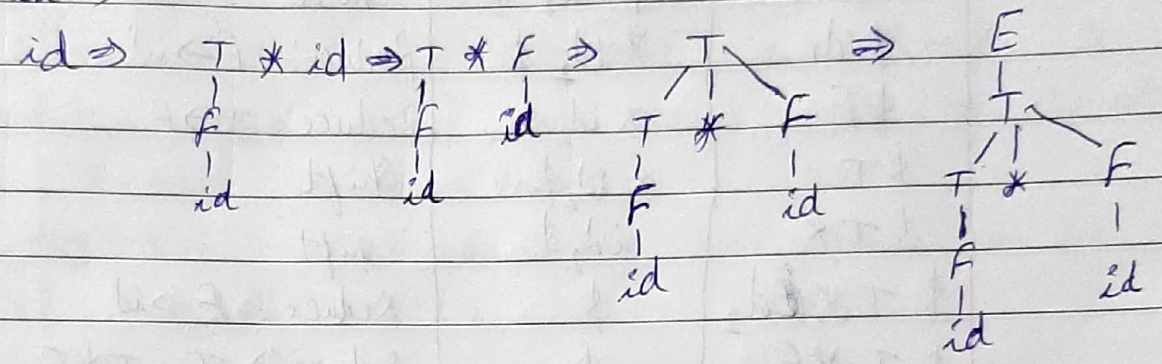
$T \to T*F$

$T \to F$

$F \to id$

⇒ $id * id$ ⇒

⇒ $F * id$ ⇒ $T * id$ ⇒ $T * F$ ⇒



Handle :- symbol of string which is replaced by single non-terminal.

$S \to aAcBe$

$A \to Ab/b$

$B \to d$

String = abbcde

$S \to aAcBe$ ↑ RMD

$\to aAcde$ — Handle

$\to aAbcde$

$\to abbcde$ ↳ Handle

$E \to E+E$

$E \to E*E$

$E \to (E)$

$E \to id$

String = $id + id * id$ using RMD

$E \to E+E$	$E \to E*E$
$\to E+\underline{E*E}$	$\to E*id$
$\to E+E*\underline{id}$	$\to E+E*id$
$\to E+\underline{id}*id$	$\to E+id*id$
$\to \underline{id}+id*id$	$\to id+id*id$

↓ Ambiguish grammer.

→ Need Un-ambiguish grammer for parsing.

PAGE NO.
DATE / /

Shift-Reduce Passing :→

$E \rightarrow E+T/T$

$T \rightarrow T*F/F$

$F \rightarrow (E)/id$

String :- $id_1 * id_2$

Stack	input	Action
$	$id_1 * id_2$ $	Shift
$ id_1	$* id_2$ $	Reduce :- $F \rightarrow id$
$ F	$* id_2$ $	Reduce :- $T \rightarrow F$
$ T	$* id_2$ $	Shift
$ T *	id_2 $	Shift
$ T * id_2	$	Reduce :- $F \rightarrow id$
$ T * F	$	Reduce :- $T \rightarrow T*F$
$ T	$	Reduce :- $E \rightarrow T$
$ E	$	Accept

Simple LR (SLR) :- first create DFA for passing table.

LR(k)

L = Left-to-right

R = Right Most derivative

k = no. of i/p symbol for lookhead.

for default k=1 i.e. LR = LR(1).

Items for $A \rightarrow XYZ$ are

$A \rightarrow \cdot XYZ$

$A \rightarrow X \cdot YZ$

$A \rightarrow XY \cdot Z$

$A \rightarrow XYZ \cdot$

For SLR → need canonical LR → canonical items.

PAGE NO.
DATE / /

For LR(0) automata → DFA, two func" needed

① ~~Closure ( )~~

① ~~Clo~~ CLOSURE( )

② GOTO( )

→ If I is a set of items of some grammer G, the CLOSURE(I) is set of items constructed from 2 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, 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).

Eg.

$E \to E+T/T$

$T \to T*F/F$

$F \to id.$

Add $S \to E$ (blc ~~starting symbol~~ E may ~~appear~~ in RHS) → Augemented grammer

Now closure $(S \to .E) = \{S \to .E,\ E \to .E+T,\ E \to .T,$
$T \to .T+F,\ T \to .F,\ F \to .id\}$ ---- I_0

↑ state

Apply GOTO() on all which have . before T or Variables.

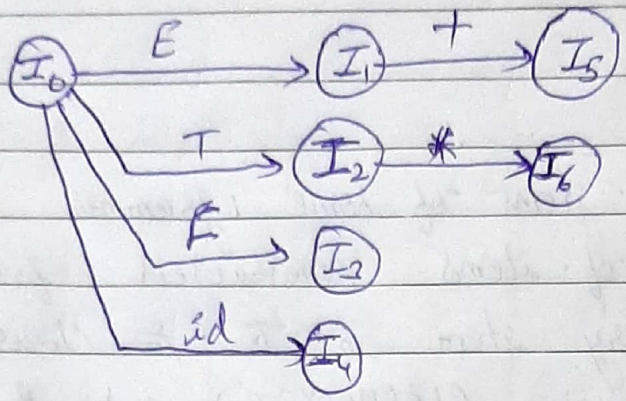
GOTO(I_0, E) =
= closure $(S \to E.,\ E \to E.+T)$
= $\{S \to E.,\ E \to E.+T\}$ --- I_1

GOTO(I_0, T) = closure $(E \to T.,\ T \to T.*F)$
= $\{E \to T.,\ T \to T.*F\}$ --- I_2

GOTO(I_0, F) = closure $(T \to F.)$
= $\{T \to F.\}$ --- I_3

PAGE NO.
DATE / /

GOTO(I_0, id) = closure($F \to id.$)
= { $F \to id.$ } ---- I_4



GOTO(I_1, +) = closure ($E \to E+.T$)
= { $E \to E+.T$,
$T \to .T*F$,
$T \to .F$, $F \to .id$ } ---- I_5

GOTO(I_2, *) = Closure($T \to T*.F$)
= { $T \to T*.F$, $F \to .id$ } ---- I_6

GOTO(I_5, T) = Closure($E \to E+T.$, $T \to T.*F$)
= { $T \to T*.F$, $F \to .id$ } ---- I_7

GOTO(I_5, F) = closure($T \to F.$)
= { $T \to F.$ } ---- I_3

GOTO(I_5, id) = closure($F \to id.$)
= { $F \to id.$ } ---- I_4

GOTO(I_6, F) = closure($T \to T*F.$)
= { $T \to T*F.$ } ---- I_8

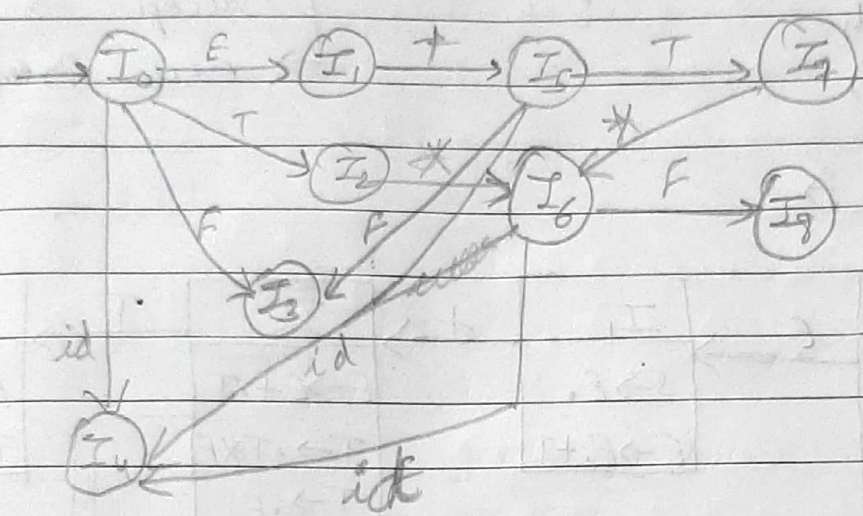
GOTO(I_6, id) = Closure($F \to id.$)
= { $F \to id.$ } ---- I_4

PAGE NO.
DATE / /

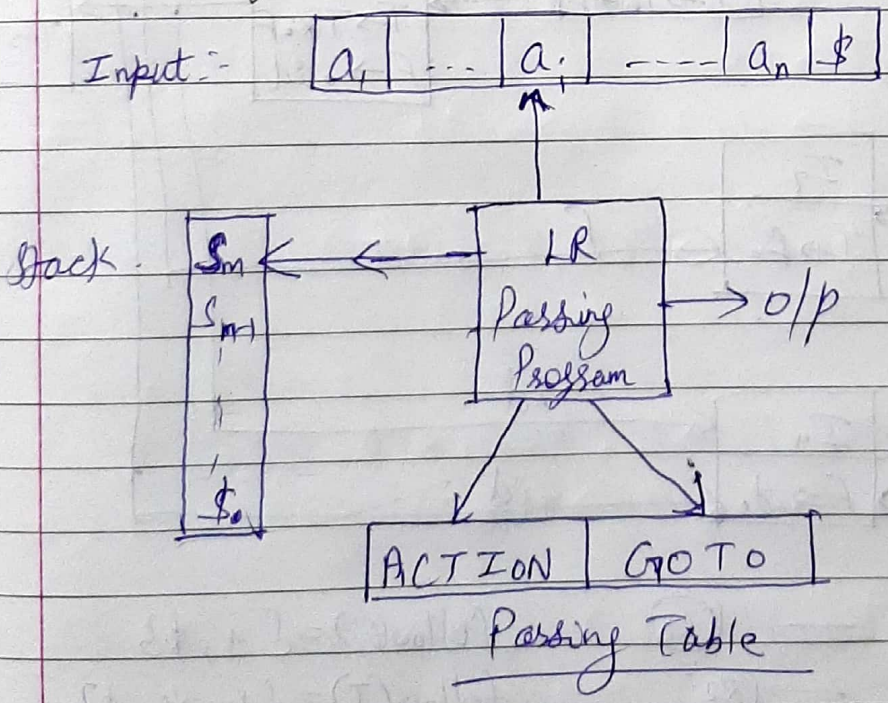
GOTO(I_7, *) = closure($T \to T * .F$)

= { $T \to T * .F$,

$F \to .id3$ ---- I_6



LR -Parsing Model :->



PAGE NO.
DATE / /

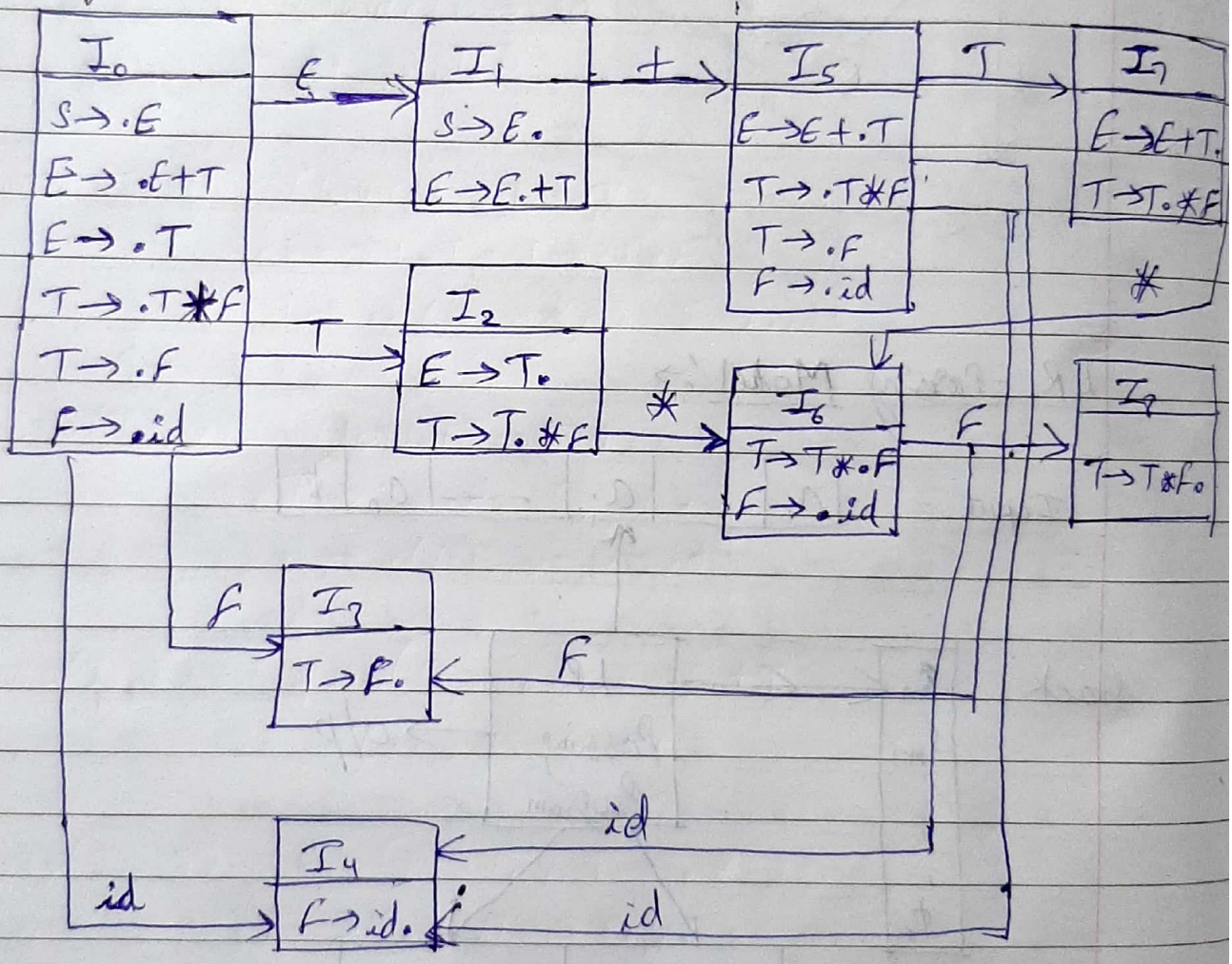
LR Parser Configuration :→

$(S_0 S_1 \dots S_m, a_i a_{i+1} \dots a_n)$

- $S_0 S_1 \dots S_m$ → Stack Symbols (States)
- $a_i a_{i+1} \dots a_n$ → i/p Symbol

→ $(S_0 S_1 \dots S_m S, a_{i+1} \dots a_n)$
→ $(S_0 S_1 \dots S_{m-r} S, a_i a_{i+1} \dots a_n)$
→ accept → $ or $
→ error → empty cell

Parsing Table :→



Let, $E \to E+T$ ——— (R1)
$E \to T$ ——— (R2)
$T \to T*F$ ——— (R3)
$T \to F$ ——— (R4)
$F \to id$ ——— (R5)

follow(E) = {+, $}
follow(T) = {+, *, $}
follow(F) = {+, *, $}

→ If an item has • at end of production in any state then reduce that item.
For ex → in I_2 :- $E \to T.$ then Reduce it in its follow (E)

PAGE NO.
DATE / /

States	Action				GOTO		
	id	+	*	$	E	T	F
I_0	S4				1	2	3
I_1		S5		accept			
I_2		R2	S6	R2			
I_3		R4	R4	R4			
I_4		R5	R5	R5			
I_5	S4					7	3
I_6	S4						8
I_7		R1	S6	R1			
I_8		R3	R3	R3			

Shift
Reduce

HW:- $E \to E+T$ create LR(1) passing table for given grammar
$E \to T$
$T \to T*F$
$T \to F$
$F \to (E)$
$F \to id$

Soln Adding more production, $S \to E$

Now

Closure $(S \to .E)$

$= \{S \to .E,\ E \to .E+T,\ E \to .T,\ T \to .T*F,\ T \to .F$
$F \to .(E),\ F \to .id\}$ ---- (I_0)

GOTO(I_0, E) = closure$(S \to E.,\ E \to E.+T)$
$= \{S \to E.,\ E \to E.+T\}$ -- (I_1)

GOTO(I_0, T) = Closure$(E \to T.,\ T \to T.*F)$
$= \{E \to T.,\ T \to T.*F\}$ --- (I_2)

PAGE NO.
DATE / /

GOTO(I_0, F) = Closure($T \to F.$)
= $\{T \to F.\}$ — — (I_9)

GOTO(I_0, ( ) = Closure($F \to (.E)$)
= $\{F \to (.E), E \to .E+T, E \to .T,$
$T \to .T*F, T \to .F,$
$F \to .(E), F \to .id\}$ — — — (I_4)

GOTO(I_0, id) = Closure($F \to id.$)
= $\{F \to id.\}$ — — (I_5)

GOTO(I_1, +) = Closure($E \to E+.T$)
= $\{E \to E+.T, T \to .T*F,$
$T \to .F, F \to .(E), F \to .id\}$ — — — (I_7)

GOTO(I_2, *) = Closure($T \to T*.F$)
= $\{T \to T*.F,$
$F \to .(E), F \to .id\}$ — — — (I_7)

GOTO(I_4, E) = Closure($F \to (E.), E \to E.+T$)
= $\{F \to (E.), E \to E.+T\}$ — — — (I_8)

GOTO(I_4, T) = Closure($E \to T., T \to T.*F$)
= $\{E \to T., T \to T.*F\}$ — — — — (I_2)

GOTO(I_4, F) = Closure($T \to F.$) = $\{T \to F.\}$ — — (I_3)

GOTO(I_4, ( ) = Closure($F \to (.E)$)
= $\{$ $\}$ — — — (I_4)

GOTO(I_4, id) = Closure($F \to id.$) = $\{$ $\}$ — — — (I_5)

PAGE NO.
DATE / /

GoTO(I_6, T) = Closure($E \to E+T.$, $T \to T.*F$)
= { $E \to E+T.$, $T \to T.*F$ } ---- I_9

GoTO(I_6, F) = Closure($T \to F.$) = { } ---- I_3

GoTO(I_6, C) = Closure($F \to (.E)$)
= { } ---- I_4

GoTO(I_6, id) = Closure($F \to id.$) = { } ---- I_5

GoTO(I_7, F) = Closure($T \to T*F.$)
= { $T \to T*F.$ } ---- I_{10}

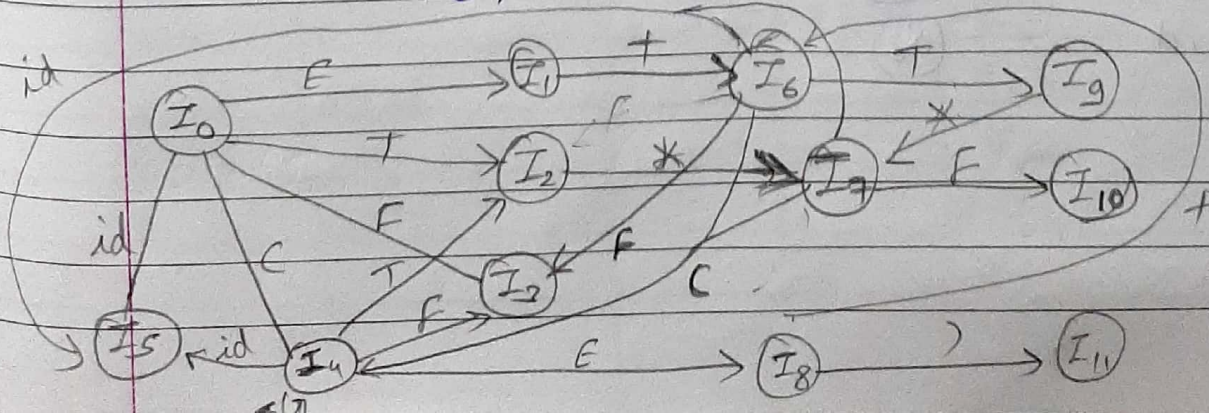
GoTO(I_7, C) = Closure($F \to (.E)$) = { } ---- I_4

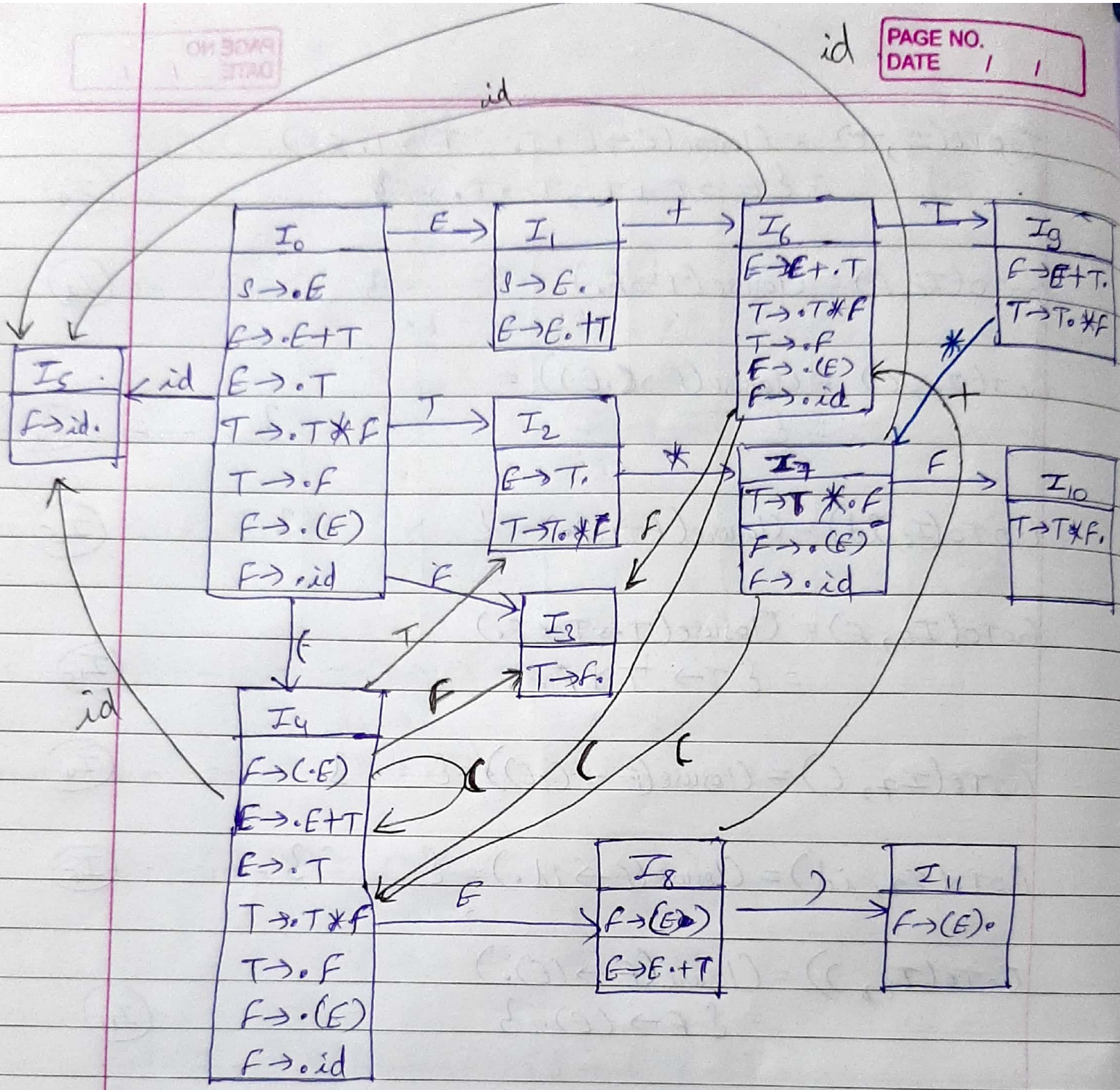
GoTO(I_7, id) = Closure($F \to id.$) = { } ---- I_5

GoTO(I_8,)) = Closure($F \to (E).$)
= { $F \to (E).$ } ---- I_{11}

GoTO(I_8, +) = Closure($E \to E+.T$)
= { $E \to E+.T$, $T \to .T*F$, $T \to .F$
$F \to .(E)$, $F \to .id$ } ---- I_6

GoTO(I_9, *) = Closure($T \to T*.F$)
= { $T \to T*.F$, $F \to .(E)$, $F \to .id$ } ---- I_7





$E \to E+T$ ---- (R1)
$E \to T$ ---- (R2)
$T \to T*F$ ---- (R3)
$T \to F$ ---- (R4)
$F \to (E)$ ---- (R5)
$F \to id$ ---- (R6)

follow(E) = {+,), $}
follow(T) = {*, +,), $}
follow(F) = {*, +,), $}

States	Action						GOTO		
	id	(	)	+	*	$	E	T	F
I_0	S5	S4					1	2	3
I_1			R1	S6		accept			
I_2			R2	R2	S7	R2			
I_3			R4	R4	R4	R4			
I_4	S5	S4						2	3
I_5			R6	R6	R6	R6			
I_6	S5	S4						9	3
I_7	S5	S4							10
I_8			S11	S6		R5			
I_9			R1	R1	S7	R1			
I_{10}			R3	R3	R3	R3			
I_{11}									

Checking for string id * id + id

Stack	Symbol	Input	Action
0		id * id + id $	shift
05	id	* id + id $	Reduce: F → id
03	F	* id + id $	Reduce: T → F
02	T	* id + id $	shift
027	T *	id + id $	Shift
0275	T * id	+ id $	Reduce: F → id
02710	T * F	+ id $	Reduce: T → T * F
02	T	+ id $	Reduce: E → T
01	E	+ id $	Shift
016	E +	id $	Shift
0165	E + id	$	Reduce: F → id
0163	E + F	$	Reduce: T → F
0	E + T	$	

PAGE NO.
DATE / /

✱ Canonical LR (CLR/LR) :->

General item

$[A \to \alpha . B, a]$
↳ Lookhead symbol

where $A \to \alpha B$

a = terminal or $\$$

if $B \neq \epsilon$

then 'a' (Lookhead symbol) has no effect on an item

else if $B = \epsilon$ or $[A \to \alpha ., a]$

then $[A \to \alpha ., a]$ reduces to $A \to \alpha$, only if the i/p symbol is 'a'.

→ Formally, LR(1) item $[A \to \alpha . B, a]$ is valid for a viable prefix γ if ~~deviation~~ there is a deviation.

$$S \overset{*}{\underset{rm}{\Longrightarrow}} \delta A w \underset{rm}{\Longrightarrow} \delta \alpha B$$

Viable prefix = symbol before handle.

where

1. $\gamma = \delta\alpha$
2. Either 'a' is first symbol of 'w' or w is ϵ and 'a' is $\$$ — FIRST(w).

Ex: $S \to L = R / R$

$L \to . * R / id$

$R \to L.$

Add, $S' \to S$

Closure $([S' \to .S, \$]) = \{ [S' \to .S, \$],$

$[S \to .L = R, \$], [S \to .R, \$],$

$[L \to . * R, =], [L \to . id, =],$

$[R \to . L, \$],$

$[L \to . * R, \$], [L \to . id, \$] \}$ ... I_0

GOTO(I_0, S) = Closure([S' → S·, $])
= {[S' → S·, $]} ---------- ($I_1$)

GOTO(I_0, L) = Closure([S → L· = R, $], [R → L·, $])
= {[S → L· = R, $], [R → L·, $]} --- ($I_2$)

GOTO(I_0, R) = Closure([S → R·, $])
= {[S → R·, $]} ---------- ($I_3$)

GOTO(I_0, *) = Closure([L → *·R, =/$])
= {[L → *·R, =/$], [R → ·L, =/$]
[L → ·*R, =/$], [L → ·id, =/$]} --- (I_4)

GOTO(I_0, id) = Closure([L → id·, =/$])
= {[L → id·, =/$]} ---------- ($I_5$)

GOTO(I_2, =) = Closure([S → L = ·R, $])
= {[S → L = ·R, $], [R → ·L, $],
[L → ·*R, $], [L → ·id, $]} ---- ($I_6$)

GOTO(I_4, R) = Closure([L → *R·, =/$])
= {[L → *R·, =/$]} ---------- ($I_7$)

GOTO(I_4, L) = Closure([R → L·, =/$])
= {[R → L·, =/$]} ---------- ($I_8$)

GOTO(I_4, *) = Closure([L → *·R, =/$])
= {[L → *·R, =/$], [R → ·L, =/$]} ---------- (I_4)
[L → ·*R, =/$], [L → ·id, =/$] ---------- (I_4)

GOTO(I_4, id) = Closure([L → id·, =/$])
= {[L → id·, =/$]} ---------- ($I_5$)

PAGE NO.
DATE / /

GoTO(I_6, R) = Closure([S → L = R·, $])
= {[S → L = R·, $]} --- ($I_9$)

GoTO(I_6, L) = Closure([R → L·, $])
= {[R → L·, $]} --- ($I_{10}$)

GoTO(I_6, *) = Closure([L → *·R, $])
= {[L → *·R, $], [R → ·L, $]
[L → ·*R, $], [L → ·id, $]} --- ($I_{11}$)

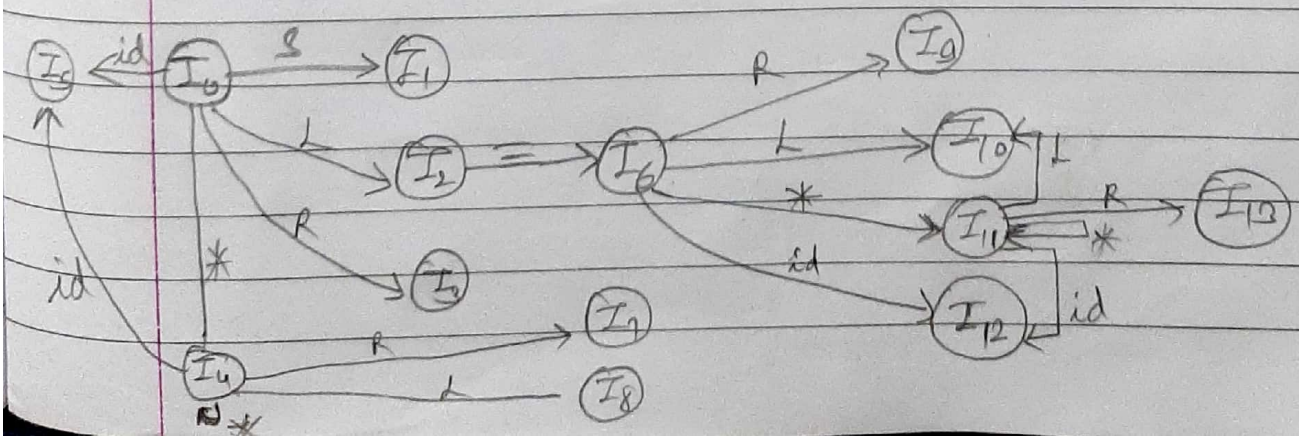
GoTO(I_6, id) = Closure([L → id·, $])
= {[L → id·, $]} --- ($I_{12}$)

GoTO(I_{11}, R) = Closure([L → *R·, $])
= {[L → *R·, $]} --- ($I_{13}$)

GoTO(I_{11}, L) = Closure([R → L·, $])
= {[R → L·, $]} --- ($I_{10}$)

GoTO(I_{11}, *) = Closure([L → *·R, $])
= {[L → *·R, $], [R → ·L, $]
[L → ·*R, $], [L → ·id, $]} --- ($I_{11}$)

GoTO(I_{11}, id) = Closure([L → id·, $])
= {[L → id·, $]} --- ($I_{12}$)



PAGE NO.
DATE / /

$S \to L = R$ — (R1)
$S \to R$ — (R2)
$L \to *R$ — (R3)
$L \to id$ — (R4)
$R \to L$ — (R5)

follow(S) = {$}
follow(L) = {=, $}
follow(R) = {=, $}

LR Passing Table:

State	Action				S	L	R
	id	*	=	$			
I_0	S5	S4			1	2	3
I_1				accept			
I_2			S6	R5			
I_3				R2			
I_4	S5	S4				8	7
I_5			R4	R4			
I_6	S12	S11				10	9
I_7			R3	R3			
I_8			R5	R5			
I_9				R1			
I_{10}				R5			
I_{11}	S12	S11				10	13
I_{12}				R4			
I_{13}				R3			