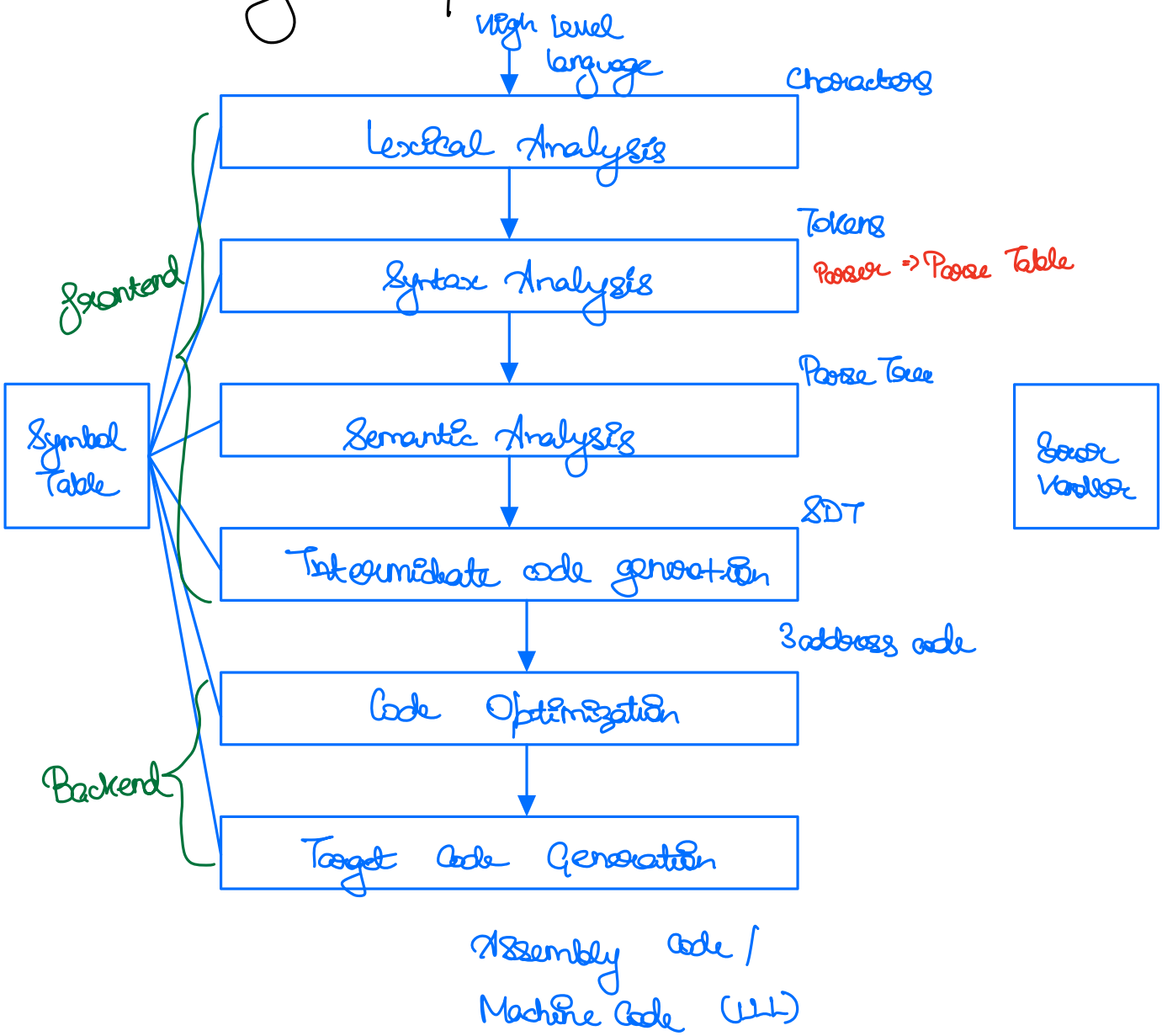
Compiler

Compiler is used to convert between high level language and low level language.

Phases of Compiler



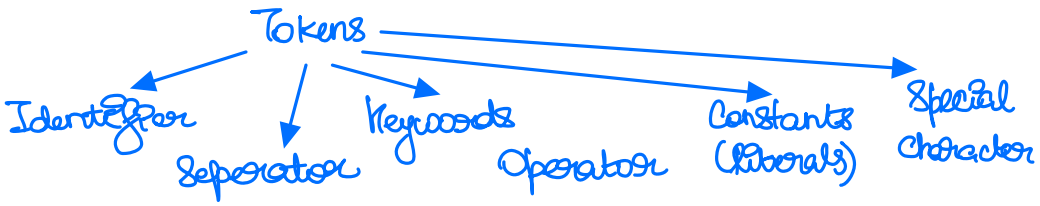
Lexical Analysis

→ Input: characters, output: tokens

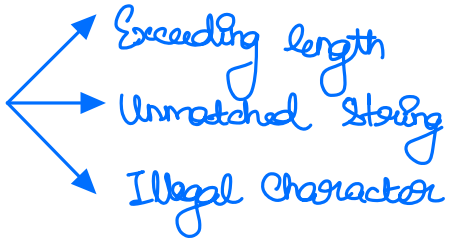
→ Done by Lexer (aka tokenizer, scanner)

Roles

1) Tokenization



2) Give Error Messages



3) Eliminate comments, whitespaces

It is an application of finite Automata and uses DFA / NFA

First()

First(A) contains all terminals present in first place of every string derived by A.

S → abc | def | ghi ⇒ Ans: a, d, g

First (Terminal)

First (ϵ) = ϵ

→ S → ABc | gHi | gd ⇒ a, b, c, g, g

A → a | b | c ⇒ a, b, c

B → b ⇒ b

D → d ⇒ d

→ S → ABd ϵ ϵ ϵ a, b, c, d, e, g, ϵ

A → a | b | ϵ a, b

B → c | d | ϵ c, d

C → e | g | ϵ e, g

→ E → TE' Id, C

E' → *TE' | ϵ *, ϵ

T → FT' Id, C

T' → ϵ | +FT ϵ, +

F → Id | (E Id, C

Follow ( )

Follow(A) contains set of all terminals present immidiate in right of A.

Rules

→ Follow of start symbol is $

Fo(A) = {$}

Questions

→ S → ACD

C → a | b = a, b

Fo(A) = first(C) = {a, b}

Fo(D) = Follow(S) = {$}

→ S → aSbS | bSaS | ∈ = $, b, a

Follow never contains ∈

→ S → AaAb | BbBa

A → ∈

B → ∈

Fo(A) = a, b

Fo(B) = b, a

→ S → ABC

A → DEF

B → ε

C → ε

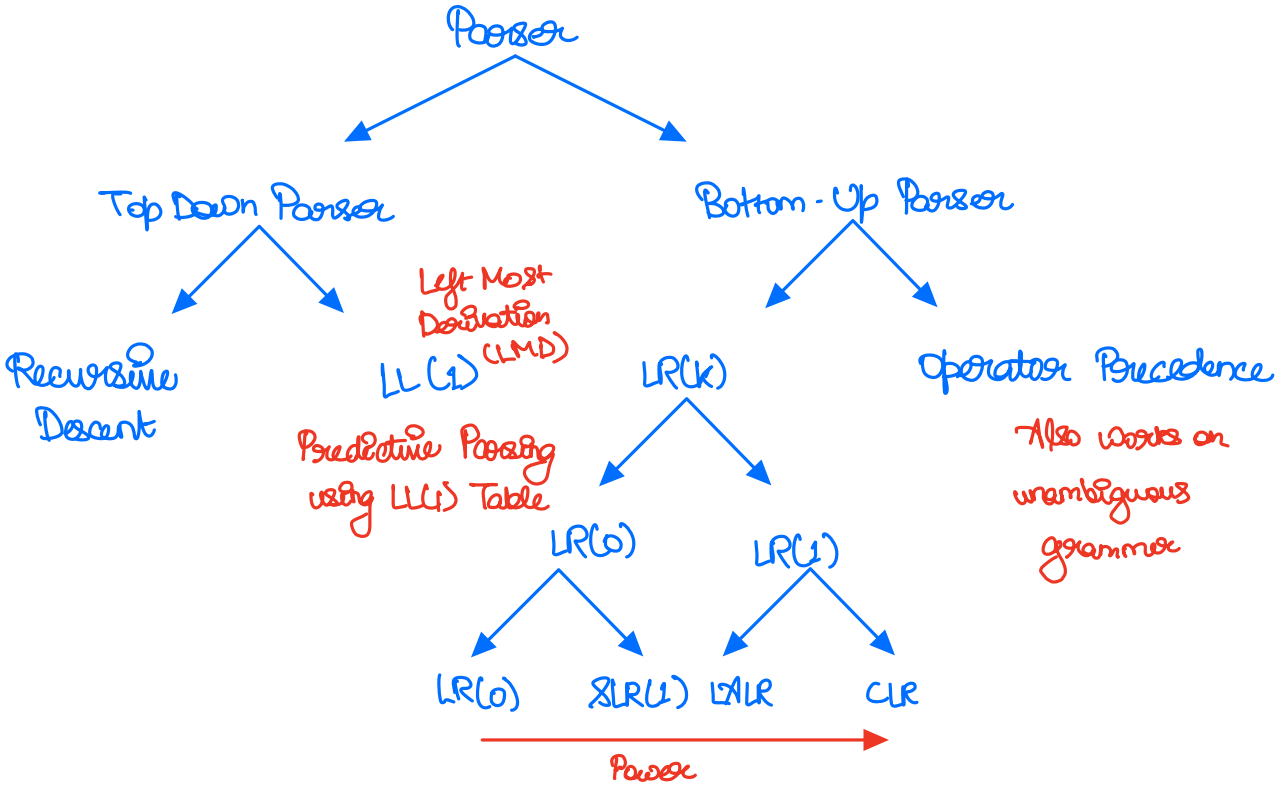
D → ε

E → ε

F → ε

∴ $

Parsing



Parsing is the process of deriving string from a given grammar.

It uses context-free grammar.

LL Grammar Parsing Table

	First	Follow
$S \to (L) \mid a$	$\{(, a\}$	$\{\$, ,,)\}$
$L \to SL'$	$\{(, a\}$	$\{)\}$
$L \to \epsilon \mid , SL'$	$\{\epsilon, ,\}$	$\{)\}$

	(	)	a	,	$
S	1		2		
S'	3		3		
L		4		5	

$S \to (L)$

$S \to a$

Since each box has 1 production, it is LL(1) grammar

$S \to aSbS \mid bSaS \mid \epsilon$

	First	Follow
	$\{a, b, \epsilon\}$	$\{a, b, \$\}$

	a	b	$
S	1,3	2,3	3

It is not an LL(1) grammar.

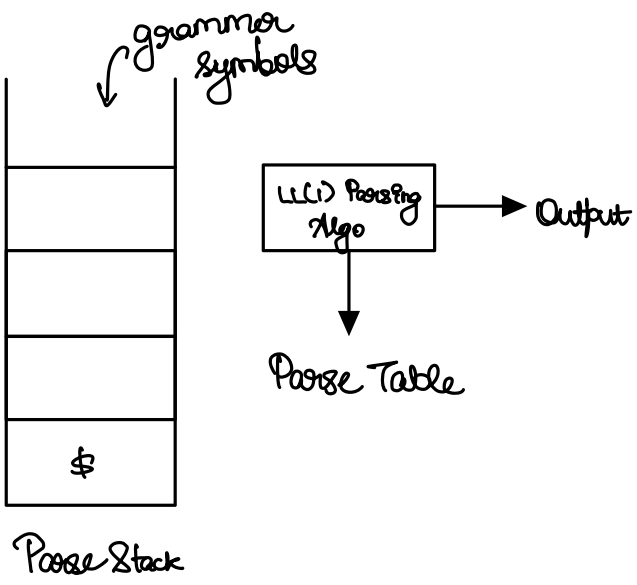
LL(1) Parser

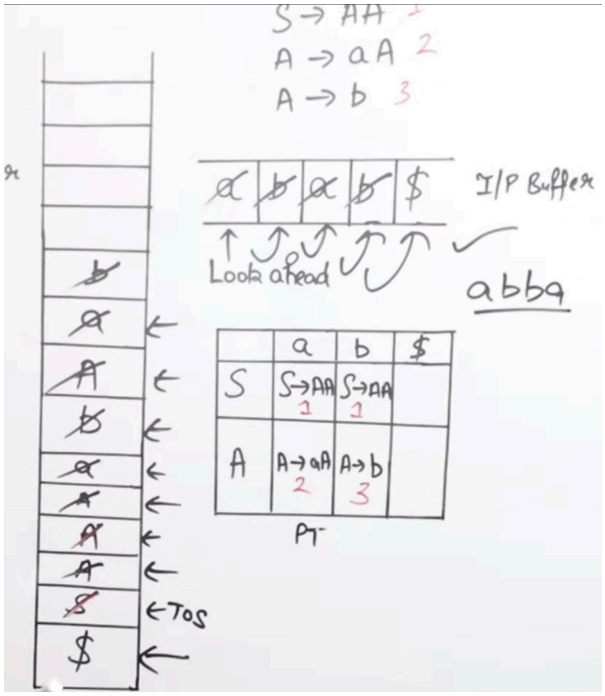
LL(1) parser is a top down parser

L : Left Most Derivation

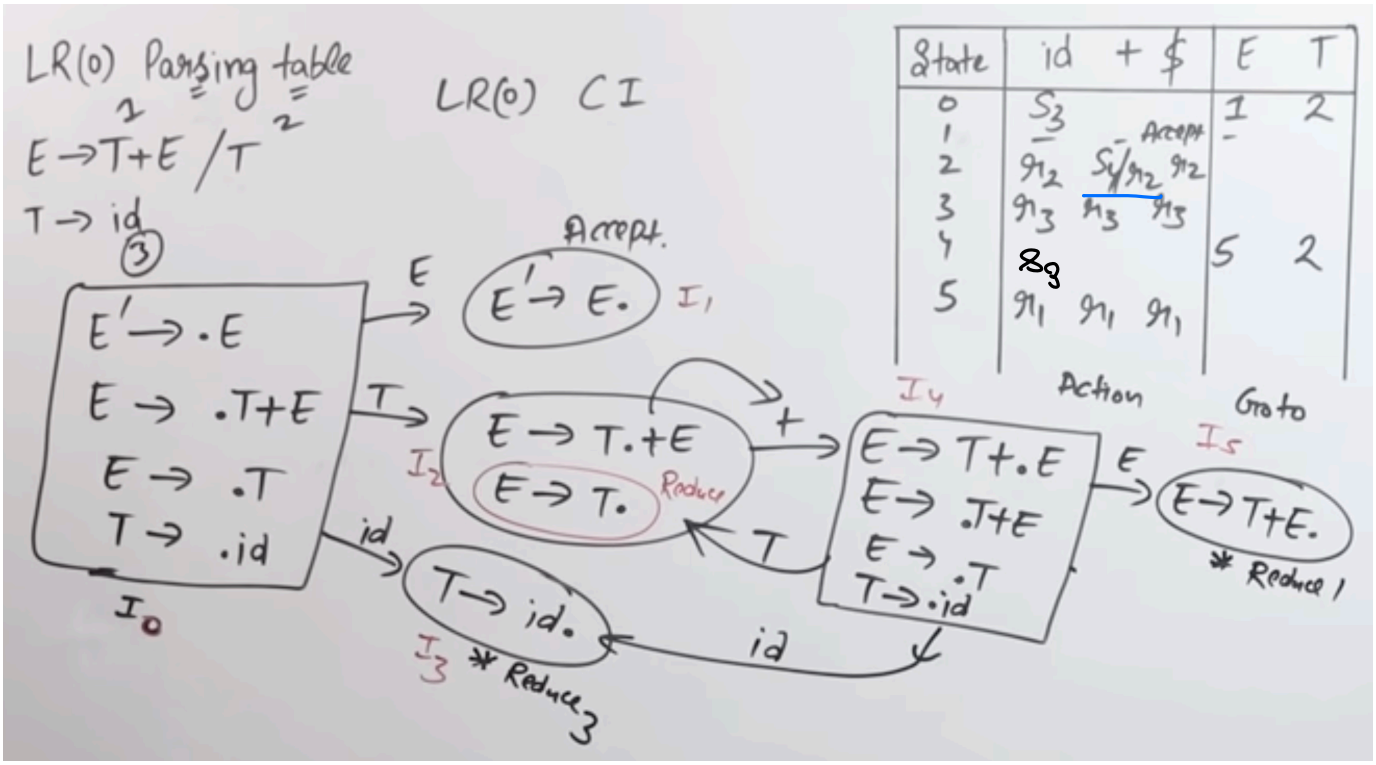
L : Left to right scanning

(1) : look ahead symbol (by 1)





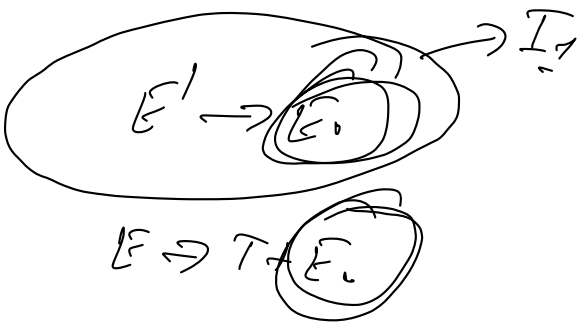
LR(0) Parsing



State	id	+	$	E	T
0	S_3			1	2
1			Accept		
2	r_2	S_4/r_2	r_2		
3	r_3	r_3	r_3		
4	S_3			5	2
5	r_1	r_1	r_1		

Shift, reduce in same column, i.e not LR(0) grammar.

State	Action id	+	$	Goto E	T
0	S_3			1	2
1	Accept				
2	r_2	S_4/r_2	r_2		
3	r_3	r_3	r_3		
4	S_3			5	2
5	r_1	r_1	r_1		



SLR Grammar

	Follow
$E \to T+E \mid T$	$
$T \to id$	+, $

State	Action		
	id	+	$
0	s_3		
1			Accept
2		s_4	r_2
3		r_3	r_3
4	s_3		
5			r_1

No s/r conflict, therefore acceptable by SLR grammar