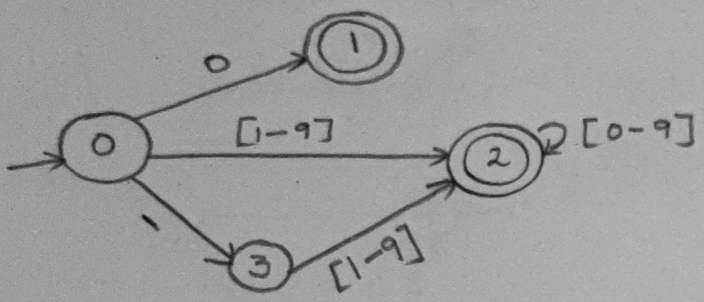
Assignment-1 (PART-3)

LEXING

Regular Expression for integers:

`0|[-]?[1-9][0-9]*`

Required finite automata for Regex of integers:

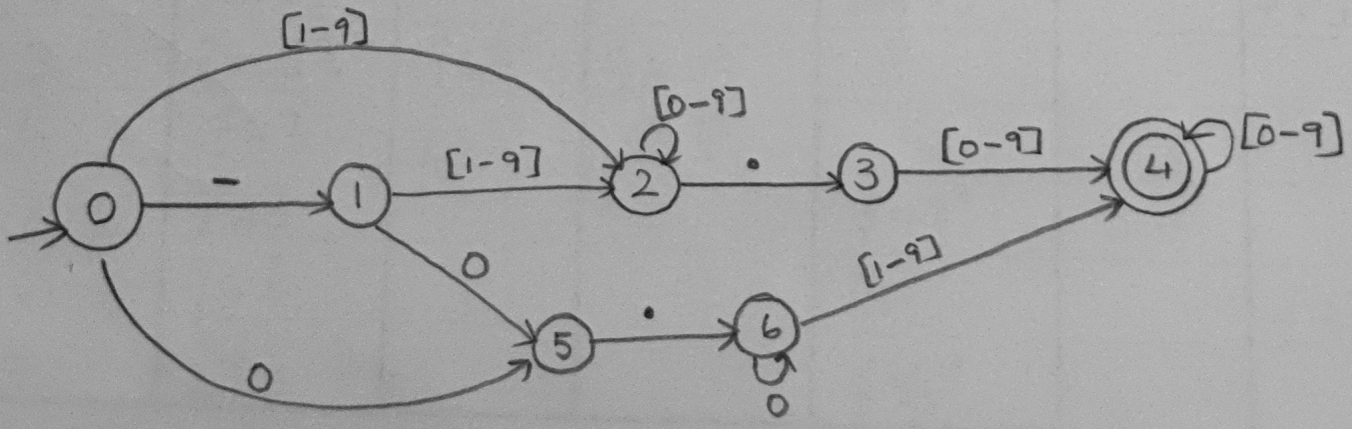


(where states 1 & 2 are final states)

Regular expression for floating numbers:

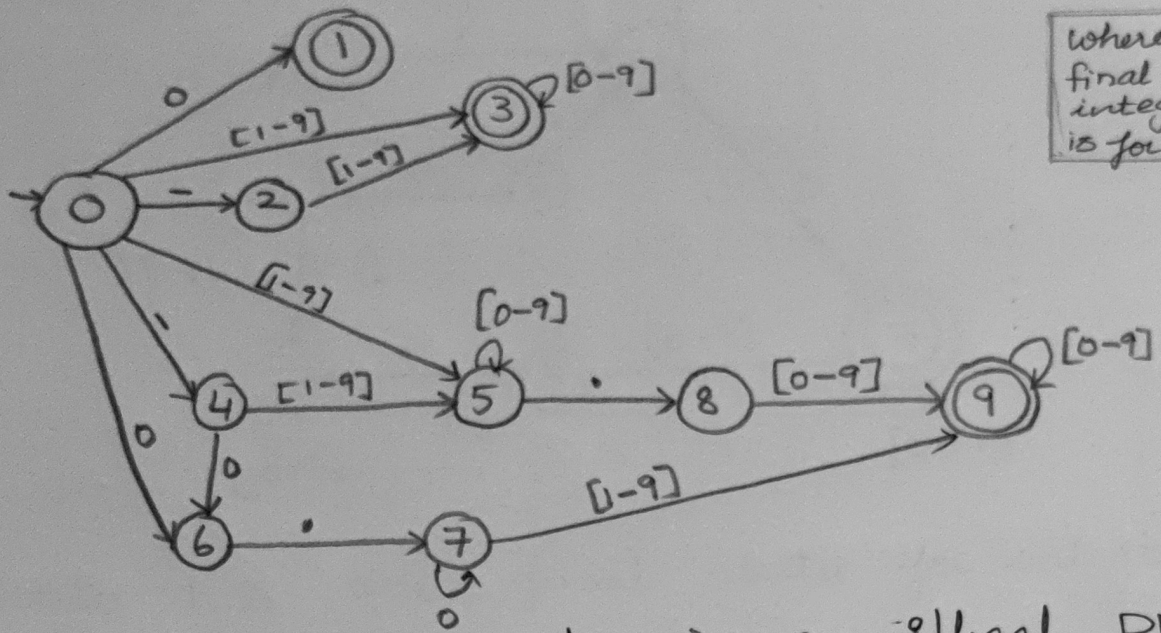
`0|[-]?[1-9][0-9]*.[0-9]+|[-]?[0].[0]*[1-9][0-9]*`

Required finite automata for Regex of floating numbers:



(where 4 is the final state)

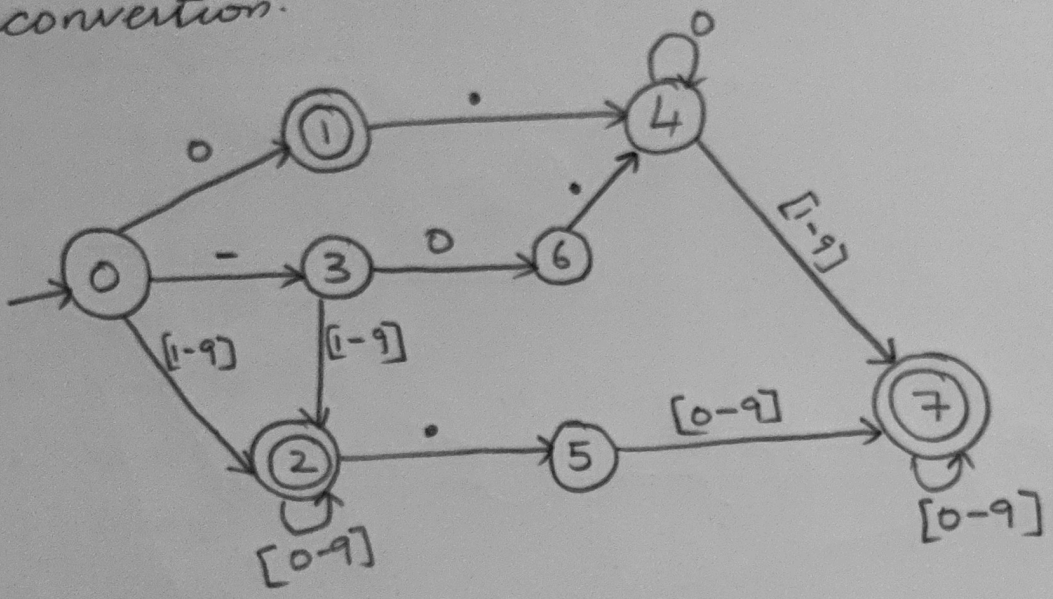
Merging the finite automata of integers and floating numbers.



The obtained automata is an illegal DFA (NFA). Using Power Construction Set algorithm to convert it to DFA.

DFA	NFA	0	[1-9]	[0-9]	-	.
0	{0}	{1, 6}^{1}	{3, 5}^{2}		{2, 4}^{3}	
1	{1, 6}					{7}^{4}
2	{3, 5}			{3, 5}^{2}		{8}^{5}
3	{2, 4}	{6}^{6}	{3, 5}^{2}			
4	{7}	{7}^{4}	{9}^{7}			
5	{8}			{9}^{7}		
6	{6}					{7}^{4}
7	{9}			{9}^{7}		

Representation of finite automata after convertion.



States 1, 2 are final states for integers

State 7 is final state for floating numbers.

PARSING

Given grammar

1. $S \rightarrow aTUF$
2. $T \rightarrow c$
3. $T \rightarrow bT$
4. $U \rightarrow VF$
5. $V \rightarrow Vd$
6. $V \rightarrow \varepsilon$
7. $F \rightarrow e$
8. $F \rightarrow \varepsilon$

→ To construct the parse table First and follows of the non-terminals are required. Below table depicts them:

	First	Follows
S	a	$
T	c, b	d, e, ε
U	d, e, ε	f
V	d, ε	e, d, ε
F	e, ε	f

→ The parse table is constructed using the first and follows functions of the non-terminals.

	a	b	c	d	e	f	$
S	S→aTUF						
T		T→bT	T→c				
U				U→VF	U→VF	U→VF	
V				V→Vd V→ε	V→ε		V→ε
F					F→e	F→ε	F→ε

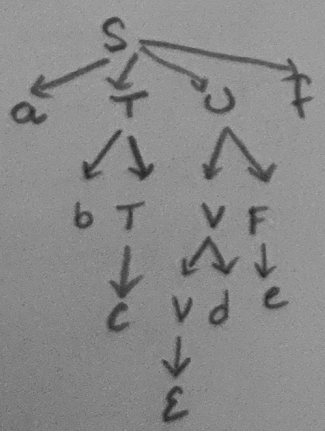
The given grammar is not LL(1) grammar. As the row 'V' with column 'd' has two production rules, which is not acceptable in the case of LL(1) grammar.

To convert the given grammar into LL(1) grammar - the given grammar requires elimination of ambiguity, left recursion and non-determinism.

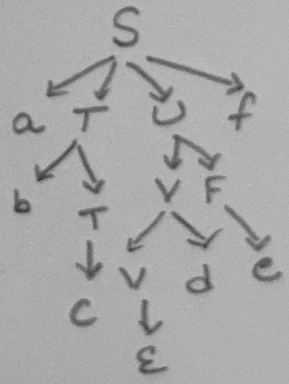
Elimination of ambiguity:

The given grammar doesbot have ambiguity with respect to the string given 'abcdef'.

Left most derivation



Right most derivation.



The parse trees obtained in both derivations is similar. So, it proves that this grammar is non-ambigous.

Elimination of Left Recursion:

For production in the form $A \rightarrow A\alpha / \beta$

$A \rightarrow A'\beta$

$A' \rightarrow \alpha A' / \varepsilon$

So, this is applicable to Rule 5 & Rule 6 combined.

$V \rightarrow Vd / \varepsilon$

This gives rise to:

$V \rightarrow V'$

$V' \rightarrow dV' / \varepsilon$

Elimination of non-determinism:

The given grammar has no issue of non-determinism.

After Elimination of ambiguty, left recursion and non-determinism production rules are

1.) $S \rightarrow aTUf$

2.) $T \rightarrow c / bT$

3.) $U \rightarrow VF$

4.) $F \rightarrow e / \varepsilon$

5.) $V \rightarrow V'$

6.) $V' \rightarrow dV' / \varepsilon$

For construction of parse table, first and follows functions are required and are represented in the table below:

	First	Follows
S	a	$
T	c, b	d, e, ε
U	d, e, ε	f
V	d, ε	e, ε
V'	d, ε	e, ε
F	e, ε	f

The following parser table

	a	b	c	d	e	f	$
S	S→aTUf						
T		T→bT	T→c				
U				U→VF	U→VF	U→VF	
V				V→V'	V→V'		
V'				V'→dV'	V'→ε		V'→ε
F					F→e	F→ε	F→ε

Now, the grammar is LL(1) grammar.

Using the string : abcdef simulating stack-based execution of an LL(1) parser.

Converting the production rules to reduction rules

1.) $S \leftarrow fUTa$
2.) $T \leftarrow c$
3.) $T \leftarrow Tb$
4.) $U \leftarrow FV$
5.) $F \leftarrow e$
6.) $F \leftarrow \varepsilon$
7.) $V \leftarrow V'$
8.) $V' \leftarrow V'd$
9.) $V' \leftarrow \varepsilon$

Stack	Input	Note
$ S	abcdef $	reduce with R_1 $S \leftarrow fUTa$
$ fUTa	abcdef $	consume a
$ fUT	bcdef $	reduce with R_3 $T \leftarrow Tb$
$ fUTb	bcdef $	consume b
$ fUT	cdef $	reduce with R_2 $T \leftarrow c$
$ fUc	cdef $	consume c
$ fU	def $	reduce with R_4 $U \leftarrow FV$
$ fFV	def $	reduce with R_7 $V \leftarrow V'$

Stack	Input	Note
$fFV'	def$	reduce with R_8 $V' \leftarrow Vd$
$fFV'd	def$	Consume d
$fFV'	ef$	reduce with R_9 $V' \leftarrow \varepsilon$
$fF	ef$	reduce with R_5 $F \leftarrow e$
$fe	ef$	consume e
$f	f$	consume f
$	$	Match occured.