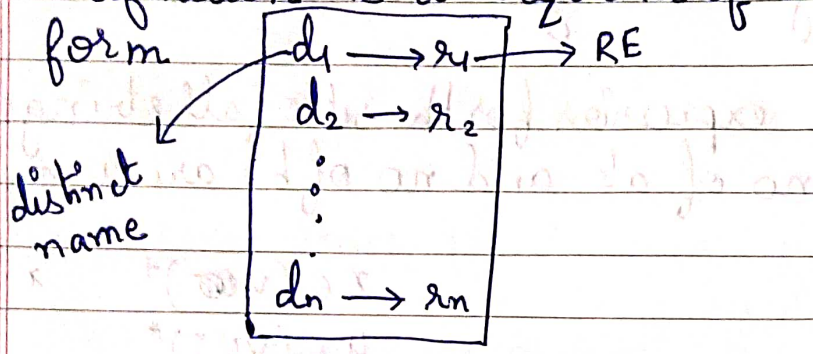
Date __/__/____

18/07/2022

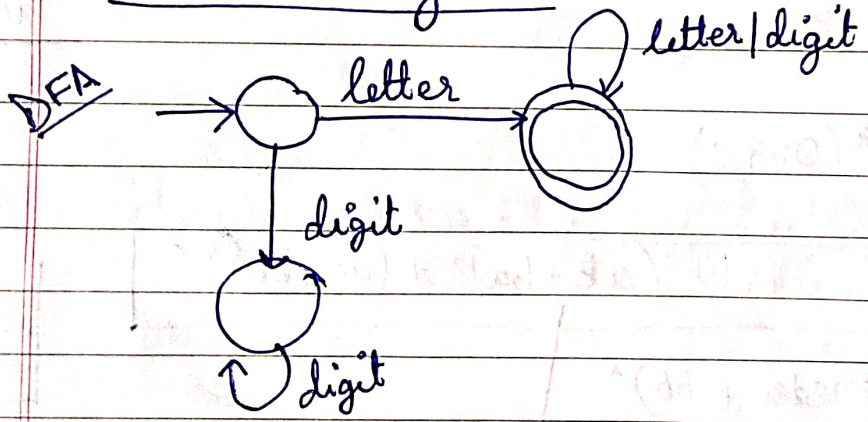
Regular Definitions

If Σ is an input alphabet ~~in sigma is~~ of basic symbols, then a regular definition is a sequence of definitions of the form



1) letter → a|A| . . . z|Z
2) digit → 0|1| . . . |9
3) id → letter (letter | digit)*

Transition diagram



* Conversion of a NFA to DFA

Subset Construction

1. ε - closure (s)
2. ε - closure (t)
3. ~~ε - closure (move)~~ move (T, a)

Saathi

Date ___/___/___

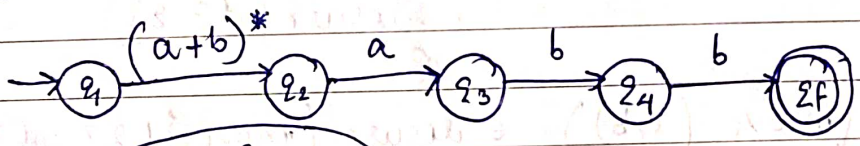
① ε-closure(s) → set of NFA states reachable from NFA state S from ε transition alone.

② ε-closure(t) → set of NFA states reachable from some NFA state 's' in set T on ε-transition.

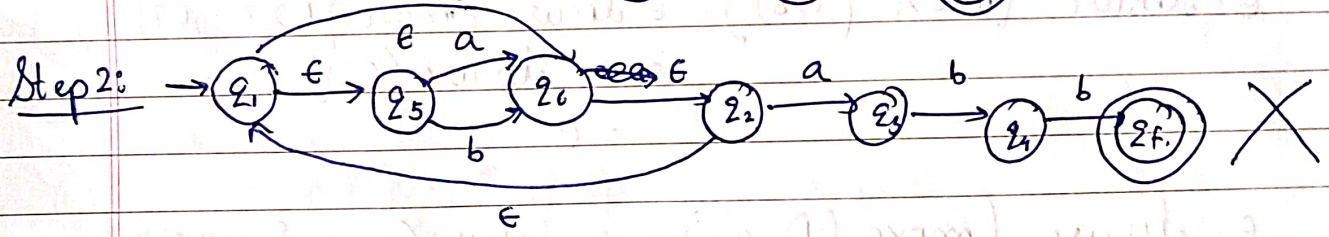
③ move(T,a) → set of NFA states to which there is a transition on input symbol a from some state s in T.

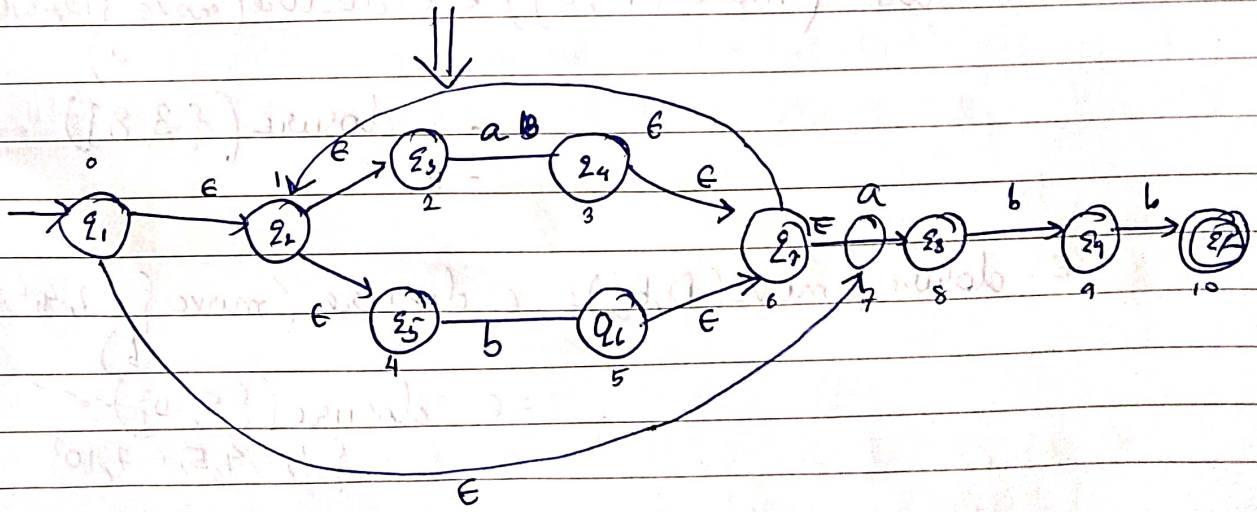
a) $(a+b)^*abb$

Step1:



Step2:





ε-closure(0) = {0, 1, 7, 2, 4} = A

ε-closure(move(A,a)) = ε-closure(move({0,1,2,7,4} a))
= ε-closure({3,8})
= {1,2,3,4,6,7,8} = B

Saathi

Date ___ / ___ / ____

ϵ-closure(move(A, b)) = ϵ-closure(move{0, 1, 2, 4, 7}, b)
= ϵ-closure({5})
= {1, 2, 4, 5, 6, 7} = C

ϵ-closure(move(B, a)) = ϵ-closure(move{1, 3, 3, 4, 6, 7, 8}, a)
= ϵ-closure({3, 8})
= B

ϵ-closure(move(B, b)) = ϵ-closure(move{1, 3, 3, 4, 6, 7, 8}, b)
= ϵ-closure({5, 9})
= {1, 2, 4, 5, 6, 7, 9} = D

ϵ-closure(move(C, a)) = ϵ-closure(move{1, 2, 4, 5, 6, 7}, a)
= ϵ-closure({3, 8})
= B

ϵ-closure(move(C, b)) = ϵ-closure(move{1, 2, 4, 5, 6, 7}, b)
= ϵ-closure({5})
= C

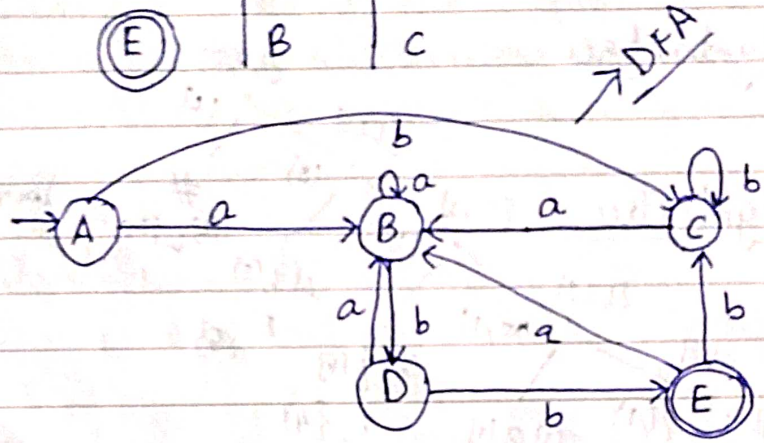
ϵ-closure(move(D, a)) = ϵ-closure(move{1, 2, 4, 5, 6, 7, 9}, a)
= ϵ-closure({3, 8})
= B

ϵ-closure(move(D, b)) = ϵ-closure(move{1, 2, 4, 5, 6, 7, 9}, b)
= ϵ-closure({5, 10})
= {1, 2, 4, 5, 6, 7, 10}
= E

ϵ-closure(move(E, a)) = ϵ-closure(move{1, 2, 4, 5, 6, 7, 9, 10}, a)
= ϵ-closure({3, 8})
= B

ϵ-closure(move(E, b)) = ϵ-closure(move{1, 2, 4, 5, 6, 7, 10}, b)
= ϵ-closure({5}) = C

State Input	a	b
→ A	B	C
B	B	D
C	B	C
D	B	E
(E)	B	C



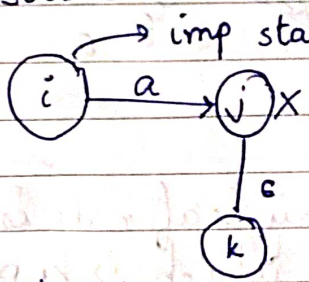
25/07/2022

Regular Expression to DFA

Direct Method

* **Step1:** Create an augmented RE

Important States: Those states with no ϵ- transition are important state.

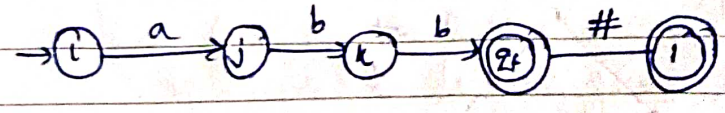


$\exists \forall$

$(\Omega)\#$ → Endmarker to denote the accepting/imp state

Augmented RE

Eg:



* **Step2:** Syntactic Step

Saathi

Date ___ / ___ / _____

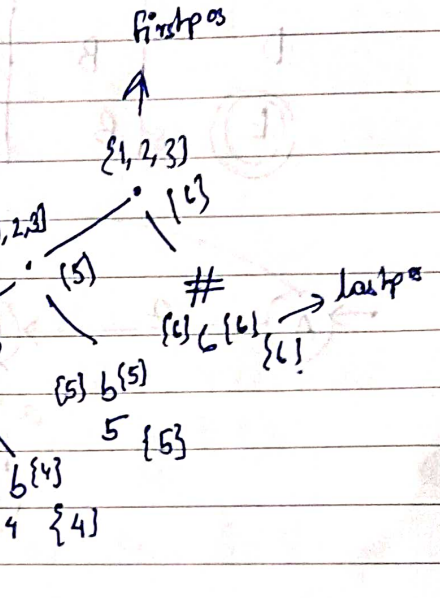
In this step syntax tree is created

(a+b)* abb #

Three types of node:

1. Cat-node (•)
2. Or-node (|)
3. Star-node (*)

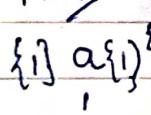
Parse / Syntax tree



Step 4

Four Function of Syntax Tree

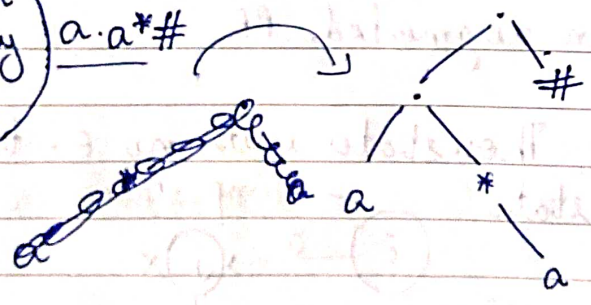
1. nullable(n)
2. firstpos(n)
3. lastpos(n)
4. followpos(p)



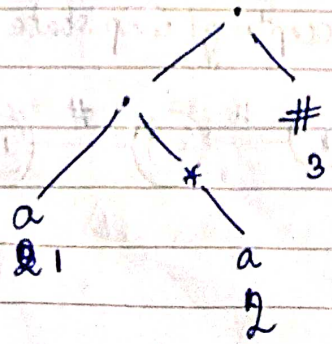
Expression: $(a|b)^*.a.b.b.\#$

position for only leaves

$a.a^*\#$



* Step 3: Every leaf node is labelled with unique number which is known as its position



Saathi

Date

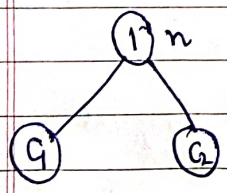
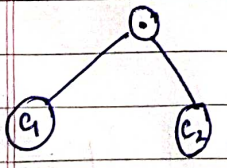
* Step 4:

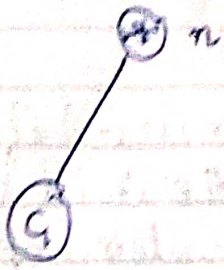
* Nullable(n) is true for a syntax tree node (n) iff the sub-expression represented by n # ε in its language. That is subexpression can be made null or empty string.

* Firstpos(n) is the set of positions in the subtree rooted at n that corresponds to the first symbol of atleast one string in the language of the sub expression rooted at n. (1, 2, 3)

* Lastpos(n) is the set of positions in the subtree rooted at n that corresponds to the last symbol of atleast " " at n.

Rules for computing nullable firstpos and lastpos.

Node n	nullable(n)	firstpos(n)	lastpos(n)
A leaf labeled ε	true	ϕ	ϕ
A leaf with position i	false	{i}	{i}
	nullable(c_1) or nullable(c_2)	firstpos(c_1) ∪ firstpos(c_2)	lastpos(c_1) ∪ lastpos(c_2)
	nullable(c_1) and nullable(c_2)	if (nullable(c_1)) firstpos(c_1) ∪ firstpos(c_2) else firstpos(c_1)	if (nullable(c_2)) lastpos(c_1) ∪ lastpos(c_2) else lastpos(c_2)

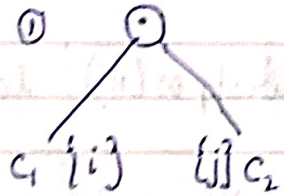
	true	firstpos(c_1)	lastpos(c_1)

followpos(i)

Rules:

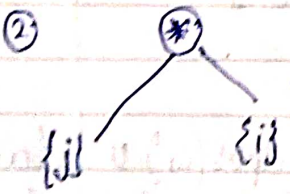
① If n is a cat-node with left child c_1 and right child c_2 and i is a position in lastpos(c_1) then all position in firstpos(c_2) are in followpos(i).

② If n is a star-node and i is a position in lastpos(n), then all positions in firstpos(n) are in followpos(i).



⇒ followpos(i) = {j} → firstpos(n)

② If n is a star node



⇒ follow(i) = {j}

i	followpos(i)
1	{1,2,3}
2	{1,2,3}
3	{4}
4	{5}
5	{6}
6	ϕ

Date

Last pos(*) = {1, 2}

followpos(1) = {1, 2}
followpos(2) = {1, 2}

Cat
followpos(1) = {3}
followpos(2) = {3}

= {1, 2, 3}

firstpos(root) = {1, 2, 3} = A → Initial state

a → followpos(1) ∪ followpos(3)
= {1, 2, 3, 4} = B

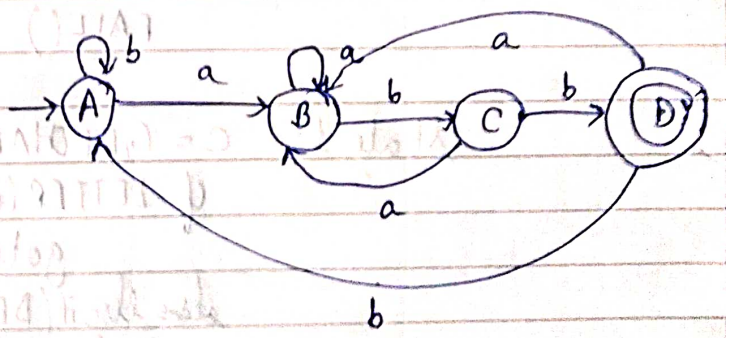
b → followpos(2) = {1, 2, 3}

a → B
b → {1, 2, 3, 5} = C

B C
a → {B}
b → f(2) ∪ f(5)
= {1, 2, 3, 6} = D

D
a →
b → {1, 2, 3} = A

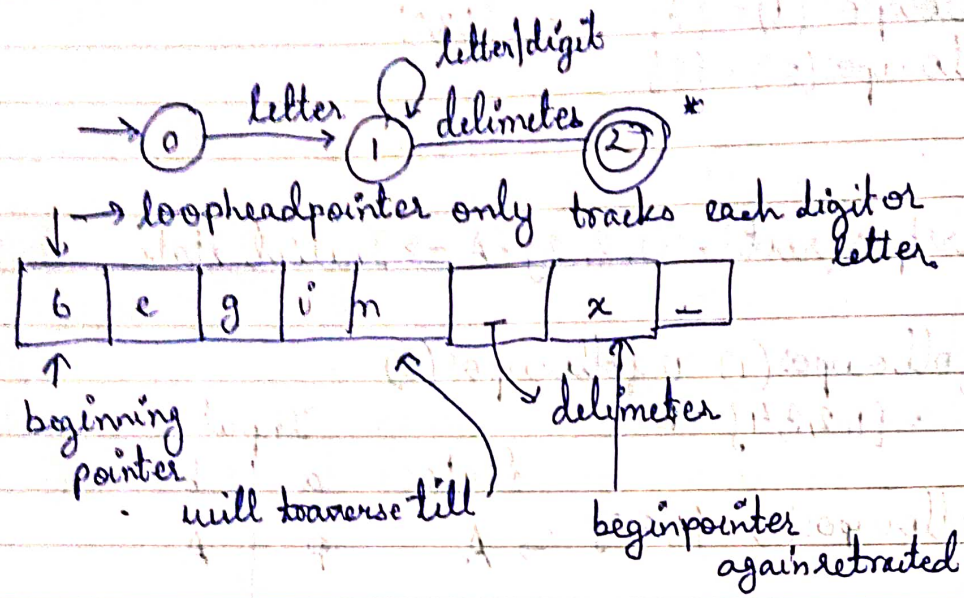
State	Input a	Input b
→ A	B	A
B	B	C
C	B	D
(D)	B	A



Design of a lexical analyser

1) Token → identifier

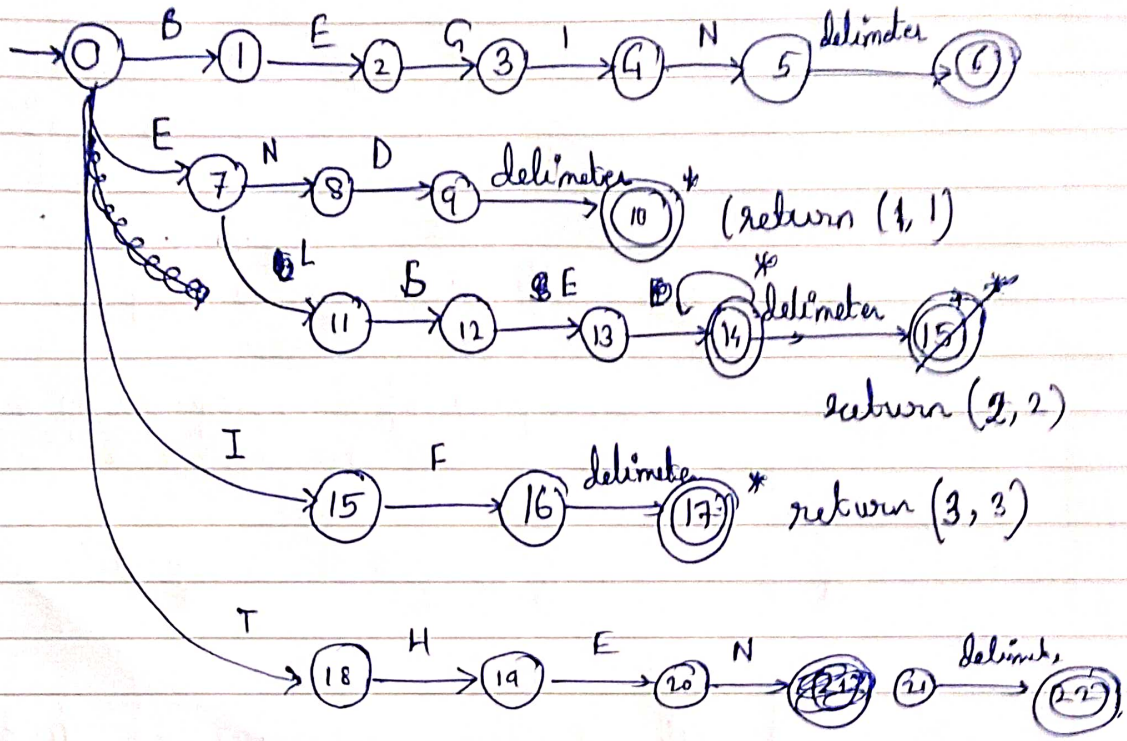
delimeter or blank or newline



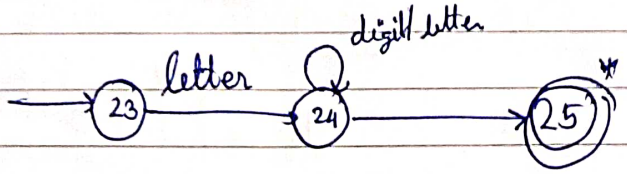

```
State 0:  c = GETCHAR()
          if LETTER(c) then
             goto State 1
          else
             FAIL()

State 1:  c = GETCHAR()
          if LETTER(c) or DIGIT(c) then
                goto State 1
          else if(DELIMETER (c))
          goto State 2
          else
             FAIL()

State 2:  RETRACT()
          return (id. installed)
```

Token → Keyword:

begin, if, end, else, then



If THENA



$(a+b+c)^*$

aabbcc