Theary of Computation

Date

Introduction

The purpose of theory of computation is to develope formal mathematical models of computation that reflect real-world computers.

Theory of complexity can be divided into three areas

- Complexity theory
- Computability theory
- Automata theory

- Complexity theory : classify problems according to their degree of "difficulty".
- Computability theory : classify problems as being solvable or unsolvable by a computer.
- Automata theory : deals with definitions and properties of different computational models. Also asks whether all models have same power or can some solve more problems than others.

Spiral

Date

Pigeon-hole principle

If (n+1) or more objects are placed in n boxes, then atleast one box has more than one object.

In other words, if A & B are sets such that $|A| > |B|$, then there is no one-one function from A to B

Finite Automata and Regular Languages

An example: a toll gate

Suppose we want a computer to control a toll gate.

The gate is opened when ₹ 25 is paid. We accept ₹5, 10 & 25 coins.

The computer will not return change.

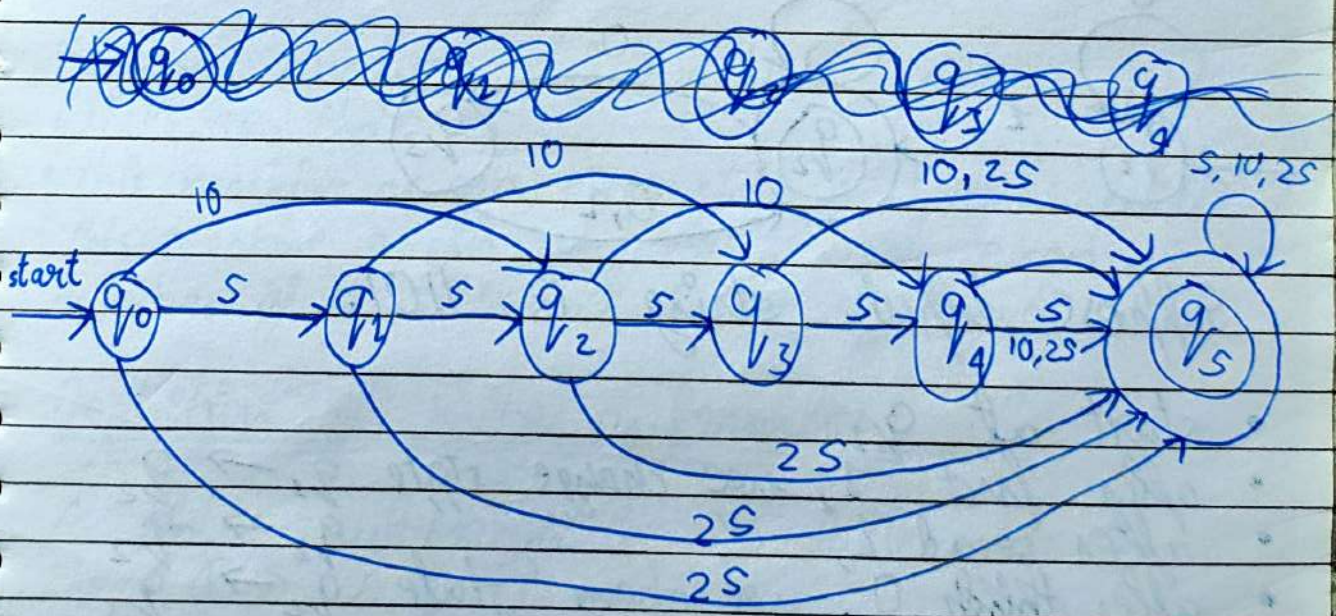
In order to decide when to open gate, machine has to be in one any of 6 states.

Date

- machine is in state q_0, if it has collected no money.
- machine is in state q_1, if it has collected ₹5
- q_2 state when ₹10 collected
- q_3 state when ₹15 collected
- q_4 " " ₹20 "
- q_5 " " ₹25 collected or more.

Initially, state is q_0.

The states and transitions can be shown as follows.



The numbers above line show what coin was inserted

q_5 is shown with double circles because it is a special state. When it reaches state q_5 it accepts payments and opens gate.

Spiral

This state is called accept state.

Date

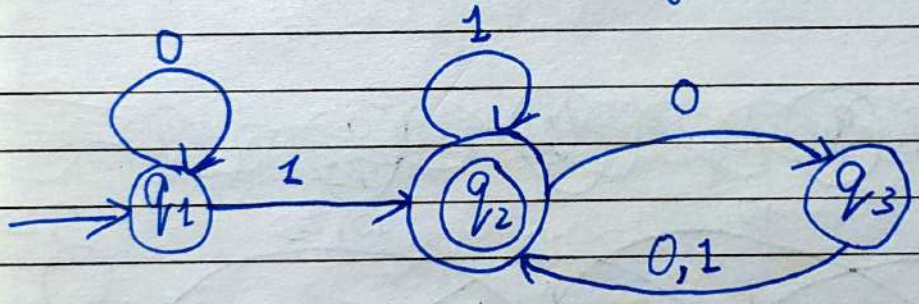
This machine only needs to remember which state it is in at any given time.

So to distinguish all 6 states, it only needs

$\lceil \log_2 6 \rceil = 3$ bits

Deterministic finite automata

Consider the state diagram



Suppose input string is 1101.

- start at q_2
- after first 1, we change state $q_1 \rightarrow q_2$
- after second 1, " " $q_2 \rightarrow q_2$
- after 0, we change state $q_2 \rightarrow q_3$
- finally after 1, we change $q_3 \rightarrow q_2$

After processing 1101, we are at q_2 which is accept state.

Date

Therefore, we say string 1101 is accepted by the machine.

Suppose string 1010

- first 1 : $q_1 \rightarrow q_2$
- second 0 : $q_2 \rightarrow q_3$
- third 1 : $q_3 \rightarrow q_2$
- fourth 0 : $q_2 \rightarrow q_3$

After processing 1010, we do not end on an accept state.
We say that machine rejects string 1010.

If we analyze this machine, we will see that
- this machine accepts every string that ends with 1
- this machine accepts every string with even number of 0's at right end.

Definition of finite automaton

A finite automaton is 5-tuple collection $M = (Q, \Sigma, \delta, q, F)$

1. Q is a finite set of states
2. Σ is a finite set of symbols. This set is called an alphabet.
3. $\delta : Q \times \Sigma \rightarrow Q$ is a function called transition function.

Spiral

Date

4. q is an element of Q; it is the start state.
5. F is a ~~subset~~ subset of Q; it is the set of accept states.

δ can be thought of as the program of the finite automaton. This function tells M what to do in "one step".

In our toll gate example.
$Q = \{q_0, q_1, q_2, q_3, q_4, q_5\}$
$\Sigma = \{5, 10, 25\}$
$q = q_0$
$F = \{q_5\}$

δ is represented by a table for inputs and outputs

	5	10	25
q_0	q_1	q_2	q_5
q_1	q_2	q_3	q_5
q_2	q_3	q_4	q_5
q_3	q_4	q_5	q_5
q_4	q_5	q_5	q_5
q_5	q_5	q_5	q_5

← symbol input

↑ state input

Spiral

Date

Transition

As we can see transition function δ has two inputs, one for current state (r) and one for a symbol (a).

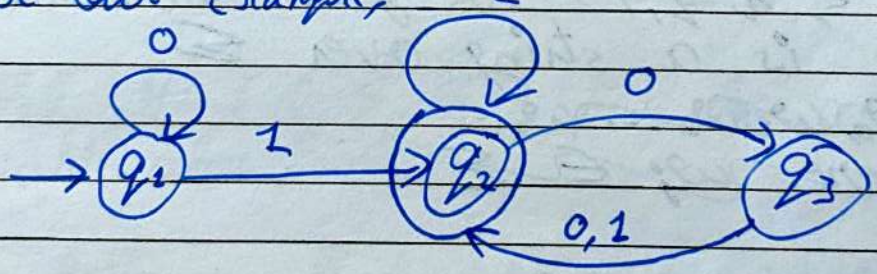
So we write transition function as

$\delta(r, a)$

r ← current state

a ← symbol

For our example,



δ is:

	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_1	q_2

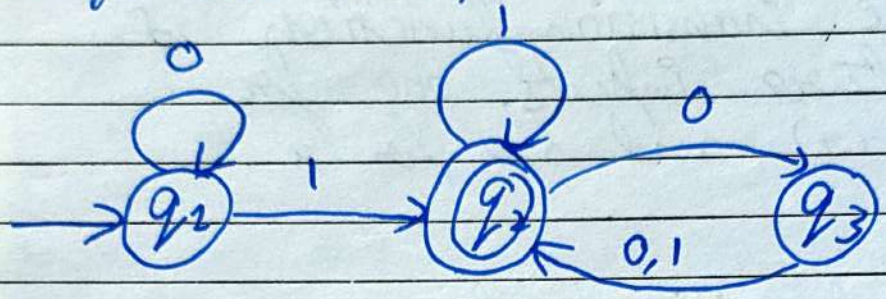
Language of a finite automaton

The language of $M = (Q, \Sigma, \delta, q, F)$ is set of all string accepted by M. It is written as $L(M)$

Spiral

Date

So for our example machine



$L(M) = \{w :$ w contains atleast one 1 and ends with even no. of 0's $\}$

Let $M = (Q, \Sigma, \delta, q, F)$ be finite automata and w is a string over Σ

i.e $w = w_0 w_1 w_2 \ldots w_n$
where $w_i \in \Sigma$

Then we define sequence of states $r_0, r_1, r_2 \ldots r_n$ such that

* $r_0 = q$
* $r_{i+1} = \delta(r_i, w_{i+1})$, for $i = 0, 1, \ldots, (n-1)$

If $r_n \in F$, then we say M accepts w
If $r_n \notin F$, then we say M rejects w.

Date

w can be an emtty string. Empty string is denoted by epsilon (ε)

In this case, sequence $r_0, r_1, \ldots r_n$ has length 1.
So it just has $r_0 = q$.

M accepts emtty string if $q \in F$.

A language A is called regular if there is a finite automata M such that $A = \mathcal{L}(M)$

Let $M = (Q, \Sigma, \delta, q, F)$ be a finite automaton.

δ tells us that when M is in state $r \in Q$ and reads a symbol $a \in \Sigma$, it switches from r to $\delta(r, a)$.

Let Σ^* be set of all possible strings over Σ (including empty string)

Then

$$\bar{\delta} : Q \times \Sigma^* \to Q$$

This function is defined as

Date

~~$\delta(r,w) = \begin{cases} r \\ (\delta(\bar{\delta}(r,v), a) \end{cases}$~~

$$\bar{\delta}(r,w) = \begin{cases} r \\ (\delta(\bar{\delta}(r,v), a); & w = va \text{ and } 'a' \text{ is a symbol} \end{cases}$$

Here, r is current state and w is a string

$\bar{\delta}(r,w)$ is the state which M reaches when it starts at state r and reads string w from left to right, using δ to switch states.

Therefore,

$L(M) = \{w : w \text{ is a string over } \Sigma \text{ and } \bar{\delta}(q,w) \in F\}$

Examples of finite automaton

1. $A = \{w : w \text{ contains an odd number of 1's}\}$

Prove that A is a regular language.

$\Sigma = \{0,1\}$

Date

We can read w from left to right and change state everytime we read a 1 in w. Since, we need odd no. of 1's every other state is accepted.

But this will lead to infinite states but finite automata can only have finite states.

Since there is repeating pattern, we can have two states.

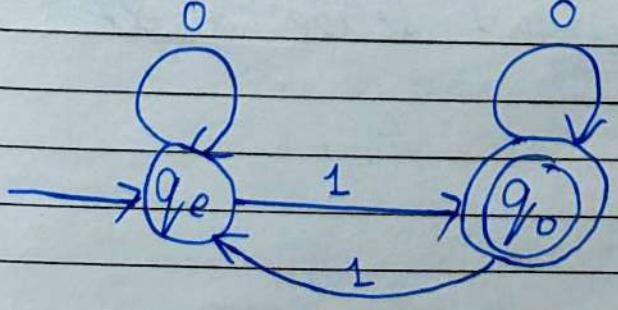
q_o: odd no. of 1's
q_e: even no. of 1's

Zero 1's is even, therefore initial state is q_e

δ is

	0	1
q_o	q_o	q_e
q_e	q_e	q_o

Finite automata is



Spiral

Date

2. $A = \{w : w$ is binary sting containing 101 substring$\}$

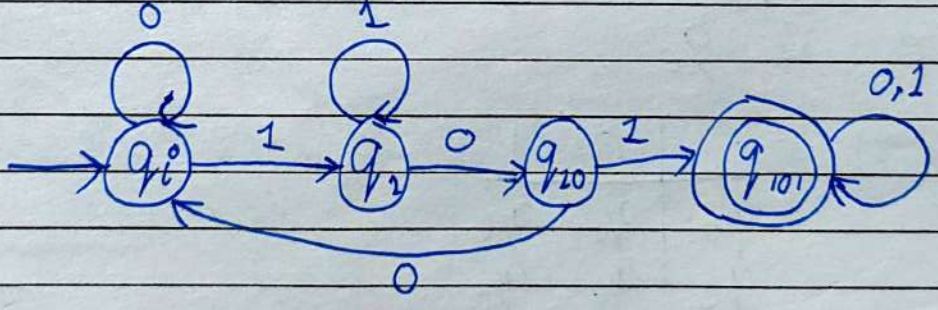
Prove that A is a regular language.

To test substring, we need to store bits as we read them in memory.

Since substring to test has three bits, we can use three states to store current substring (q_1, q_{10}, q_{101})

We also need a initial state in which we know that there is no potential substring (q_i).

The machine will be.



Once we read 101 consecutive in a string, we will enter accept state after which we won't leave accept state.

$\delta =$

	0	1
q_i	q_i	q_1
q_1	q_{10}	q_1
q_{10}	q_i	q_{101}
q_{101}	q_{101}	q_{101}

Spiral

Date

3. $A = \{w : w$ has 1 in third position from right $\}$

Show that A is a regular language.

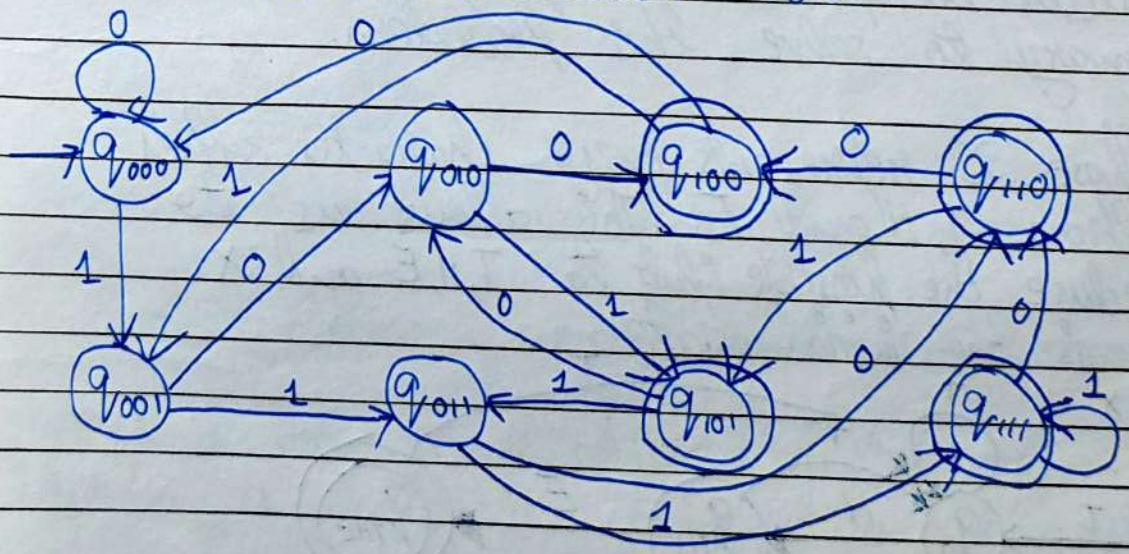
To make machine for this, we will need memory to store the last three bits that the machine has read.

But unlike last example, we can't predict when we have reached final accept state, so we need larger memory.

For three bits we will have states

$q_{000}, q_{001}, q_{010}, q_{011}, q_{100}, q_{101}, q_{110}, q_{111}$

Since we only worry about 1 bit, we can consider initial state 000



We consider initial state q_{000} to avoid errors in case string length is less than 3.

By same logic, initial state will be q_{111} if we want M for 0 in third position from right.

Spiral

Date

δ =

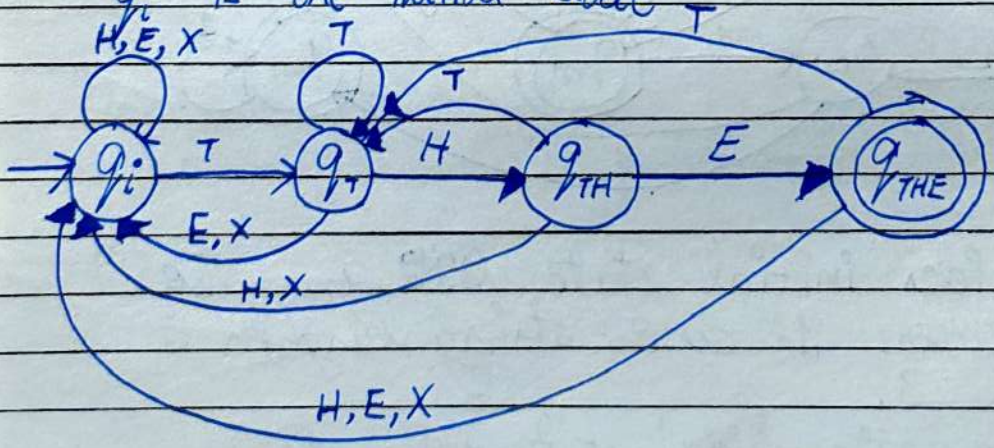
	0	1
q_{000}	q_{000}	q_{001}
q_{001}	q_{010}	q_{011}
q_{010}	q_{100}	q_{101}
q_{011}	q_{110}	q_{111}
q_{100}	q_{000}	q_{001}
q_{101}	q_{010}	q_{011}
q_{110}	q_{100}	q_{101}
q_{111}	q_{110}	q_{111}

4. A = {w : w does ends with a THE}

Σ = {A, B, C ... Z}

Similar to previous example, we need memory to solve this problem.

Suppose X represents every character other than T, H and E. This allows us to reduce the possible input to T, H, E and X.
q_0 is the initial state



Spiral

Date

$\delta =$	T	H	E	X
q_i	q_T	q_i	q_i	q_i
q_T	q_T	q_{TH}	q_i	q_i
q_{TH}	q_T	q_i	q_{THE}	q_i
q_{THE}	q_T	q_i	q_i	q_i

Regular operations

We define three operations on languages.

1. Union

The union of language A and B is

$A \cup B = \{w : w \in A \text{ or } w \in B\}$

2. Concatenation

The concatenation of two languages A and B is

$AB = \{ww' : w \in A \text{ and } w' \in B\}$

AB is a set of all strings obtained by taking a arbitrary string from A and concatnating a string from set B.

$A = \{01, 10\}$
$B = \{00, 11\}$

Then we can find AB as

Date

AB =

A \ B	00	11
01	0100	1101
10	1000	1110

i.e. $AB = \{0100, 1101, 1000, 1110\}$

Note that $w \in A$ is always left and $w' \in B$ is always at right.
This operation is like cross product

3. Star

The star of language A (A^*) is defined as

$A^* = \{u_1 u_2 \ldots u_k : k \geq 0$ and $u_i \in A$ for all $i = 1, 2 \ldots k\}$

A^* is obtained by taking k strings from set A and concatnating them.

A single string can be chosen multiple times and k can be anywhere from 0 to ∞.
Therefore A^* gives an infinite set.

Infact,

$A = \{0, 1\}$

Then
~~$A^* = \{0, 1, \ldots\}$ is a bit string~~
A^* is a set of all binary strings including ε (when $k = 0$).

Spiral

Date

The only time A^* is not infinite is when $A = \{\}$ in which case $A^* = \{\varepsilon\}$

ε is empty string

$\varepsilon \in A^*$ for every A^*.

Another definition of A^* can be through concatnation.

For a language A

$A^0 = \{\varepsilon\}$
$A^1 = A$
$A^2 = AA$
$A^3 = AA^2$
$A^k = AA^{k-1}$

So A^k is A concatnated with itself k times then

$$A^* = \bigcup_{k=0}^{\infty} A^k$$

$$A^* = A^0 \cup A^1 \cup A^2 \cup \ldots \cup A^\infty$$

Date

Set of regular languages is closed under Union

i.e If A is a regular language and B is a regular language.

Then $A \cup B$ is a regular language.

Suppose,

$A = \mathcal{L}(M_1)$; $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$
$B = \mathcal{L}(M_2)$; $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$

Then
$A \cup B = \mathcal{L}(M)$; $M = (Q, \Sigma, \delta, q, F)$

To define M, we will need a way to run both machines M_1 and M_2 simultaneously.

For this we will arrange states of M_1 & M_2 in ordered pair using cartesian product.

$Q = Q_1 \times Q_2 = \{(r_1, r_2) : r_1 \in Q_1 \text{ and } r_2 \in Q_2\}$

Then we can define δ as

$\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$

for all $r_1 \in Q_1$, $r_2 \in Q_2$ and $a \in \Sigma$

Spiral

Date

Therefore, for

$A = \mathcal{L}(M_1)$; $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$

$B = \mathcal{L}(M_2)$; $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$

There is machine M such that

$A \cup B = \mathcal{L}(M)$

and

~~$M = (Q_1 \times Q_2, \Sigma, \delta$~~

$$M = \left(Q_1 \times Q_2, \Sigma, (\delta_1(r_1, a), \delta_2(r_2, a)), (q_1, q_2), F_1 \times F_2\right)$$

The concat and * operators are also same. So if A and B are regular languages.

Then, AB is also regular language and A^* is also regular language.

But proving this in a finite automata we studied so far is hard. We will need another type of finite automaton

Finite automata studied so far have deterministic state transitions for each alphabet. They are called Deterministic Finite Automaton (DFA)

Spiral

Date

Non-deterministic Finite Automaton

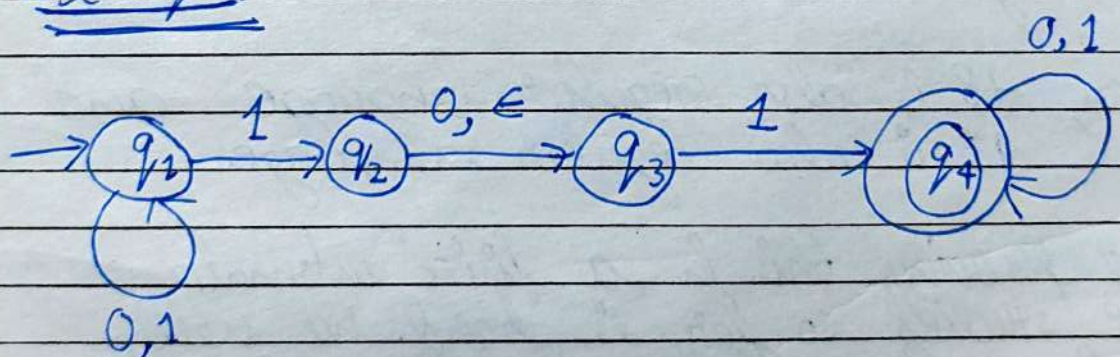
A NFA (Nondeterministic Finite Automaton) has three major additions

i) The machine can transition to any state from a set of states on reading a symbol.

, i.e, same symbol can cause different state transitions

ii) There is an additional transition ϵ, the machine can do this transition without reading a symbol.

Example



- We can see that in state q_1, on reading 1, either machine will stay in 1, or it will transition to q_2
- Machine can transition from q_2 to q_3 without reading a symbol.

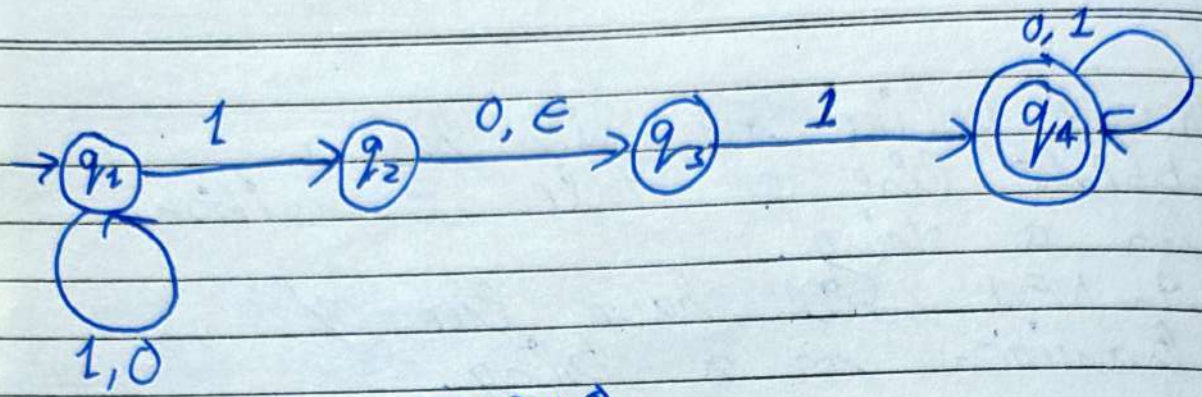
Spiral

Date

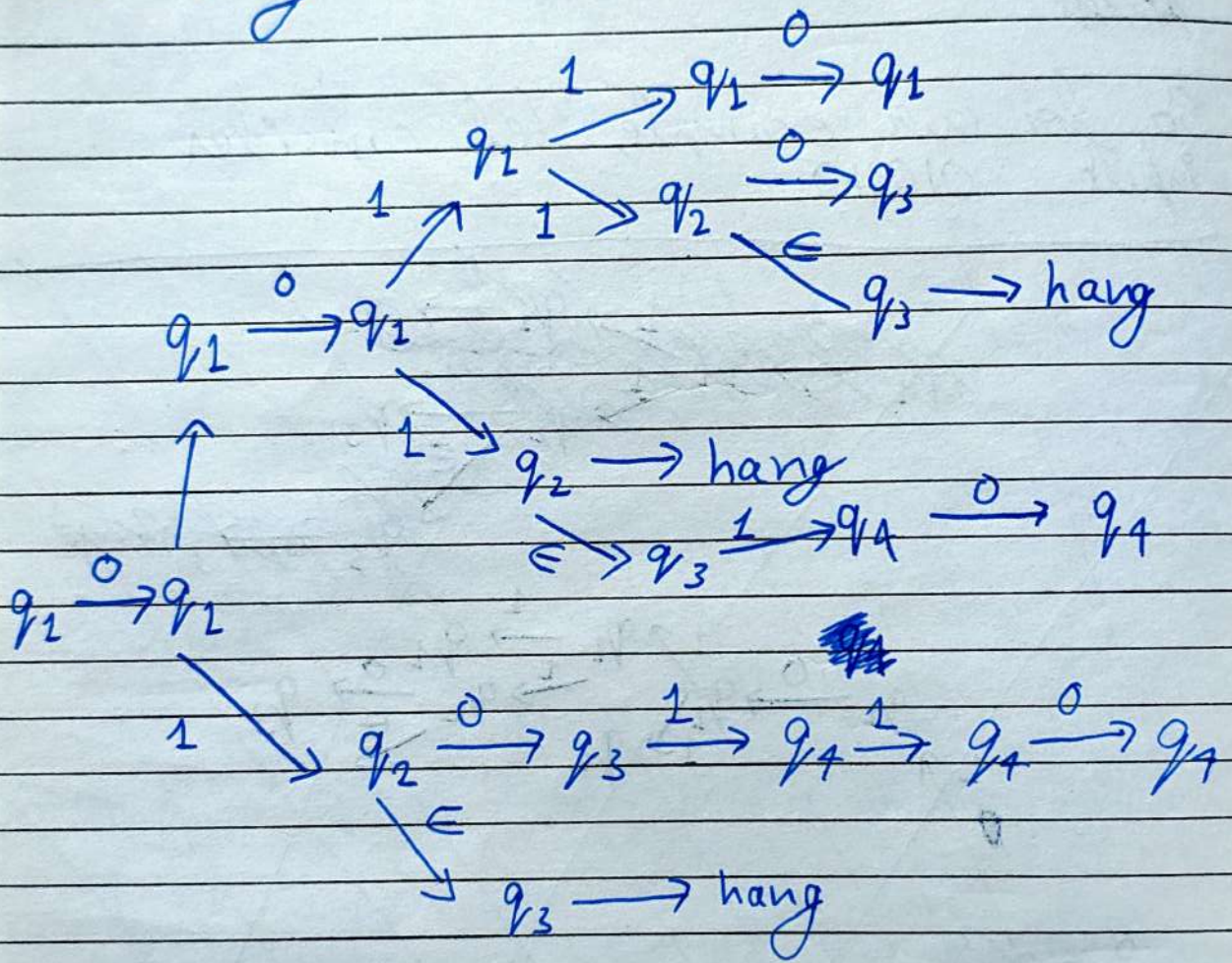
Therefore, unlike DFA which has a straight line of state transitions for a string.
An NFA will have tree of transitions for a string.

So for our example, lets consider input 010110.

Date



For string 010110,



Two branches reach q_4 at end.

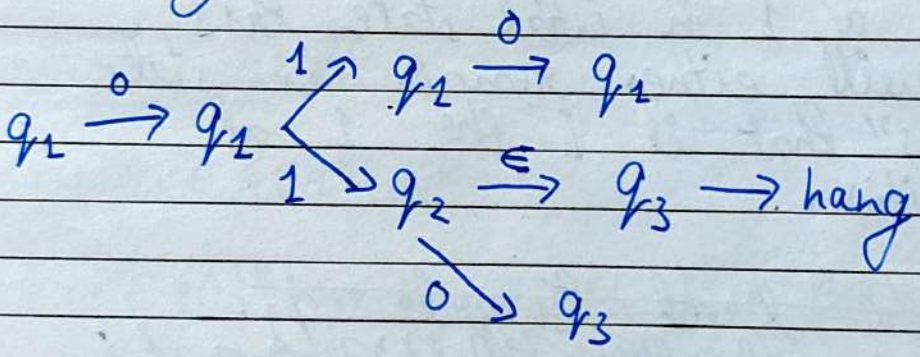
If atleast one branch is ending at an accept state, we say that string is accepted by NFA

Date

We can also see that some branches can't process the whole string because they get stuck on a state. We say that NFA hanged in this cases.

That is, NFA hangs when when no further transitions are possible and symbols are remaining in string.

For string 010,



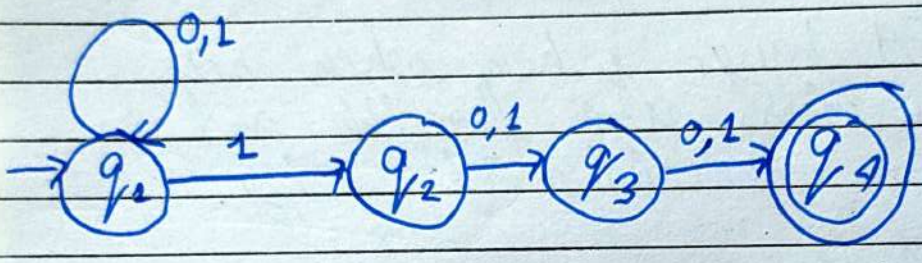
None of the branches end at accept state.
Therefore 010 is rejected by NFA

Date

Second Example,

$A = \{w : w$ has 1 in third position from right$\}$

The machine for this in DFA very complex.
In comparision NFA is very simple.



On reading a 1 in intial state, this machine will either remain in state 1 or "guess" that it is the 1 at third position.

~~It will the travel towards q_4.~~
~~If it was not the 3rd 1 in the string and halts~~

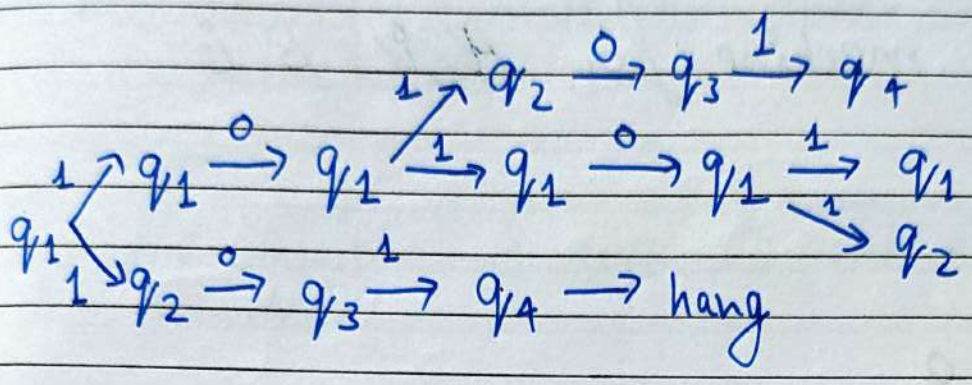
After it guesses that it's third position '1' it will move towards q_4.

- if it was 1 in second or first position from right, string will be rejected as accept was at q_4
- if there are more than three symbols after 1, machine will hang.

Spiral

Date

Example for 10101



A branch reaches q_4, thus 10101 is accepted.

Third Example

~~A= {w | w ∈ {0,1}* and~~

A = {w ∈ {0,1}* : w has multiple of 2 ~~characters~~ symbols or w has multiple of 3 symbols}

This is a language made from union of two other languages

B = {w : w has multiple of 2 symbols}
C = {w : w has multiple of 3 symbols}

A = B ∪ C

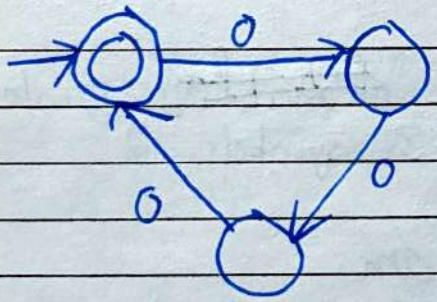
A DFA for both languages B and C is easy, but to form machine for A, we need to run them simultaneously in DFA as seen.

Date

In NFA, designing the machine is much easier.

- The machine for B is

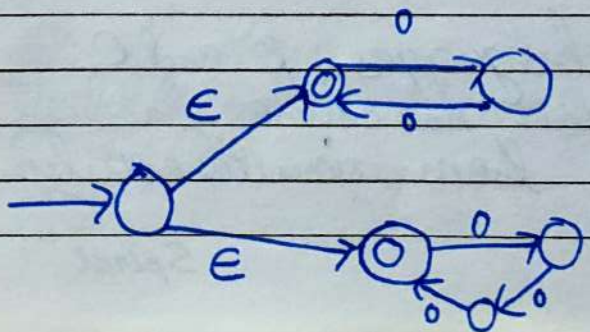
- and machine for C is



So machine for

$A = B \cup C$

in NFA is



Spiral

Date

i.e, at start, machine will guess whether it wants to check for B or check for C.

So, the union is easy to show in an NFA

Formal Definition for NFA

A NFA is defined as 5-tuple

$M = (Q, \Sigma, \delta, q, F)$

- Q is the set of states
- Σ is the alphabet and elements of Σ are called symbols
- $\delta : Q \times \Sigma_\epsilon \to P(Q)$ is the transition function

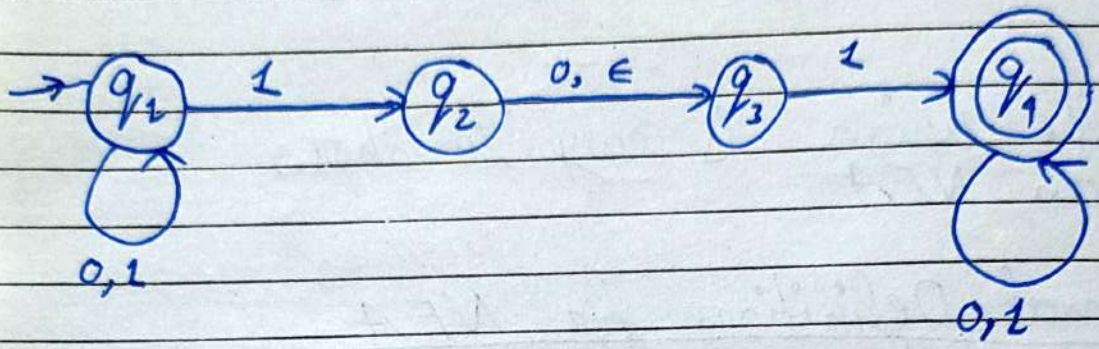
$\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$
and $P(Q)$ is powerset of Q.

So $\delta(r, a)$ takes current state, a symbol (including ϵ) and returns a set of next possible states.

- q is the start state of machine
- F is the set of accept states.

Date

Lets consider NFA



For this machine

$Q = \{q_1, q_2, q_3, q_4\}$

$\Sigma = \{0, 1\}$

$\delta =$

	0	1	∈
q_1	$\{q_1\}$	$\{q_1, q_2\}$	$\emptyset$
q_2	$\{q_3\}$	$\emptyset$	$\{q_3\}$
q_3	$\emptyset$	$\{q_4\}$	$\emptyset$
q_4	$\{q_4\}$	$\{q_4\}$	$\emptyset$

~~For the next symbol~~

$q = q_1$

$F = \{q_4\}$

Spiral

Date

Equivalence of DFAs and NFAs

If there is a NFA called N, which accepts language $\mathcal{L}(N)$, then there also exists a DFA called D which has language $\mathcal{L}(N)$.

Similarly, if there is a DFA called D with language $\mathcal{L}(D)$, there also exists an NFA with language $\mathcal{L}(D)$.

We can convert a NFA to DFA and vice versa, with both having same behaviour.

This is equivalence of DFA and NFA.

Converting DFA to NFA

The DFA is $D = (Q, \Sigma, \delta, q, F)$
it's equivalent NFA is

$N = (Q, \Sigma, \delta', q, F)$ i.e, only δ is changed

$$\delta'(r,a) = \begin{cases} \{\delta(r,a)\} & , \text{if } a \neq \epsilon \\ \phi & , \text{if } a = \epsilon \end{cases}$$

Date

Converting NFA to DFA

For a NFA,

$N = (Q, \Sigma, \delta, q, F)$

the equivalent DFA is

$D = (Q', \Sigma, \delta', q', F')$

More is changed when converting from NFA to DFA than vice-versa.

The changes are made to simulate the many many possible transitions of an NFA in a DFA.

i) $\boxed{Q' = P(Q)}$

Q' is a powerset of Q.
So the DFA will have set of states of NFA as it's own states.

~~$\delta'(R,a) = \bigcup_{r \in R} \delta(r,a)$~~

~~$\delta'(R,a) = \bigcup_{r \in R} [\delta(r,a)]$~~

Date

∈-closure

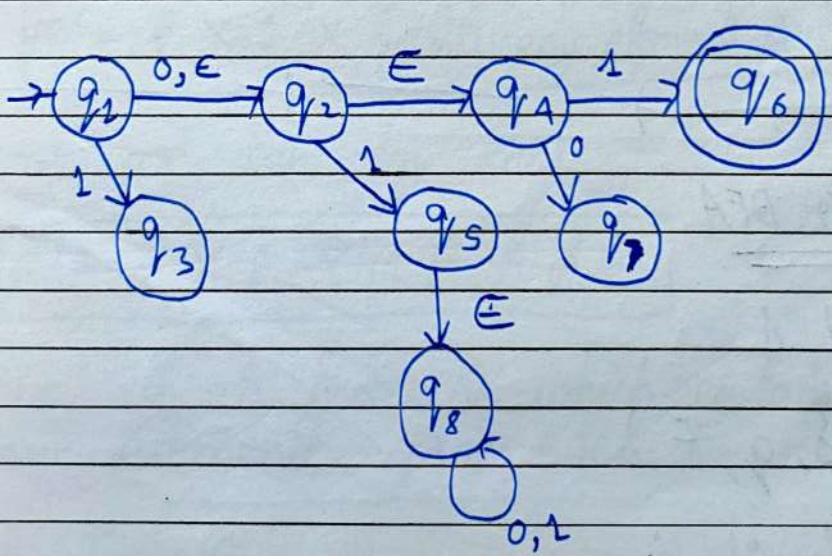
We will define ∈ closure as

$C_\epsilon(R)$ as the set of all states we can reach using one or more ∈ transitions from set of states R.

R contains states of NFA, (i.e $R \subseteq Q$)

∈ closure is used to get q' and δ' when converting from NFA to DFA.

Example for NFA



Suppose $R = \{q_1, q_5\}$

- States we can go from q_1 by zero or more ∈ transitions are $\{q_1, q_2, q_4\}$
- States we can go from q_5 from zero or more ∈ transitions are $\{q_5, q_8\}$

Spiral

Date

Therefore,

$C_\epsilon(\{q_2, q_5\}) = \{q_2, q_2, q_4\} \cup \{q_5, q_8\}$

$C_\epsilon(\{q_2, q_5\}) = \{q_2, q_2, q_4, q_5, q_8\}$

Converting NFA to DFA

Suppose the DFA is

$D = (Q, \Sigma, \delta, q, F)$

Then the NFA

Converting NFA to DFA

Suppose the NFA is

$N = (Q, \Sigma, \delta, q, F)$

Then the DFA for it is

$D = (Q', \Sigma, \delta', q', F')$

Spiral

Date

i) For Q'

$$Q' = P(Q)$$

Q' is the powerset of Q

ii) For q'

$$q' = C_\epsilon(\{q\})$$

q' is the ϵ closure of set $\{q\}$

iii) For F'

$$F' \subset P(Q)$$
$$F' = \{X : X \text{ contains any element of } F\}$$

or we can say that

$$F' = \{X : X \cap F \neq \emptyset\}$$

i.e, all sets from Q' which contain atleast one element from set F.

~~iv) For δ'~~

~~For $R = \{r_1, r_2, r_3 \ldots r_n\}$~~

~~$\delta'(R, a) = \delta(r_1$~~

Date

iv) For δ'

For any $R = \{r_1, r_2, r_3 \dots r_n\}$

$\delta'(R,a) = C_\epsilon(\delta(r_1,a)) \cup C_\epsilon(\delta(r_2,a)) \dots \cup C_\epsilon(\delta(r_n,a))$

ii) For δ'

For any $R = \{r_1, r_2, r_3 \dots r_n\}$ state of DFA

$$\delta'(R,a) = C_\epsilon(\delta(r_1,a)) \cup C_\epsilon(\delta(r_2,a)) \cup \dots \cup C_\epsilon(\delta(r_n,a))$$

i.e,

$$\delta'(R,a) = \bigcup_{r \in R} C_\epsilon(\delta(r,a))$$

Since $\emptyset$ has no elements,

$$\delta'(\emptyset, a) = \emptyset$$

$$\delta'(A,a) \cup \delta'(B,a) = \delta'(A \cup B, a)$$

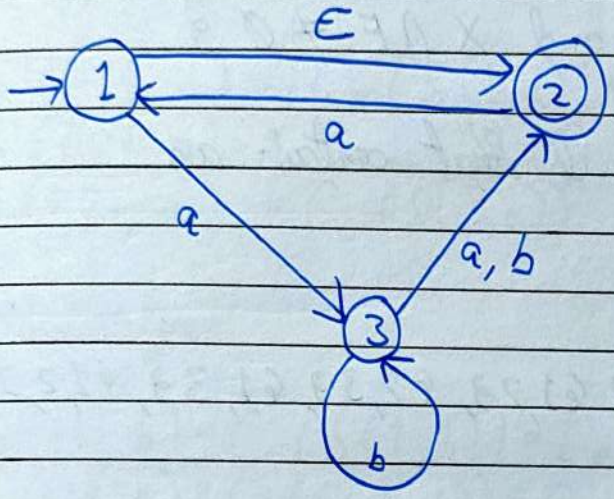
This property can help get δ' transitions faster for all states

Spiral

Date

Example of converting NFA to DFA

NFA is $(Q, \Sigma, \delta, q, F)$



$Q = \{1, 2, 3\}$

$\Sigma = \{a, b\}$

$q = 1$

$F = \{2\}$

$\delta =$

	a	b	ε
1	{3}	∅	{2}
2	{1}	∅	∅
3	{2}	{2,3}	∅

The DFA will be $(Q', \Sigma, \delta', q', F')$

i) $Q' = P(Q)$

$Q' = \{$ {1}, {2}
{3}, {1,2}
{2,3}, {1,3}
{1,2,3}, ∅
$\}$

Date

ii) $q' = C_\epsilon(\{q,3\}) = C_\epsilon(\{1,3\})$

$q' = \{1,2,3\}$

iii) $F' = \{X : X \in Q' \text{ and } X \cap F \neq \emptyset\}$

so all elements of Q' that contain an element from F

$F = \{2\}$

$Q' = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1,2\}, \{2,3\}, \{1,3\}, \{1,2,3\}\}$

Therefore

$F' = \{\{2\}, \{1,2\}, \{2,3\}, \{1,2,3\}\}$

iv) Fa finding δ'

We need to get the complete table

	a	b
$\emptyset$		
$\{1\}$		
$\{2\}$		
$\{3\}$		
$\{1,2\}$		
$\{2,3\}$		
$\{1,3\}$		
$\{1,2,3\}$		

Spiral

Date

* $\delta'(\emptyset, a) = \emptyset$ and $\delta'(\emptyset, b) = \emptyset$

* Now we can get δ' for all states having single element

$\delta'(\{1\}, a) = C_\epsilon(\delta(1,a)) = C_\epsilon(\{3\}) = \{3\}$
$\delta'(\{2\}, a) = C_\epsilon(\delta(2,a)) = C_\epsilon(\{1\}) = \{1,2\}$
$\delta(\{3\}, a) = C_\epsilon(\delta(3,a)) = C_\epsilon(\{2\}) = \{2\}$

Similarly,

$\delta'(\{1\}, b) = C_\epsilon(\delta(1,b)) = C_\epsilon(\emptyset) = \emptyset$
$\delta'(\{2\}, b) = C_\epsilon(\delta(2,b)) = C_\epsilon(\emptyset) = \emptyset$
$\delta'(\{3\}, b) = C_\epsilon(\delta(3,b)) = C_\epsilon(\{2,3\}) = \{2,3\}$

* Now we can use the property,
$\delta'(A \cup B, x) = \delta'(A,x) \cup \delta'(B,x)$

So to get,

$\delta'(\{1,2\}, a)$, we can get

$\delta'(\{1,2\}, a) = \delta'(\{1\}, a) \cup \delta'(\{2\}, a)$
$\delta'(\{1,2\}, a) = \{3\} \cup \{1,2\}$
$\delta'(\{1,2\}, a) = \{1,2,3\}$

Now by doing this process we can get complete δ'

Spiral

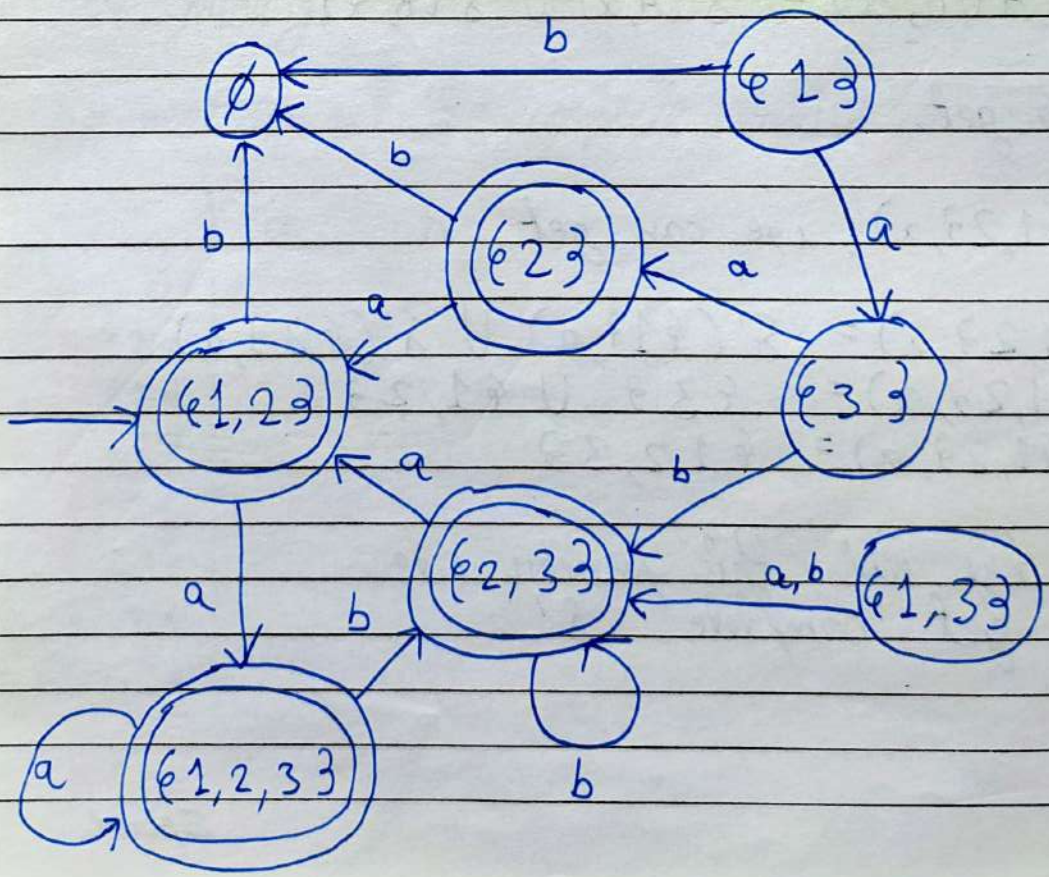
Date

δ' is

	a	b
$\emptyset$	$\emptyset$	$\emptyset$
{1}	{3}	$\emptyset$
{2}	{1,2}	$\emptyset$
{3}	{2}	{2,3}
{1,2}	{1,2,3}	$\emptyset$
{2,3}	{1,2}	{2,3}
{1,3}	{2,3}	{2,3}
{1,2,3}	{1,2,3}	{2,3}

So we got the DFA,

state diagram for this DFA is



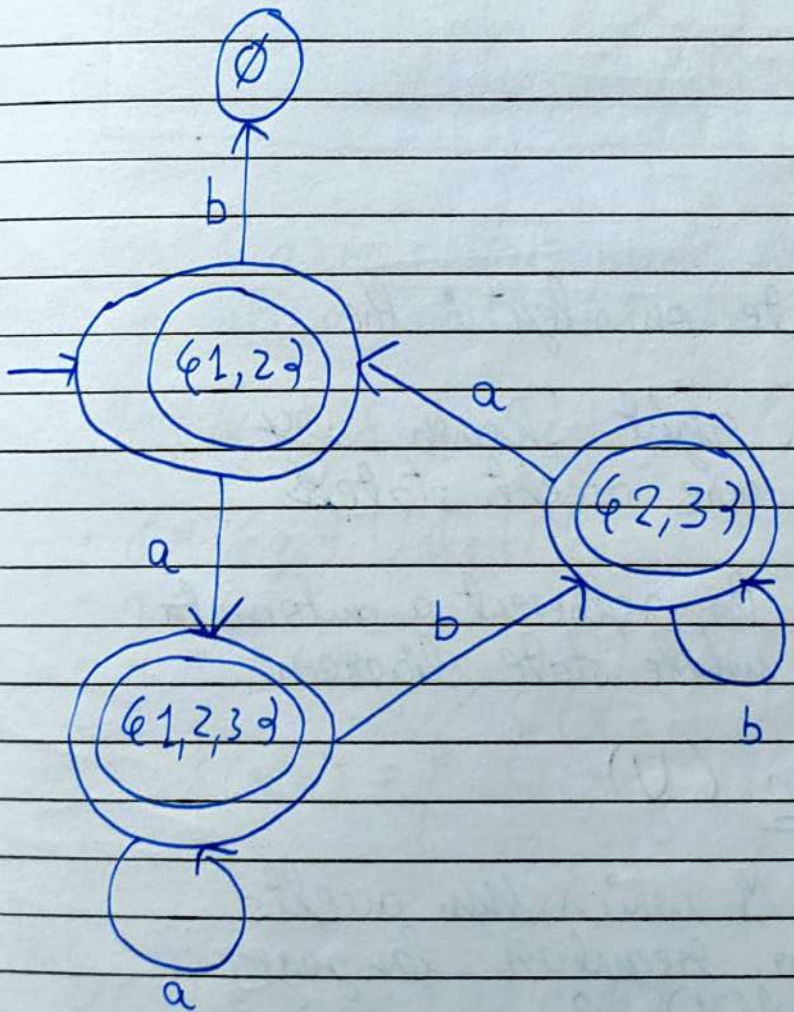
Spiral

Date

This DFA works perfectly, but we can minimize the number of states.

- States {1} and {1,3} have no incoming transitions so they can be removed
- After previous step, {3} has no incoming transition so it is also removed
- After previous steps, {2} has no incoming transitions, so it also removed

The minimized DFA is

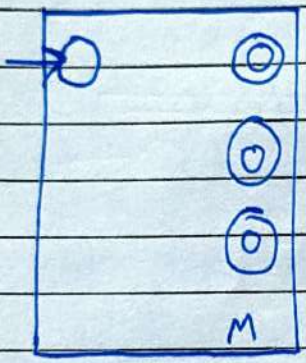


Spiral

Date

Closure of regular languages under regular operations

Suppose we represent a finite automata M as



→ The single state on left is the start state
→ and states on right shown by double circles are accept states

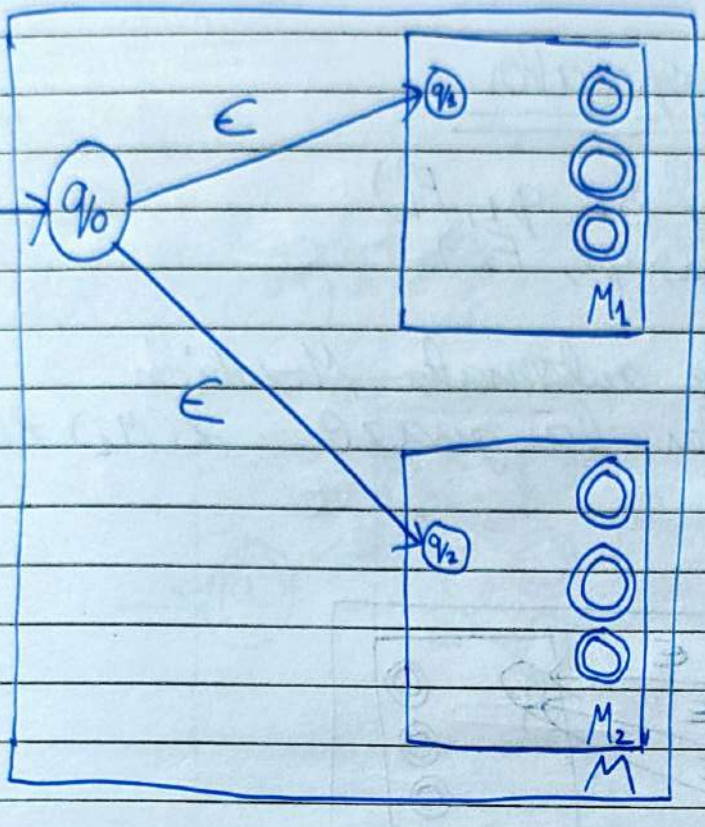
This allows us to represent a automata without drawing whole state diagram.

i) Union operator (U)

The machine M which accepts union of two regular languages $L(M_1)$ and $L(M_2)$ is.

$M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$
$M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$

Date



q_0 is a new state made for machine M
q_0 acts as start state for M.

$M = (Q, \Sigma, \delta, q_0, F)$

- $Q = \{q_0\} \cup Q_1 \cup Q_2$
- $q_0 = q_0$
- $F = F_1 \cup F_2$
- $$\delta(r,a) = \begin{cases} \delta_1(r,a) & \text{if } r \in Q_1 \\ \delta_2(r,a) & \text{if } r \in Q_2 \\ \{q_1, q_2\} & \text{if } r = q_0 \text{ \& } a = \epsilon \\ \emptyset & \text{if } r = q_0 \text{ \& } a \neq \epsilon \end{cases}$$

Here,
$L(M) = L(M_1) \cup L(M_2)$

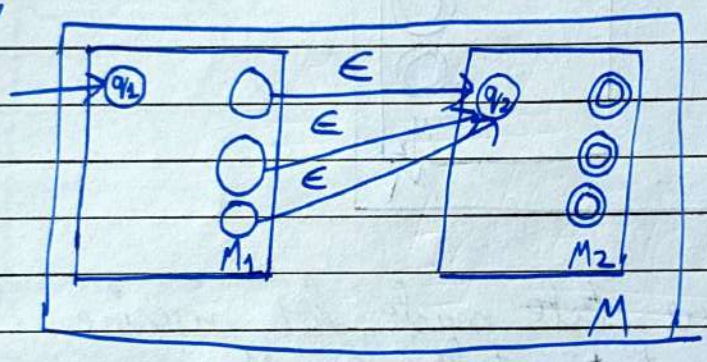
Spiral

Date

ii) Concatnation operator

$M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$
$M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$

The the finite automata M which accepts regular language $\mathcal{L}(M_1)\mathcal{L}(M_2)$ is,



$M = (Q, \Sigma, \delta, q, F)$

- $Q = Q_1 \cup Q_2$
- $q = q_1$
- $F = F_2$ (only accept states of F_2)
-

$$\delta(r,a) = \begin{cases} \delta_1(r,a) & ; \text{if } r \in F_1 \text{ and } a \neq \epsilon \\ \delta_1(r,a) & ; \text{if } r \in Q_1 \text{ and } r \notin F_1 \\ \delta_1(r,a) \cup \{q_2\} & ; \text{if } r \in F_1 \text{ and } a = \epsilon \\ \delta_2(r,a) & ; \text{if } r \in F_2 \end{cases}$$

$\mathcal{L}(M) = \mathcal{L}(M_1)\mathcal{L}(M_2)$

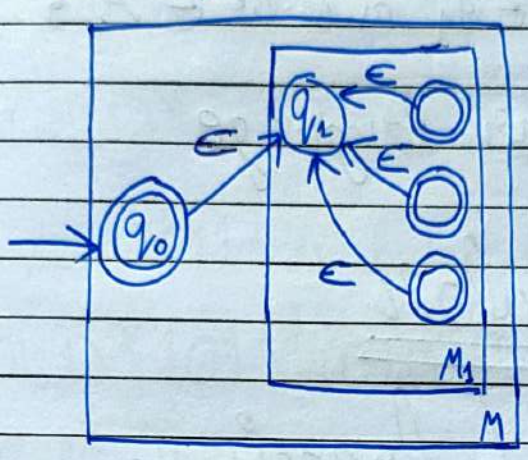
Spiral

Date

iii) Star operator

$M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$

The finite automata M which accepts $(L(M_1))^*$ is,



$M = (Q, \Sigma, \delta, q, F)$

- $Q = Q_1 \cup \{q_0\}$
- $q = q_0$
- $F = F_1 \cup \{q_0\}$
- $\delta(r,a) = \begin{cases} \delta_1(r,a) ; \text{if } r \in Q_1 \text{ and } r \notin F_1 \\ \delta_1(r,a) ; \text{if } r \in F_1 \text{ and } a \neq \epsilon, \\ \delta_1(r,a) \cup \{q_1\} ; \text{if } r \in F_1 \text{ and } a = \epsilon, \\ \{q_1\} ; \text{if } r = q_0 \text{ and } a = \epsilon \\ \emptyset ; \text{if } r = q_0 \text{ and } a \neq \epsilon \end{cases}$

iv) Complement

If A is regular language over alphabet Σ, then

$\bar{A} = \{w : w \in \Sigma^* \text{ and } w \notin A\}$

is also a regular language

Spiral

Date

v) Intersection

If A_1 and A_2 are two regular languages over alphabet Σ, then

$A_1 \cap A_2 = \{w : w \in A_1 \text{ and } w \in A_2\}$

is also a regular language

Regular Expressions

Regular expressions are means to describe languages.

The class of languages described by regular expressions are the class of regular languages.

A regular expression is

$$(0 \cup 1)\,01^*$$

This regular expression describes the language

- start with either 0 or 1. Indicated by $(0 \cup 1)$
- has 0 as second symbol. Indicated by 0
- end with zero or more 1s. Indicated by 1^*

Spiral

Date _____

Here are a few more examples

i) $\{w : w$ contains exactly two zeros$\}$

$$1^*01^*01^*$$

ii) $\{w : w$ contains atleast two 0s$\}$

$$(0 \cup 1)^* 0 (0 \cup 1)^* 0 (0 \cup 1)^*$$

iii) $\{w :$ 1011 is substring of $w\}$

$$(0 \cup 1)^* 1011 (0 \cup 1)^*$$

iv) $\{w :$ length of w is even$\}$

$$((0 \cup 1)(0 \cup 1))^*$$

v) $\{w :$ length of w is odd$\}$

$$(0 \cup 1)((0 \cup 1)(0 \cup 1))^*$$

vi) $\{w :$ first and last symbols of w are equal$\}$

.

vii) $\{$ 1011, 0 $\}$

$$1011 \cup 0$$

Note : + is also used inplace of $\cup$ in some places

Spiral

Date

Properties of regular expressions

For any alphabet Σ

1. ϵ is a regular expression
2. $\emptyset$ is a regular expression
3. For all $a \in \Sigma$, a is a regular expression.

Rules that can be used to combine regular expressions are

1. If R_1 and R_2 are regular expressions, then $R_1 \cup R_2$ is also a regular expression.
2. If R_1 and R_2 are regular expressions, the $R_1 R_2$ is also a regular expression. (this is simple concatnation)
3. If R is a regular expression, then R^* is a regular expression.

What is $\emptyset$ and ϵ?

$\emptyset$ is used to show a regular expression with no accepted language, whereas ϵ shows a regular expression with one accepted language, i.e, $\{\epsilon\}$

Spiral

Date

i.e.
Language of regular expression $\emptyset$ = $\{\}$
Language of regular expression ϵ = $\{\epsilon\}$

Languages described by regular expressions

For regular expression R_1 describing language L_1. And R_2 describing L_2,

Regular Expression	Language
$\emptyset$	$\{\}$ or $\emptyset$
ϵ	$\{\epsilon\}$
(For some $a \in \Sigma$); a	$\{a\}$
$R_1 \cup R_2$	$L_1 \cup L_2$
$R_1 R_2$	$L_1 L_2$
R^*	L^*

Example, suppose regular expression is

$(0 \cup \epsilon)(1 \cup \epsilon)$

We know
$L(0) = \{0\}$
$L(\epsilon) = \{\epsilon\}$
$L(1) = \{1\}$

So,
$L((0 \cup \epsilon)(1 \cup \epsilon)) = (\{0\} \cup \{\epsilon\})(\{1\} \cup \{\epsilon\})$
$= (\{0, \epsilon\})(\{1, \epsilon\})$
$= \{01, 0, 1, \epsilon\}$

Spiral

Converting regular expression to NFA

We can convert regular expression to NFA by following inductive steps.

For base case, we will show automatas for
i) Finite automata for regular expression ϵ
ii) Finite automata for regular expression $\emptyset$
iii) Finite automata for regular expression a (for all $a \in \Sigma$)

For inductive step, we show
i) If regular expression R_1 has equivalent finite automata M_1 & regular expression R_2 has equivalent automata M_2.
Then a finite automata exists for $R_1 \cup R_2$

ii) Finite automata for $R_1 R_2$ exists if finite automata for R_1 & R_2 exists.

Spiral

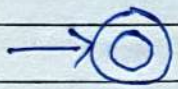
Date

iii) Finite automata R^* exists if finite automata for R exists.

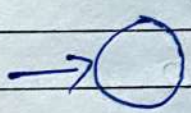
Proof:

Base Cases:

- NFA for regular expression ϵ

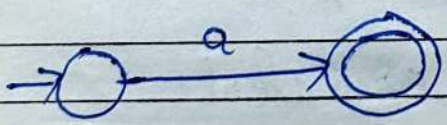


- NFA for regular expression $\emptyset$



(No accept state)

- NFA for regular expression a



Induction steps

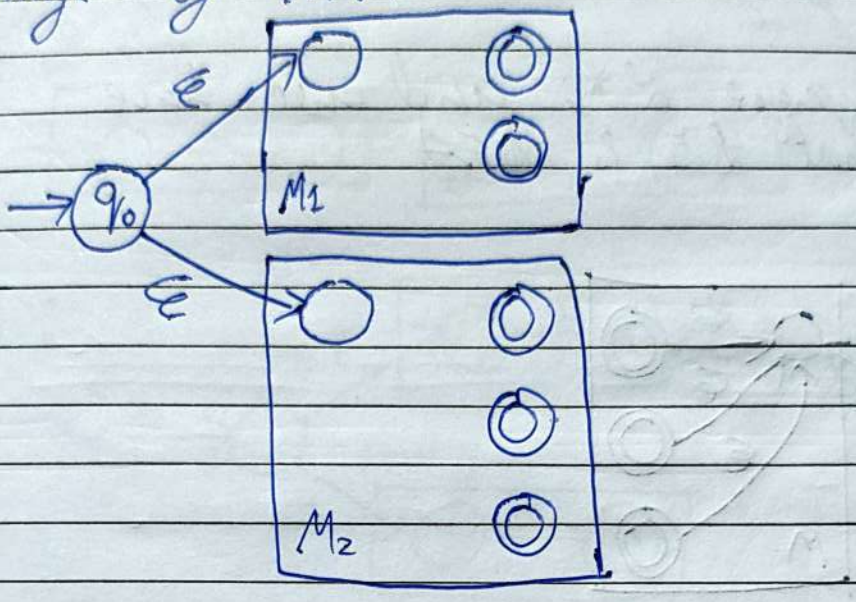
- NFA for $R_1 \cup R_2$

R_1 has language L_1 and NFA M_1
R_2 has language L_2 and NFA M_2

Spiral

Date

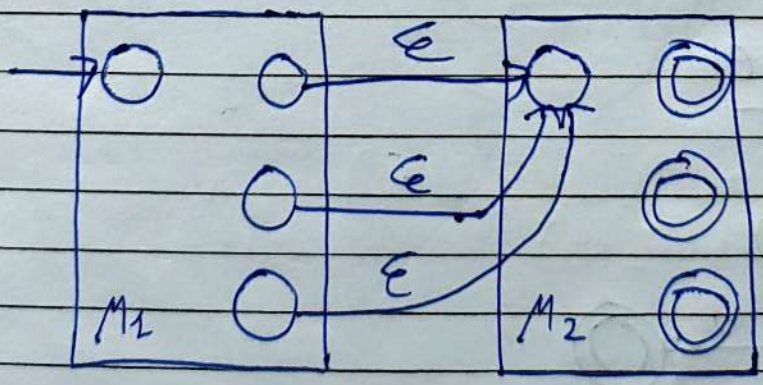
We know that $R_1 \cup R_2$ describes $L_1 \cup L_2$ and have shown that $L_1 \cup L_2$ is language of NFA



- NFA for $R_1 R_2$

R_1 has language L_1 and NFA M_1
R_2 has language L_2 and NFA M_2

$R_1 R_2$ has language $L_1 L_2$ and we have shown that $L_1 L_2$ is described by NFA



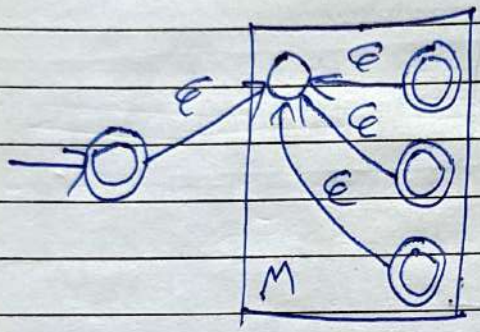
Spiral

Date

- NFA for R*

R has language L and NFA M

R* has language L*, and we have shown that L* is NFA



Example,
convert regular expression
$(ab \cup a)^*$
to NFA

$\Sigma = \{a, b\}$

Base cases are:

$\emptyset$:

ϵ :

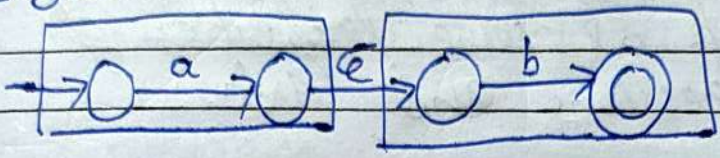
a :

b :

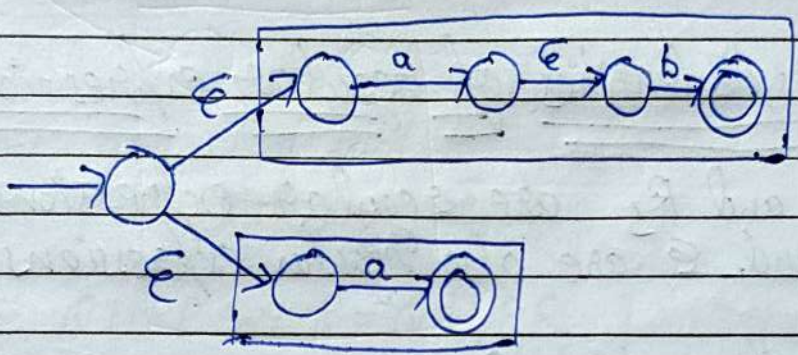
Date

Creating NFA

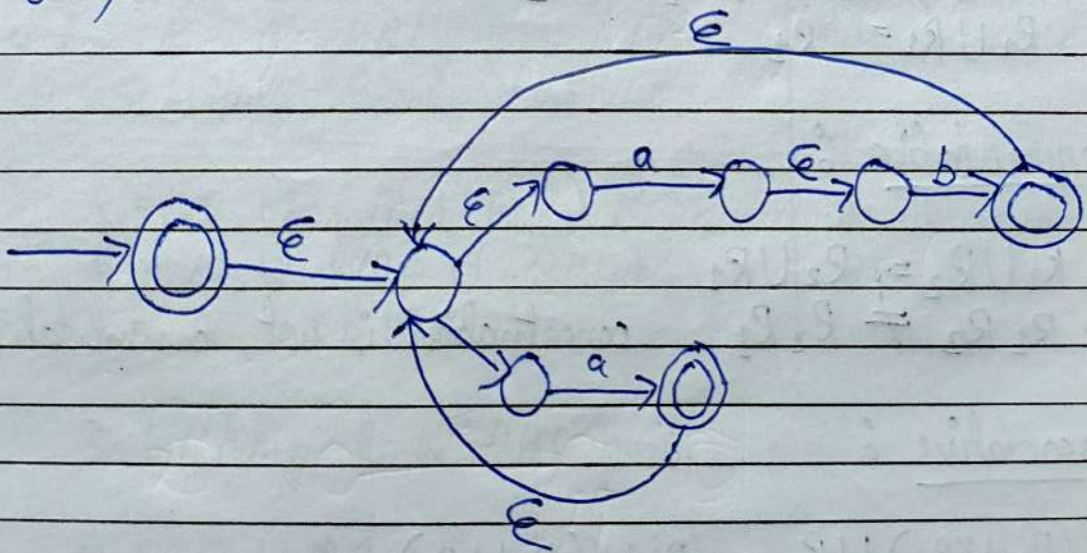
ab:



ab Va:



(ab Va)*:



Spiral

Date

Converting DFA to regular expression

To convert DFA to regular expression, we need to look at some properties of regular expressions and Arden's theorem.

Properties / Identities of regular expressions

R_1, R_2 and R_3 are regular expressions.
$\emptyset$ and ϵ are also regular expressions

1. $R_1 \emptyset = \emptyset R_1 = \emptyset$
2. $R_1 \epsilon = \epsilon R_1 = R_1$
3. $R_1 \cup \emptyset = \emptyset \cup R_1 = R_1$
4. $R_1 \cup R_1 = R_1$

Commutative :

5. $R_1 \cup R_2 = R_2 \cup R_1$
6. $R_1 R_2 \neq R_2 R_1$ concatnation is not commutative

Associative :

7. $(R_1 \cup R_2) \cup R_3 = R_1 \cup (R_2 \cup R_3)$
8. $(R_1 R_2) R_3 = R_1 (R_2 R_3)$

Distributive :

Date

9. $R_1(R_2 \cup R_3) = R_1R_2 \cup R_1R_3$
10. $(R_1 \cup R_2)R_3 = R_1R_3 \cup R_2R_3$

11. $R_1 \cup (R_2R_3) \neq (R_1 \cup R_2)(R_1 \cup R_3)$
Union is not distributive over concatenation

Arden's theorem

For alphabet Σ, the three regular expressions are P, Q and R and P does not contain ϵ.

If $R = Q \cup RP$ or $R = Q \cup PR$
then R has a unique solution
$R = QP^*$ or $R = P^*Q$ respectively

Proof: $R = Q \cup RP$
Substitute R in RHS.

$R = Q \cup (Q \cup RP)P$
$R = Q \cup (QP \cup RPP)$
$R = Q \cup QP \cup RP^2$

Substituting R in RHS again

$R = Q \cup QP \cup (Q \cup RP)P^2$
$R = Q \cup QP \cup QP^2 \cup RP^3$

Therefore, by repeatedly substituting

$R = Q \cup QP \cup QP^2 \cup QP^3 \cup \ldots\ldots$

Hdr

$R = P \cup QR$
$\Downarrow$
$R = Q^*P$

Date

Using distributive property

$R = Q(\varepsilon \cup P \cup P^2 \cup P^3 \cup \dots)$

~~Using property $P \cup P^2 \cup P^3 \cup \dots = P^*$~~

~~$R = Q(\varepsilon \cup P^*)$~~

~~Since, $\varepsilon \cup R_1 = R_1$~~

Using property $\varepsilon \cup P \cup P^2 \cup P^3 \cup \dots = P^*$

$\boxed{R = QP^*}$

So if given $R = Q \cup RP$ and P does not contain ε Solution is $R = QP^*$ ($R = Q \cup PR$ solves to $R = P^*Q$)

Conversion from DFA to regular expression

We can now use Arden's theorem for this conversion.

We do this using a series of regular expression equations and then solving using Arden's theorem.

Let $M = (Q, \Sigma, \delta, q, F)$ be DFA to convert.

We create a equation for every state in this DFA.

Spiral

Date

For a state r in set Q, regular expression is

$$R_r = \begin{cases} \bigcup_{a \in \Sigma} a R_{\delta(r,a)} & , \text{if } r \notin F \\ \left(\bigcup_{a \in \Sigma} a R_{\delta(r,a)} \right) \cup \epsilon & , \text{if } r \in F \end{cases}$$

R_r is the regular expression for M if M had start state r.

Reason why

R_r is $\left(\bigcup_{a \in \Sigma} a R_{\delta(r,a)} \right)$ is because, if M does a transition for a symbol 'a', the regular expression for such can become $aR_{\delta(r,a)}$ so it matches the first processed symbol and regular expression from resulting state $\delta(r,a)$ i.e $R_{\delta(r,a)}$

Accounting for all outgoing transitions given

$$R_r = \left(\bigcup_{a \in \Sigma} a R_{\delta(r,a)} \right)$$

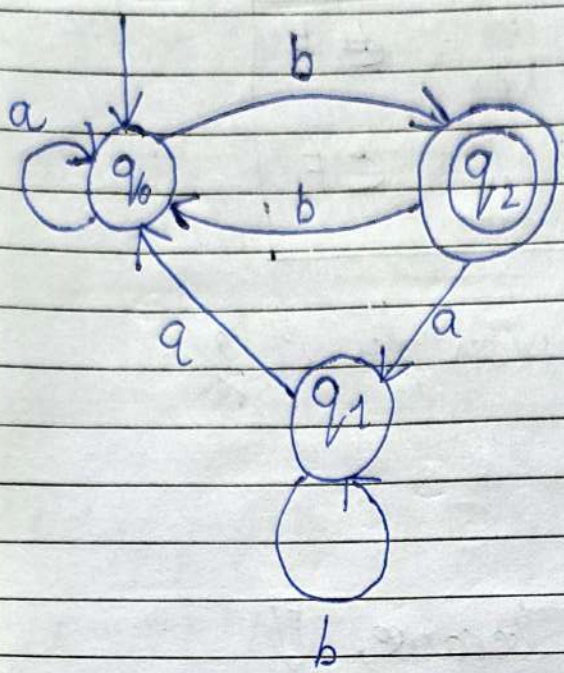
Reason why we also have ϵ if $r \in F$ is because if M starts at r, it can accept ϵ.

These set of equations are then used to solve for R_q

Spiral

Date

Example,



For this DFA, regular expressions are

$R_{q_0} = aR_{q_0} \cup bR_{q_2}$ — (i)

$R_{q_1} = aR_{q_0} \cup bR_{q_1}$ — (ii)

For final states

$R_{q_2} = aR_{q_1} \cup bR_{q_0} \cup \epsilon$ — (iii)

Since actual start state is q_0, we need to solve R_{q_0}

Putting eq (iii) in eq (i)

$R_{q_0} = aR_{q_0} \cup b(aR_{q_1} \cup bR_{q_0} \cup \epsilon)$

Spiral

Date

$R_{q_0} = aR_{q_0} \cup baR_{q_1} \cup bbR_{q_0} \cup b$ — (iv)

Taking eq (iii)

$R_{q_1} = aR_{q_0} \cup bR_{q_1}$

Using Arden's Theorem

$R_{q_1} = \underbrace{a}_{R}\underbrace{R_{q_0}}_{Q} \cup \underbrace{b}_{P}\underbrace{R_{q_1}}_{R}$

$R_{q_1} = b^* a R_{q_0}$

Putting this result in eq (iv)

$R_{q_0} = aR_{q_0} \cup ba b^* a R_{q_0} \cup bbR_{q_0} \cup b$

$R_{q_0} = (a \cup bab^*a \cup bb)R_{q_0} \cup b$

Using Arden's Theorem

$R_{q_0} = \underbrace{(a \cup bab^*a \cup bb)}_{P}\underbrace{R_{q_0}}_{R} \cup \underbrace{b}_{Q}$

$\boxed{R_{q_0} = (a \cup bab^*a \cup bb)^* b}$

Spiral

Date

Pumping Lemma

Pumping lemma states that for a regular language L, there is a constant p such that any string $w \in L$ can be split into three substrings if $|w| \geq p$ so that

$$w = xyz \quad ; y \neq \epsilon \text{ and } |xy| \leq p$$

such that y can be pumped and it will still be part of language.

i.e,

~~$\{xyz, xyyz, xyyyz, \ldots, xy^i z\}$~~

$\{xyz, xyyz, xyyyz, \ldots, xy^i z\} \subset L$

This property is exclusive to regular languages thus it can be used to distinguish regular and non regular languages.

* Pumping lemma thus shows that all languages with finite no. of strings are regular, since we can simply choose p to be larger than longest string

Date

Applications of pumping lemma

The most important application of pumping lemma is to show that a language in non-regular.

First Example $A = \{w : 0^n 1^n \text{ and } n \geq 0\}$

We proof by contradiction that A is non-regular.

Suppose A is regular
∴ it has pumping length (p) which is some positive integer.

Let's consider $0^p 1^p$
∵ It's length is greater than pumping lenght

So it must have y component which can be pumped such that $y \neq \epsilon$ and $|xy| \leq p$

So the string is

000... (p times) 111 ... (p times)

Since $|xy| \leq p$, $|z| \geq (2p - p)$
So
000... 111...
↓
z

Spiral

Date

All the 1's on right have to be a part of $|z|$ to satisfy this condition.

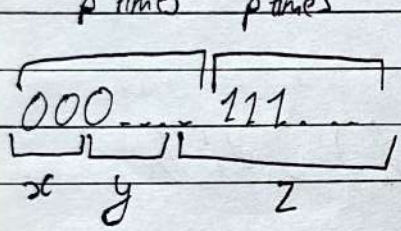
Therefore, y is only made up of 0's

So

$x \Rightarrow$ zero or more 0's

$y \Rightarrow$ one or more 0's

$z \Rightarrow$ zero or more 0's then 1's



Now if we pump y, the number of 0's will be greater than 1's

So $xy^iz \notin A$ for $i > 1$

Therefore by contradiction A is non-regular.

Second Example

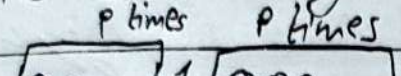
$A = \{ w : xx \text{ where } x \in \{0,1\}^* \}$

i.e, every $w \in A$ can be split into two equal strings.

Date

Assume A is a regular language with pumping length p.

Then, $0^p 1\, 0^p 1$ belongs to A since it is symmetrical.

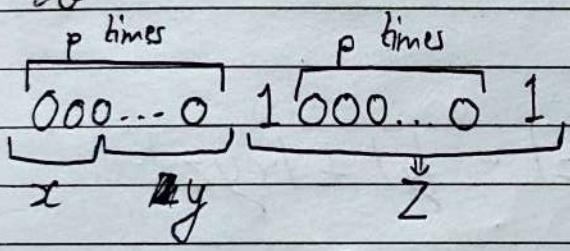


Total length = $2p+2$

Since $|xy| \leq p$ then $|z| \geq (2p+2) - p$

$|z| \geq p+2$

So



y is always only 0's.

If we try to pump y, string no longer belongs to A

So, $xy^iz \notin A$ for $i > 1$

So by contradiction, A is not regular.

Spiral

Date

Third example $A = \{1^{n^2} : n \geq 0\}$
i.e, 1 repeated n^2 times

Assume A is regular, so assume pumping length is p

1^{p^2} has length p^2 which is bigger than p

So 1^{p^2} should be in A according to assumption.

1^{p^2} has three components x, y and z such that $y \neq \epsilon$ and $|xy| \leq p$

So $|xyz| = p^2$

Now,

$$|xy^2z| = |xyyz| = |y| + |xyz|$$

$$|xy^2z| = |y| + p^2 \quad \text{—(1)}$$

Since, $|xy| \leq p$

$$1 \leq |y| \leq p$$

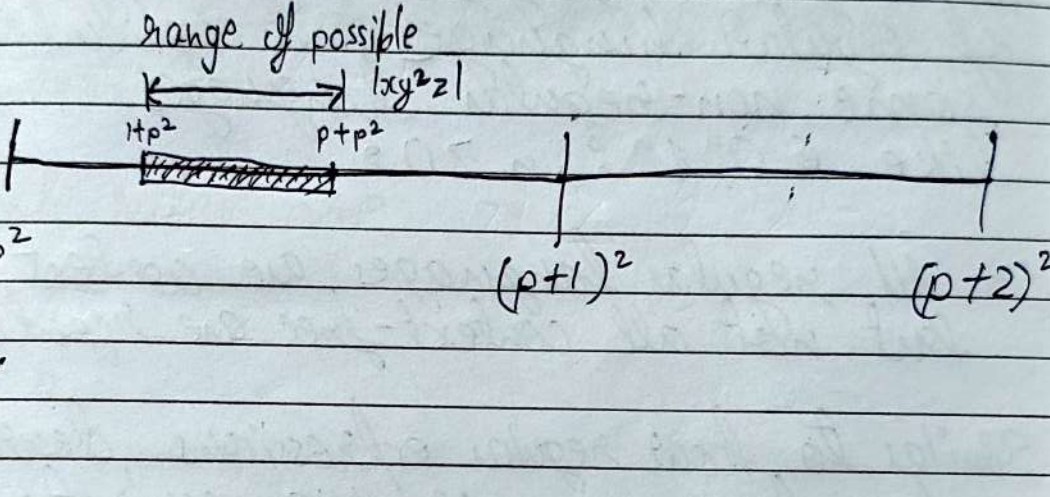
$$1 + p^2 \leq |y| + p^2 \leq p + p^2$$

Using (1)

$$\boxed{1 + p^2 \leq |xy^2z| \leq p + p^2}$$

Date

Now, since $|xy^2z|$ will have longer length than $|xyz|$ it must fall range should cover atleast one complete square But it does not fall in any square



So, $|xy^2z|$ cannot be a perfect square

$xy^2z \notin A$

So by contradiction, A is not regular.