Deterministic Finite Automata

* An automaton is a simple computing machine.

* We want to know how the computation works. That is why we use simpler machine like DFA to understand the underline works properly.

* In DFAs we only want to store the useful informations as the memory is lot less in DFA.

* Symbol, alphabet, string, Language.

* We construct DFAs in such a manner, that it recognizes a set of strings with one/many properties.

set of strings ⟹ Language.

* Each language has a set of alphabets.

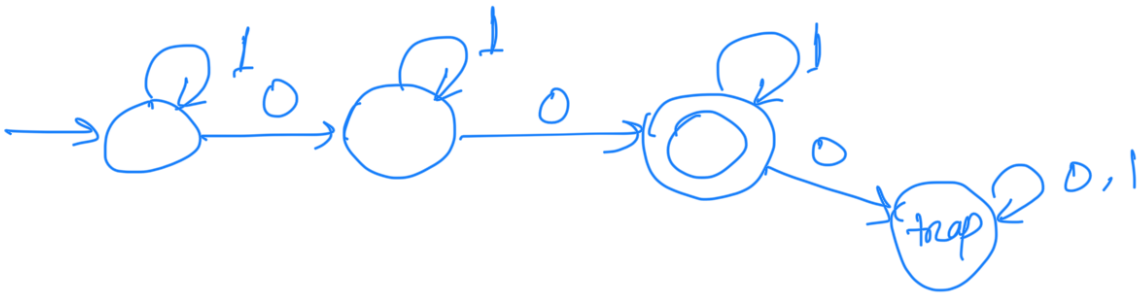
* Any language recognized by a DFA is called a regular language.

* Think yourself as a DFA.

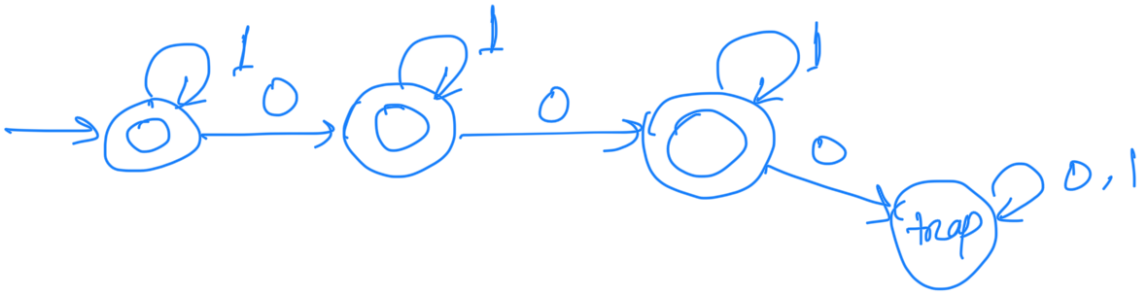
* We will learn how to construct DFAs using examples.

(1) $L = \{ w \in \{0,1\}^* , w \text{ contains at least two 0's} \}$

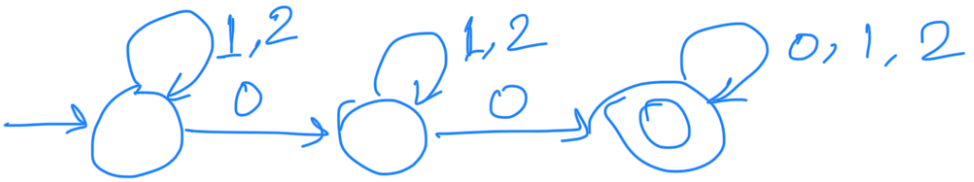
(2) $L = \{ w \in \{0,1\}^* , w \text{ contains exactly two 0's} \}$



(3) $L = \{ w \in \{0,1\}^* , w \text{ contains at most two 0's} \}$

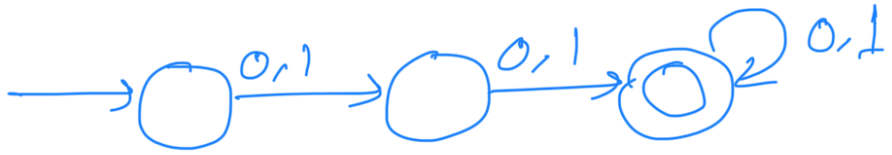


(4) $L = \{ w \in \{0,1,2\}^* , w \text{ contains at least two 0's} \}$



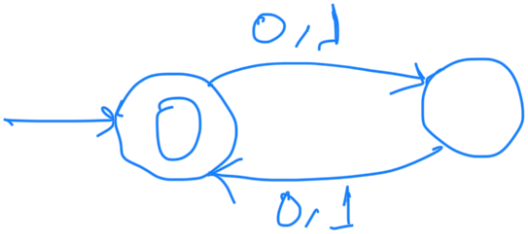
[check the difference]

(5) $L = \{ w \in \{0,1\}^*,$ length of w is at least two $\}$

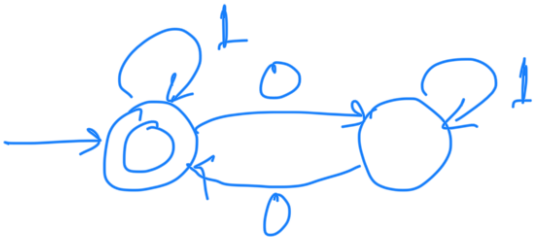


[try to solve length of w
① exactly two
② at most two]

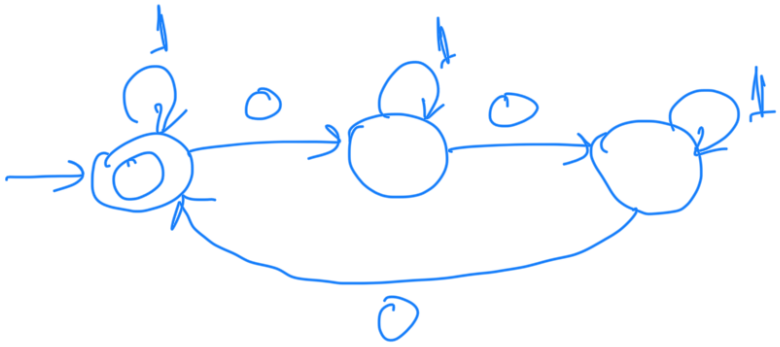
(6) $L = \{ w \in \{0,1\}^*,$ length of w is even / multiple of two $\}$



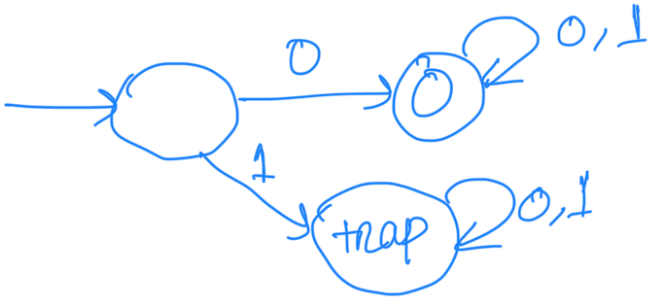
(7) $L = \{ w \in \{0,1\}^*,$ the count of 0's in w is even/multiple of two $\}$



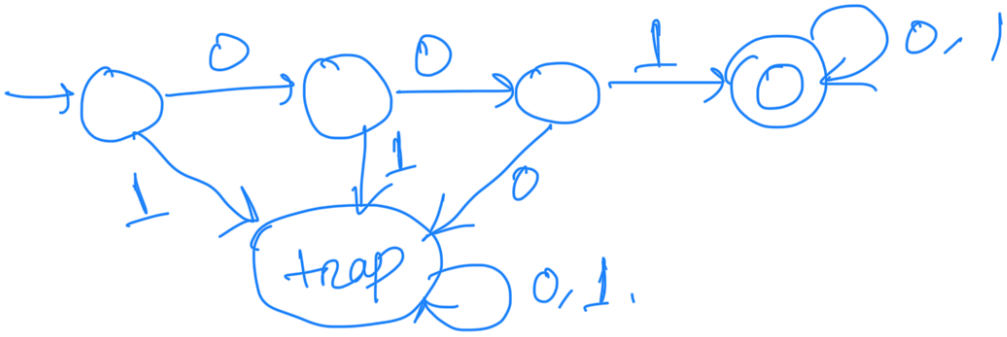
(8) $L = \{ w \in \{0,1\}^* $, the count of 0's in w is a multiple of three $\}$



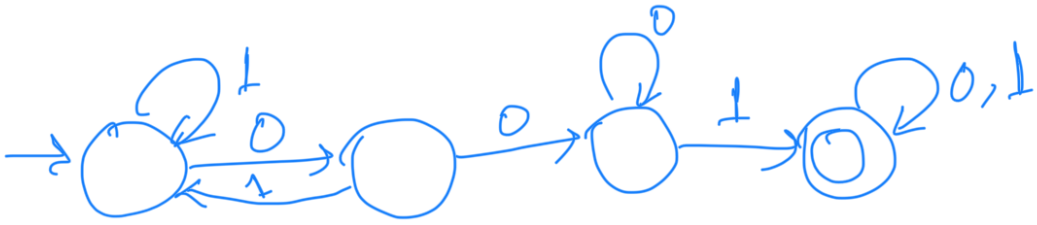
(9) $L = \{ w \in \{0,1\}^* $, w starts with 0 $\}$



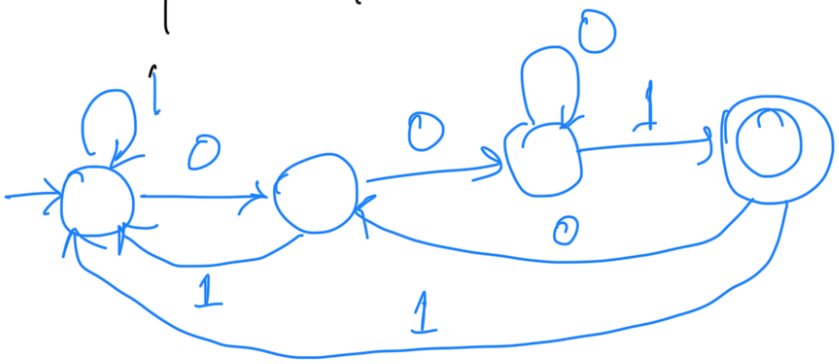
(10) $L = \{ w \in \{0,1\}^* $, w starts with 001 $\}$



(11) $L = \{ w \in \{0,1\}^* $, w contains 001 as a substring $\}$



(12) $L = \{ w \in \{0,1\}^* , w \text{ ends with } 001 \}$

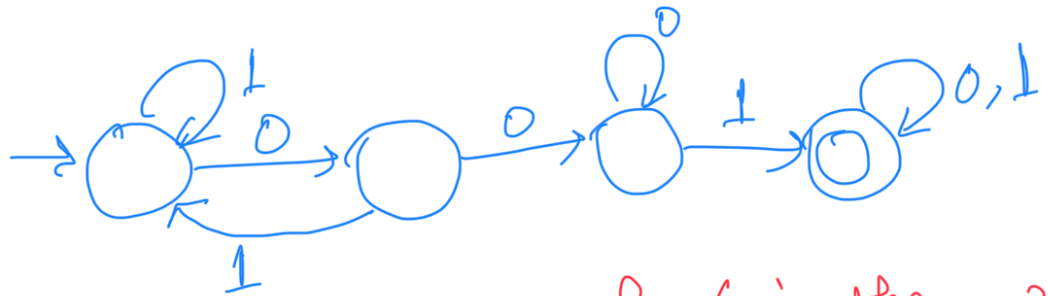


(*) Sometimes some problems state the word "does not contain 001". This is the opposite of (11) example.

Let, L = "contains 001 as a substring"

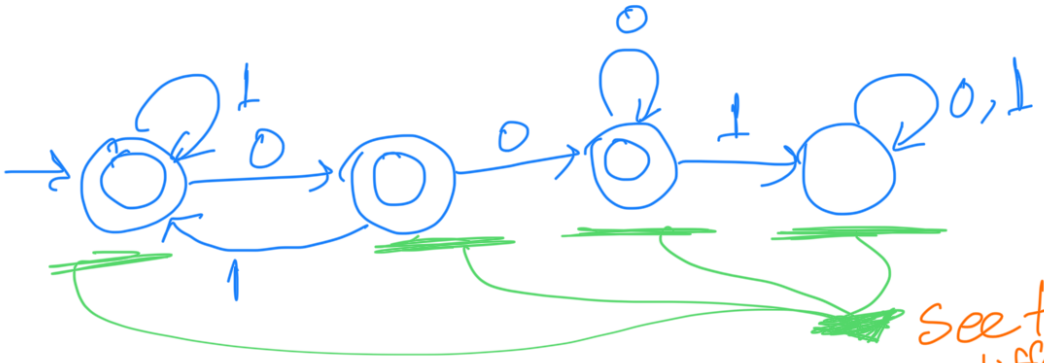
L' = "does not contain 001"

L

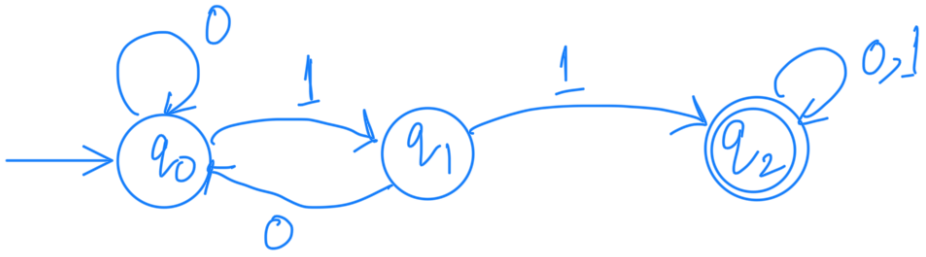


* Accepting state of L' is the opposite of L

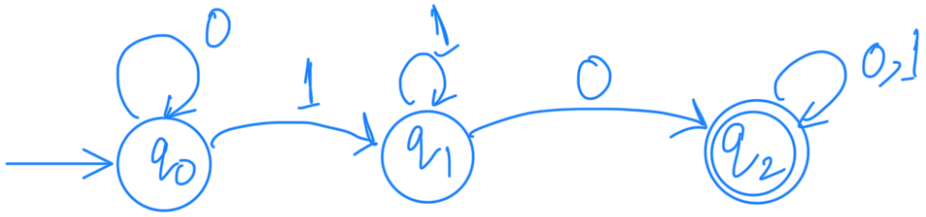
L'



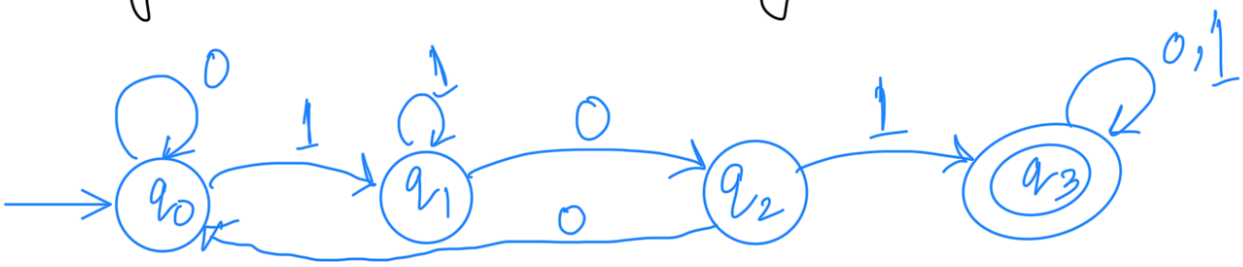
13 Construct a DFA that recognizes strings with 11 as substring.



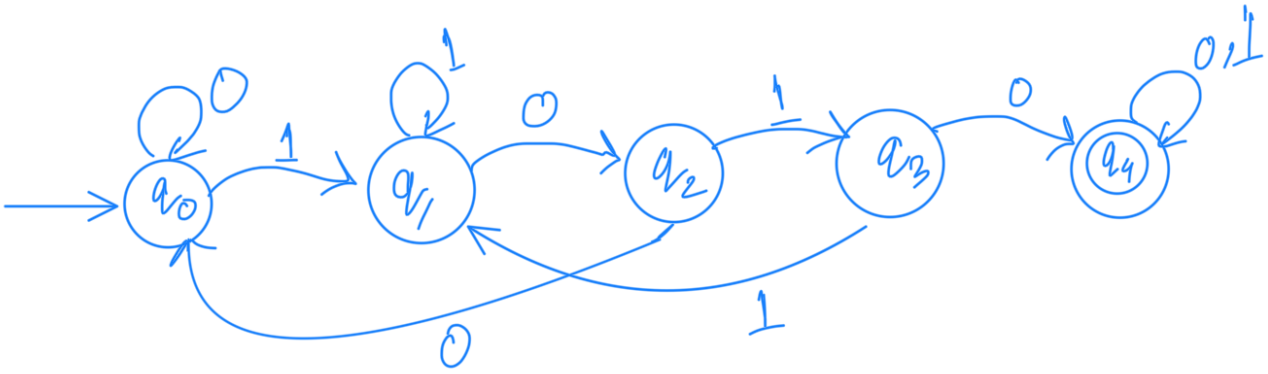
14 Construct a DFA that recognizes strings with 10 as substring.



15 Construct a DFA that recognizes strings with 101 as substring.

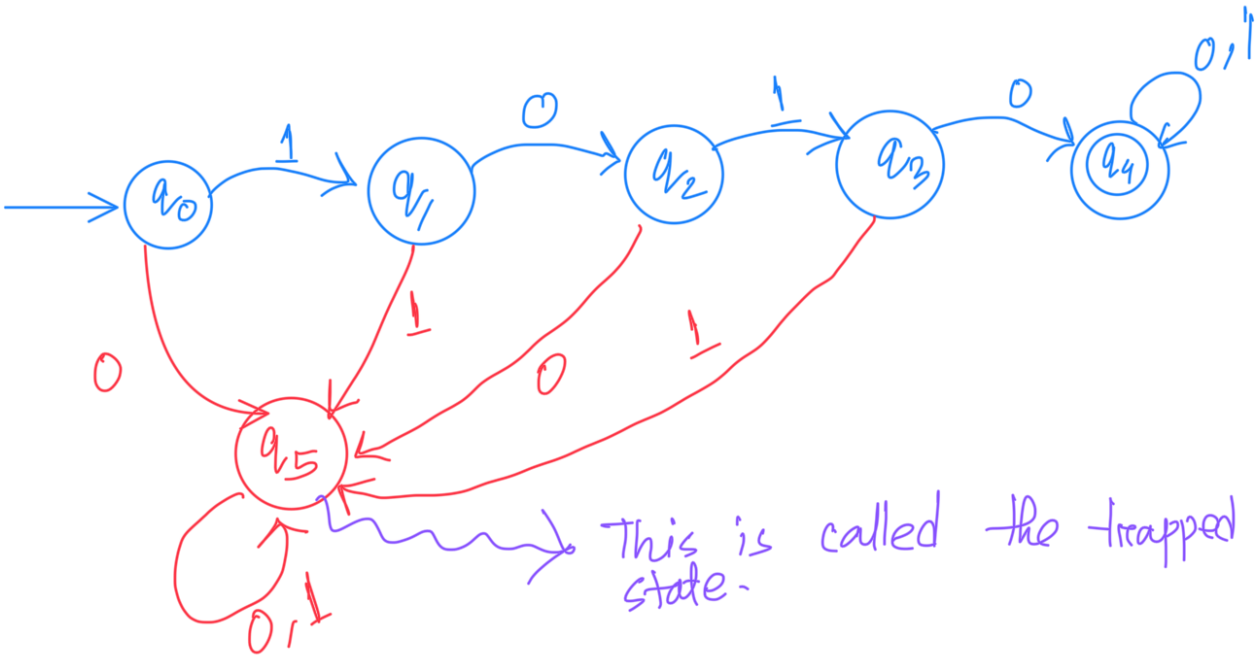


16 Construct a DFA that recognizes strings with 1010 as substrings.

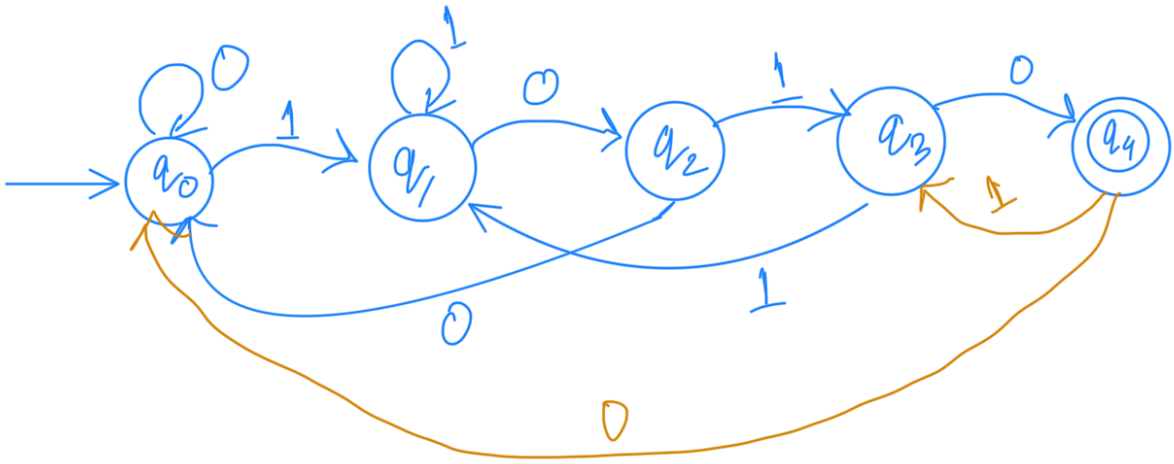


17 Construct a DFA that recognizes strings that start with 1010.

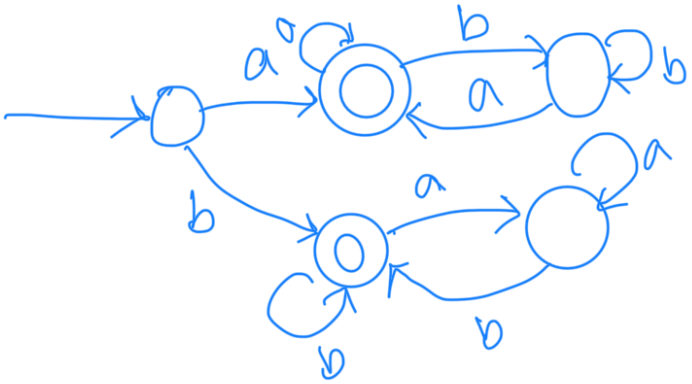
strings that start with 1010



(18) Construct a DFA that recognizes strings that end with 1010



(19) $L = \{ w \in \{a, b\}^* , w$ starts and ends with same symbol $\}$



Practice:

(i) $L = \{ w \in \{0,1\}^* $, number of 0's in w is not a multiple of 3 $\}$

(ii) $L = \{ w \in \{0,1\}^* : w$ contains exactly one 00 $\}$

(iii) $L = \{ w \in \{0,1\}^*$, every 3rd symbol in w is 1 $\}$

(iv) $L = \{ w \in \{a,b\}^*$, w contains atleast two "ab"s $\}$

(v) $L = \{ w \in \{0,1\}^*$, An odd number of 0's follow the last 1 $\}$

Regular Operations:

DFA recognises regular languages. The operations that can be applied upon some regular languages and the resulting languages still be regular languages are called regular operations.

We have 3 regular operations.

- Union
- Concatenate
- Star

A, B → regular languages

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

Concatenate: $A \circ B = \{ xy \mid x \in A \text{ and } y \in B \}$

Star : $A^* = \{ x_1 x_2 \cdots x_k \mid k \geq 0 \text{ and } x_i \in A \}$

$= A^0 \cup A^1 \cup A^2 \cup A^3 \cup \cdots$

$= \{\epsilon\} \cup \{x_i \in A\} \cup \{x_i \in A \circ A\} \cup \{x_i \in A \circ A \circ A\} \cup \cdots$

Union :

Suppose DFA M_A recognizes language A.
" M_B recognizes language B.

If string s is in language $A \cup B$, then we cannot parse s through M_A and rewind it and again pass it through M_B. It's because after reading each alphabet from s, we forget that and don't store s. So we need to pass s in M_A and M_B

simultaneously. The DFA M that recognizes AUB is the cartesian product of the states.

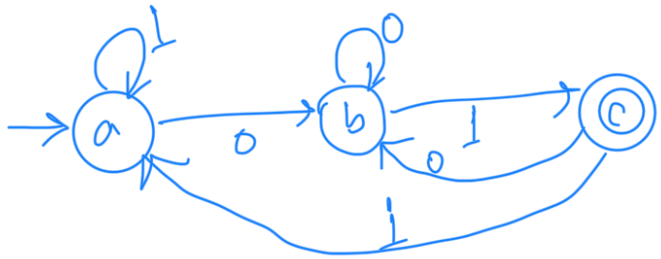
* See an example to fully understand it.

Example:

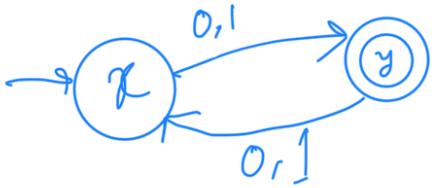
$A = \{ w \in \{0,1\}^* : w \text{ ends with } 01 \}$

$B = \{ w \in \{0,1\}^* : \text{Length of } w \text{ is odd} \}$

$M_A \Rightarrow$

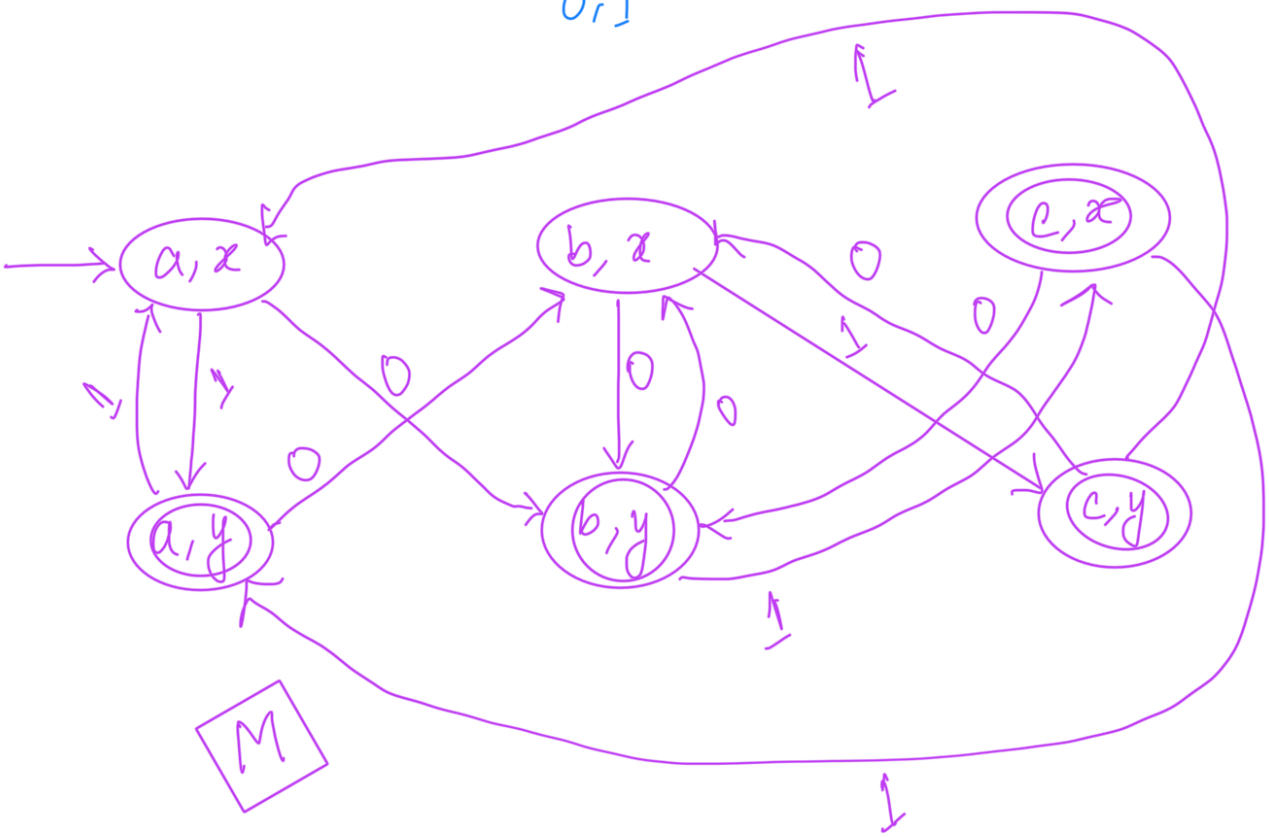


$M_B \Rightarrow$



L = AUB

M is the DFA that recognises L.

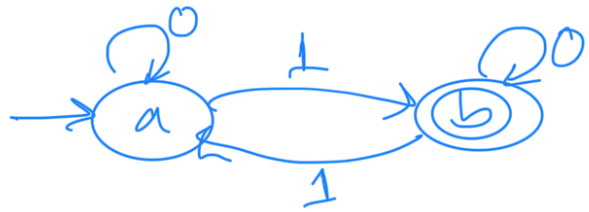


Example 7:

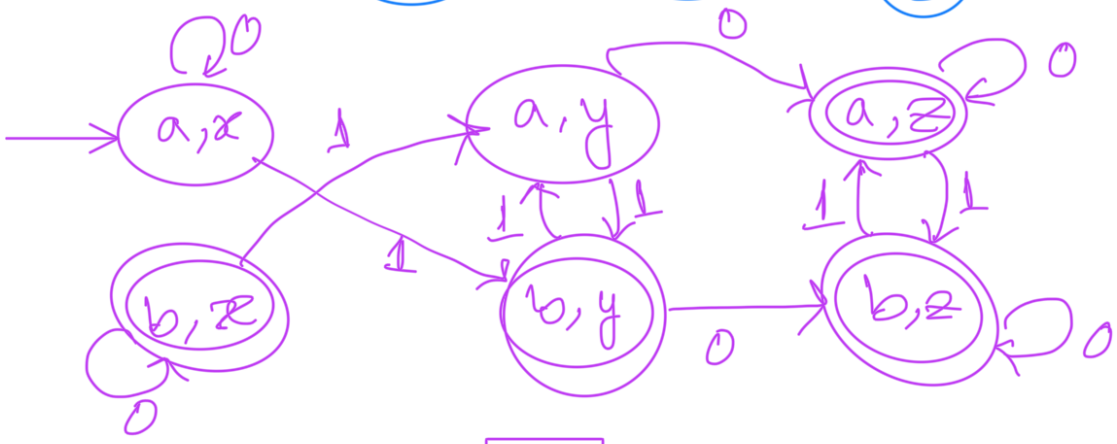
A = strings with odd number of 1's

B = " " 10 as substring.

$M_A \Rightarrow$



$M_B \Rightarrow$



M

start state ⇒ that contains both a and x.
((a,x))

Accepting state ⇒ that contains b or z or both.
((b,x), (b,y), (a,z), (b,z))

Cross Product

Similarly, we can use AND or NOT just like Union (OR), ↳ Intersection

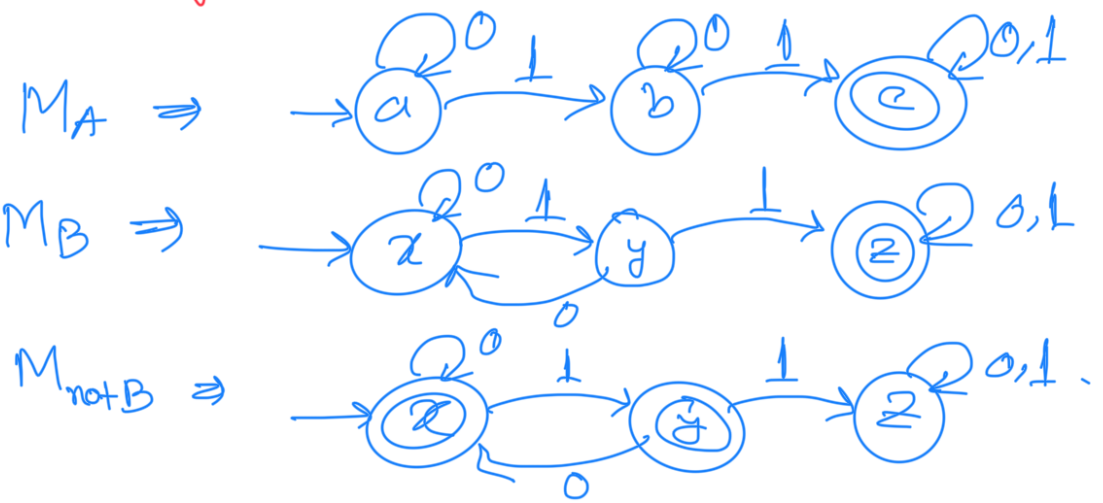
Example 8: Construct a DFA that recognizes strings with atleast two 1's but no consecutive 1's.

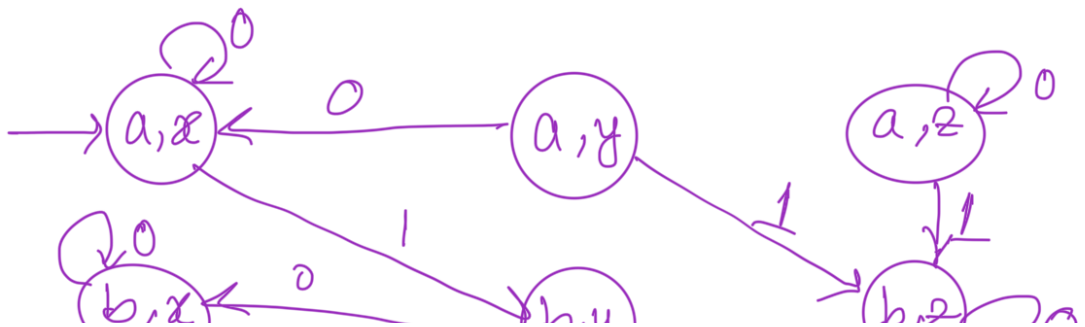
A = strings with atleast two 1's

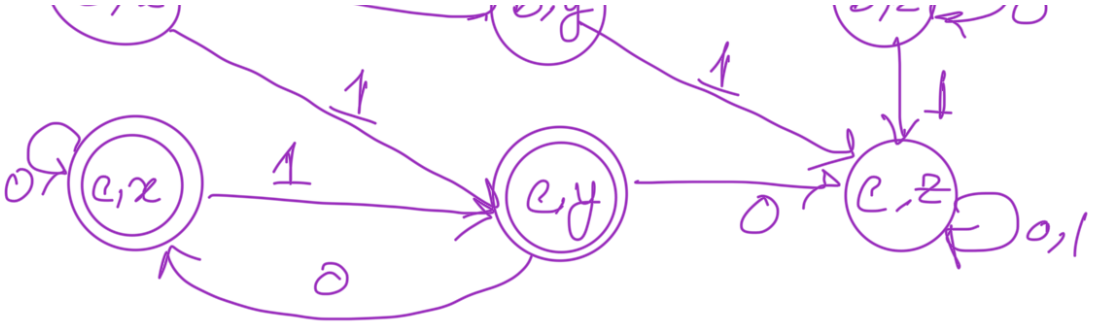
B = consecutive 1's.

NotB = no consecutive 1's

M recognizes (A and (not B))







starting state ⇒ (q,x)

Accepting state ⇒ that contains both {c and (x or y)} ⇒ (c,x) (c,y)

Concatenation:

$$A \circ B = \{ xy \mid x \in A \text{ and } y \in B \}$$

The main problem is that we don't know where to split any input string s to get x, y where $x \in A$ and $y \in B$.

For example:

$$A = \{ w \mid w \text{ ends with } 10 \}$$

$$B = \{ w \mid \text{length odd} \}$$

Suppose input is 00110110 .

We can divide it in two ways.

00110110 ⇒ if we do that, then the input is not in the language that represents $\boxed{A \circ B}$. $x \in A$, $y \notin B$

But this is infact in the language .

00110110 ⇒ $x = 00110110$, $x \in A$

$y = 0$, $y \in B$

We can not explain this with the help of DFA.

We need NFA.