Theory of Computation
Assignment-1

Soumik Mandal (S20160010091) soumik.m16@iiits.in

1

Question 1

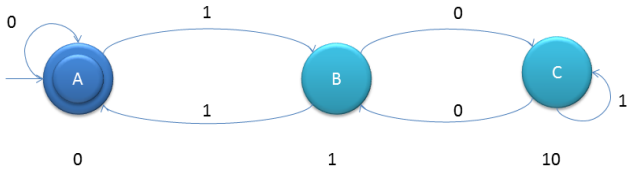
When the language consisting of binary strings divisible by 3 but not by 4. Using the product construction procedure build DFA to recognize the language L. Show all steps clearly. Transition diagrams are enough (there is no need to specify a DFA by explicitly giving all its 5 components; transition diagram has all these).

For a binary string divisible by 3

Transition Table :

Delta	0	1
q_0	q_0	q_1
q_1	q_2	q_0
q_2	q_1	q_2

Figure 1: Transition Diagram for a binary string divisible by 3

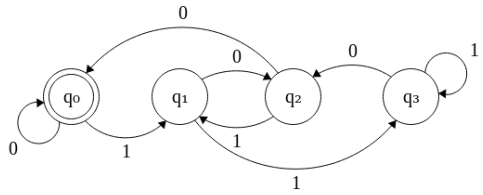


For a binary string divisible by 4

Transition Table :

	0	1
q_0	q_0	q_1
q_1	q_2	q_3
q_2	q_0	q_1
q_3	q_2	q_3

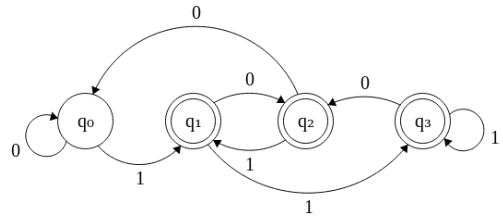
Figure 2: Transition Diagram for a binary string divisible by 4



If the above DFA is complemented we get this DFA(i.e reversal of initial and final states.)

For a binary string not divisible by 4

Figure3: Transition Diagram for a binary string not divisible by 4

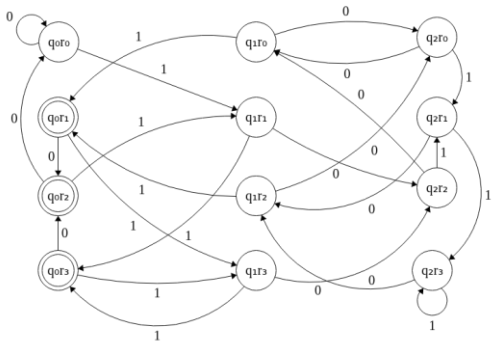


For a string divisible by 3 but not by 4

	0	1
q_0r_0	q_0r_0	q_1r_1
q_0r_1	q_0r_2	q_1r_3
q_0r_2	q_0r_0	q_1r_1
q_0r_3	q_0r_2	q_1r_3
q_1r_0	q_2r_0	q_0r_1
q_1r_1	q_2r_2	q_0r_3
q_1r_2	q_2r_0	q_0r_1
q_1r_3	q_2r_2	q_2r_3
q_2r_0	q_1r_0	q_2r_1
q_2r_1	q_1r_2	q_2r_3

q_2r_2	q_1r_0	q_2r_1
q_2r_3	q_1r_2	q_2r_3

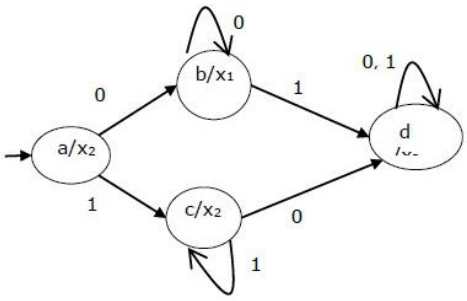
Illustration 4: Transition diagram for a binary string divisible by 3 but not by 4



The state table of a Moore Machine is shown below –

Present state	Next State		Output
	Input = 0	Input = 1	
→ a	b	c	x_2
b	b	d	x_1
c	c	d	x_2
d	d	d	x_3

The state diagram of the above Moore Machine is –



QUESTION-2

Define the finite state machines, viz., Moore and Mealy machines which produces output (apart from just accept or reject). Discuss equivalence between these two machines. Use a simple example whenever needed.

Finite automata may have outputs corresponding to each transition. There are two types of finite state machines that generate output −

- Mealy Machine
- Moore machine

A Moore machine can be described by a 6 tuple (Q, ∑, O, δ, X, q_0) where −

- **Q** is a finite set of states.
- ∑ is a finite set of symbols called the input alphabet.
- **O** is a finite set of symbols called the output alphabet.
- **δ** is the input transition function where δ: Q × ∑ → Q
- **X** is the output transition function where X: Q → O
- **q_0** is the initial state from where any input is processed (q_0 ∈ Q).

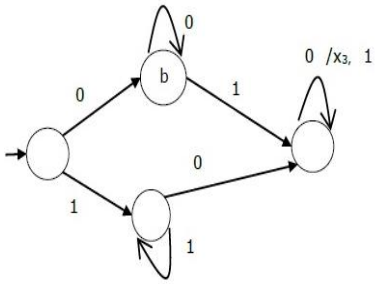
Mealy Machine

A Mealy Machine is an FSM whose output depends on the present state as well as the present input.

It can be described by a 6 tuple (Q, ∑, O, δ, X, q_0) where −

- **Q** is a finite set of states.
- ∑ is a finite set of symbols called the input alphabet.
- **O** is a finite set of symbols called the output alphabet.
- **δ** is the input transition function where δ: Q × ∑ → Q
- **X** is the output transition function where X: Q × ∑ → O
- **q_0** is the initial state from where any input is processed (q_0 ∈ Q).

The state diagram of the above Mealy Machine is –



Example:-
Input: 1101
Transition: δ (q0,1101)=> δ(q2,1)=>
δ(q2,0)=> δ(q1,1)=>q2
Output: 0011

QUESTION-3

State the Myhill–Nerode theorem (along with necessary definitions needed to make this statement clear). Discuss uses of this theorem with a simple example.

The Myhill Nerode Theorem states that for a language L such that L C Σ*, the following statements hold good :-

- There is a DFA that accepts L(L is regular)
- There is a right invariant equivalence relation ~ of finite index such L is a union of some of the equivalence classes of ~.
- L is of finite index.

The Myhill–Nerode theorem may be used to show that a language L is regular by proving that the number of equivalence classes of RL is finite. This may be done by an exhaustive case analysis in which, beginning from the empty string, distinguishing extensions are used to find additional equivalence classes until no more can be found.

Example:
The language consisting of binary representations of numbers that can be divided by 4 is regular. Given the empty string, 00 (or 100), 01 , 11 and 10 are distinguishing extensions resulting in the three classes (corresponding to numbers that give remainders 0, 1 , 2 and 3 when divided b
4), but after this step there is no distinguishing extension anymore. The minimal automaton accepting our language would have three states corresponding to these three equivalence classes.