Finite State Machine
Or
Finite Automata

Module 1

Syllabus

Module No.	Unit No.	Details	Hrs.	CO
1	**Finite Automata**		08	CO1
	1.1	Introduction: Alphabets, String, Language, Basic Operations on language, Concatenation, Kleene Star Introduction to different phases of compiler.		
	1.2	Finite. Automata (FA) -its behavior; DFA -Formal definition, state transition diagram, transition table, Language of a DFA. NFA -Formal definition, state transition diagram, transition table Language of an NFA. FA with epsilon-transitions, Eliminating epsilon-transitions, Equivalence of DFAs and NFAs, Conversion from NFA to DFA. Moore machine and Mealy Machine- Formal definition, state transition diagram, transition table, Conversion from Mealy to Moore machine and Moore to Mealy machine. Application of Finite Automata for Lexical Analysis and Lex tools		

Computational models

- CMs
- Real computers are quite complicated to allow us to set up a manageable mathematical theory for them directly
- Instead we can use an idealized computer called computational model in order to begin the theory of computation

Finite automata

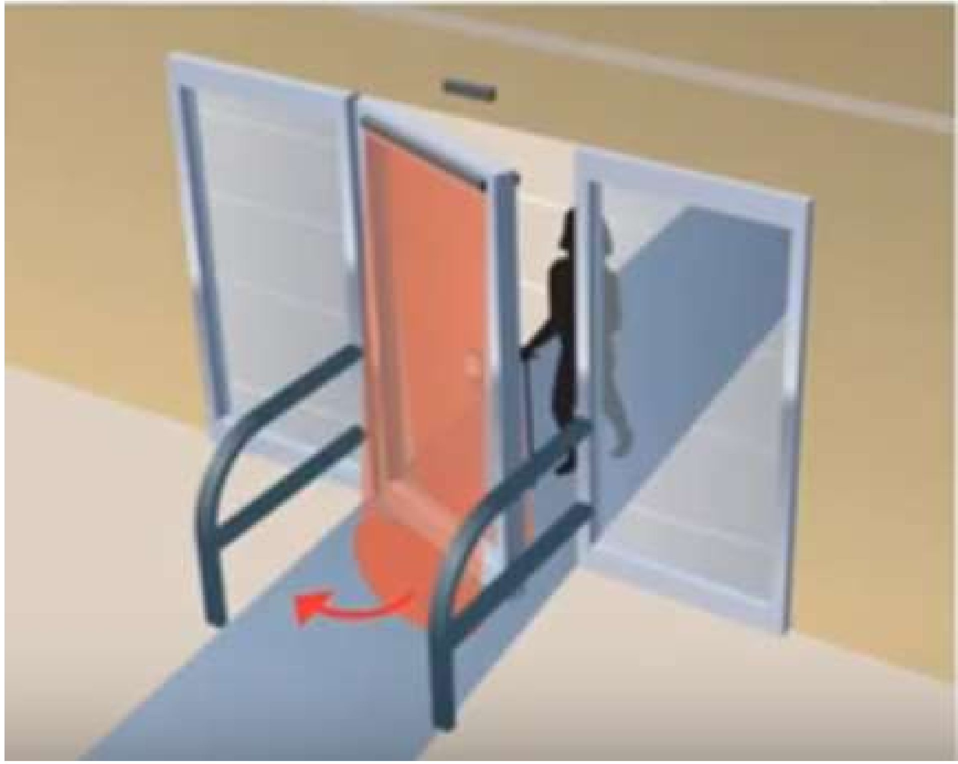
- The simplest computational model is called a finite state machine or a finite automaton
- FA are good models for computers with an extremely limited amount of memory

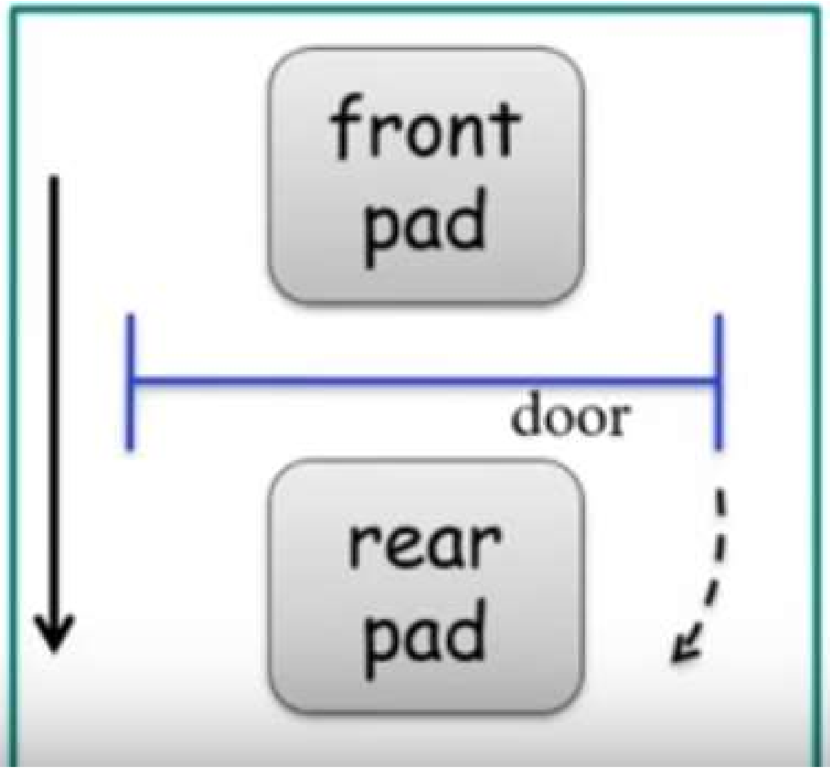
Finite automata

- Before developing the mathematics of finite automata we will examine the usage of a concrete finite automaton:
  - The controller of an automatic door

The controller of an automatic door

- Automatic doors are often found at supermarket entrances and exits
- An automatic door swing open when sensing that a person is approaching





The controller of an automatic door

- An automatic door is controlled by a simple automaton seen in Figure 1

- **One way door**

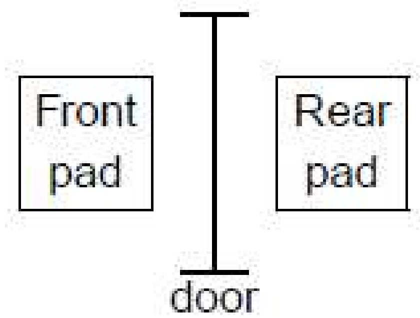
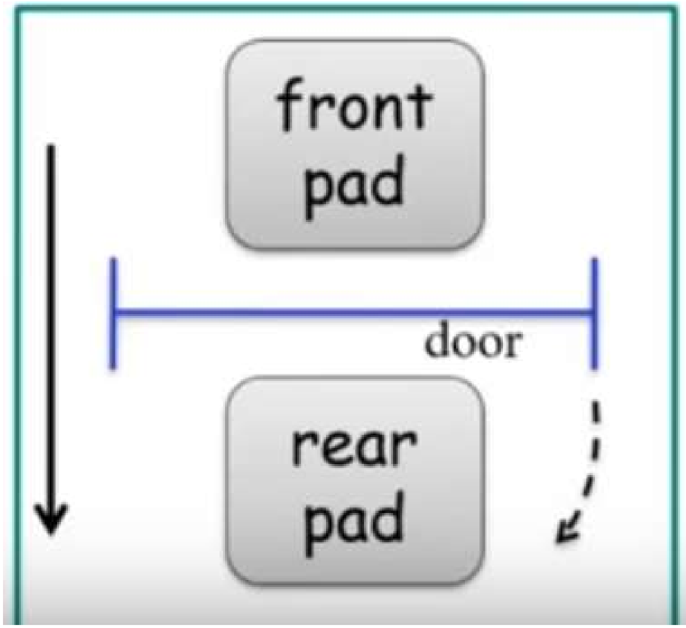


Figure 1: Controller of an automatic door

Behaviour

- An automatic door has a pad in front to detect the presence of a person about to walk through the door

- Another pad is located to the rear of the doorway so that the controller can hold the door open long enough for the person to pass all the way through



States of the controller

- The controller is in either of two states:
  - open or closed
  - representing the condition of the door

Inputs to the controller

- There are 4 possible input conditions:
  - front, meaning that a person is standing on the front pad
  - rear, meaning that a person is standing on the rear pad
  - both, meaning that people are standing on both pads
  - neither, meaning that no one is standing on either pad

Interpretation

- When controller is in the state closed and the input is REAR or NEITHER the controller remains in the state closed.

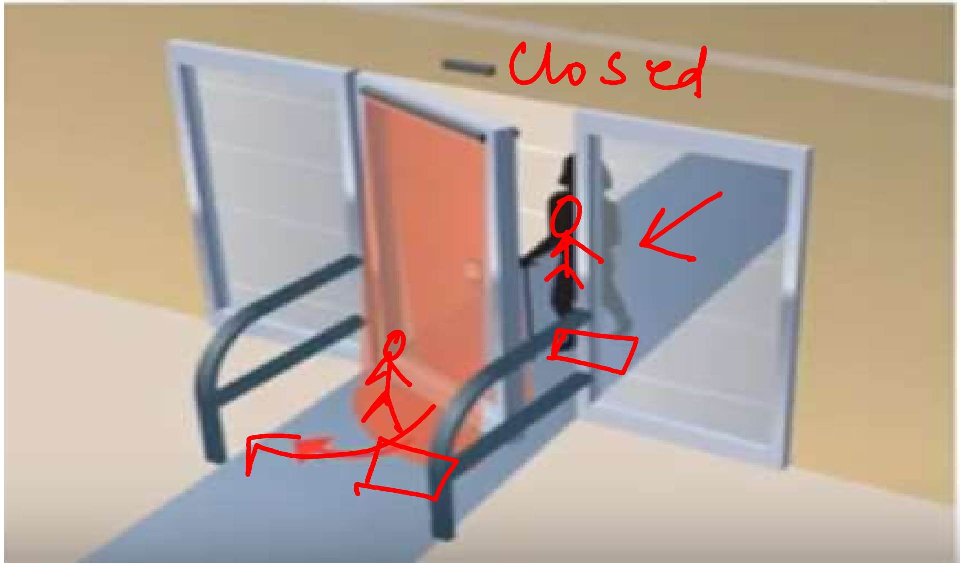
1

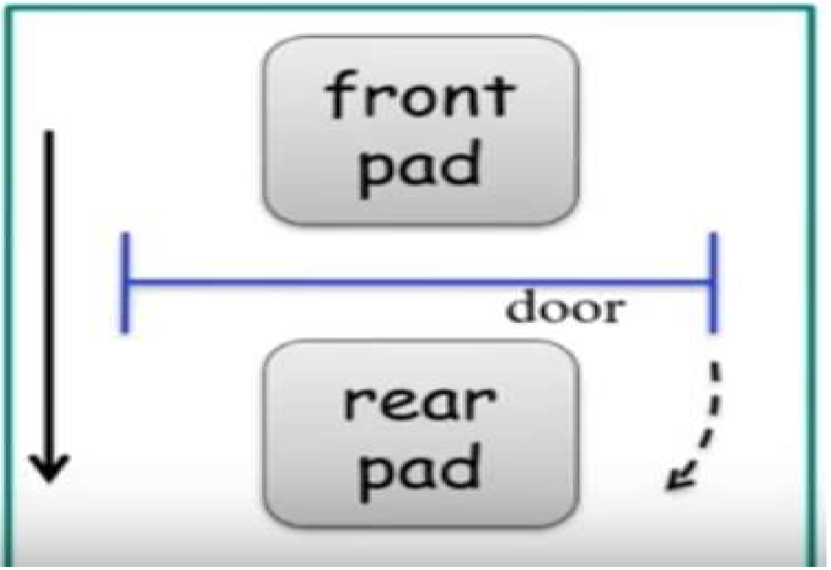
C

C

C

- In addition, If the input BOTH is received, it stays closed because opening the door risks knocking someone over the rear pad. But if input FRONT arrives, it moves to the open state

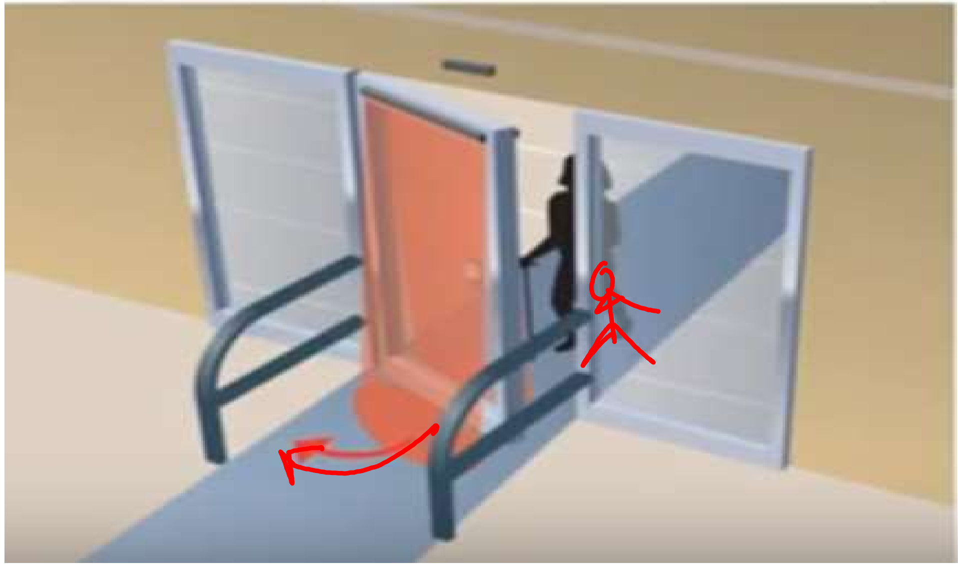


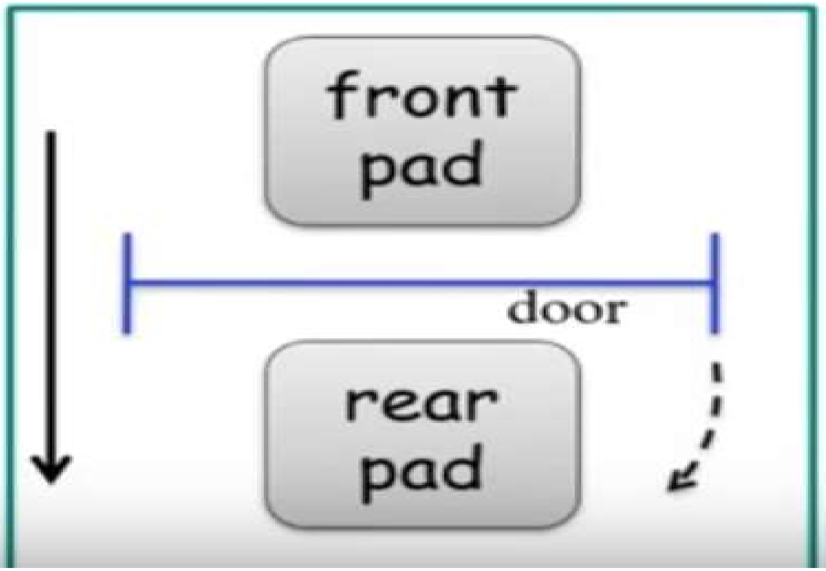


Interpretation

- When controller is in the state closed and the input is front the controller moves to the state open

O

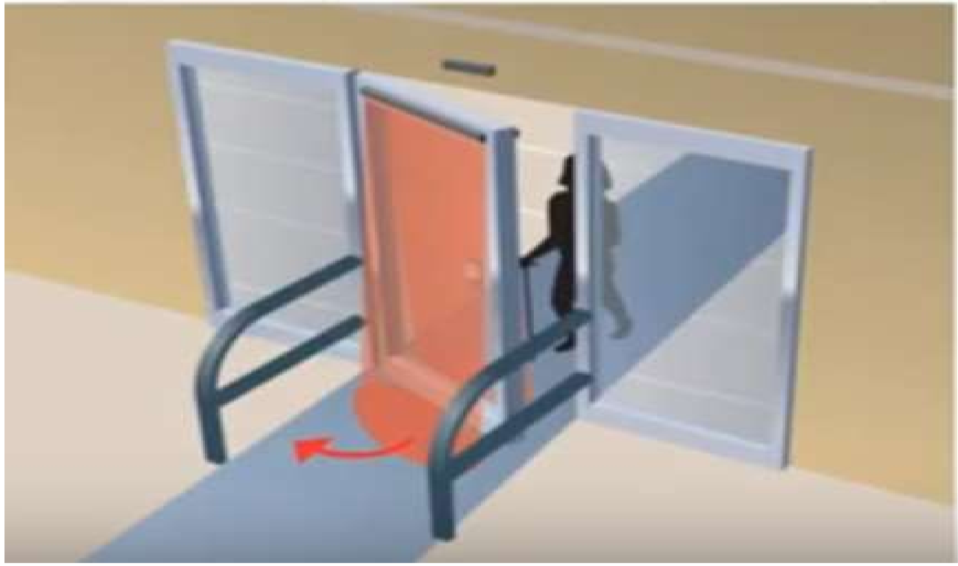


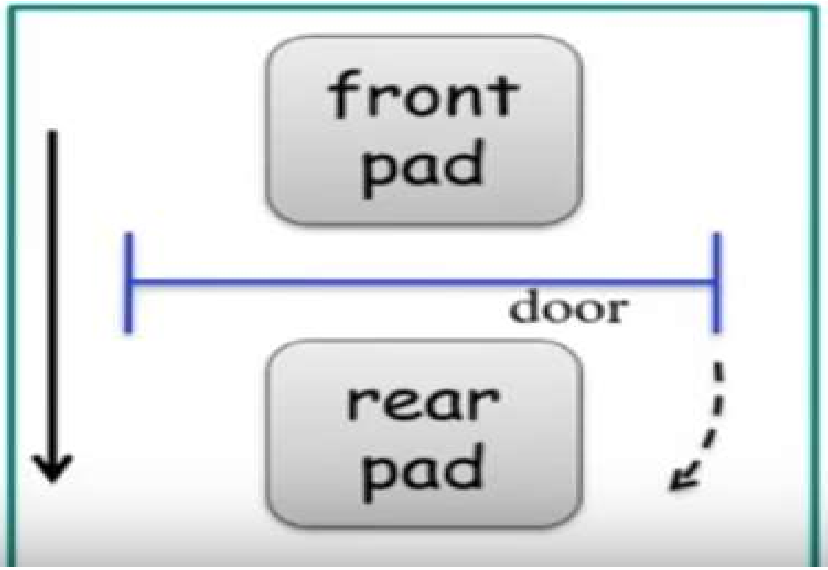


Interpretation

- **When the controller is in the state open**
- **Input is one of front, rear, both, the controller remains in the state open**

- When the controller is in the state open
- Input is neither the controller moves to the state closed



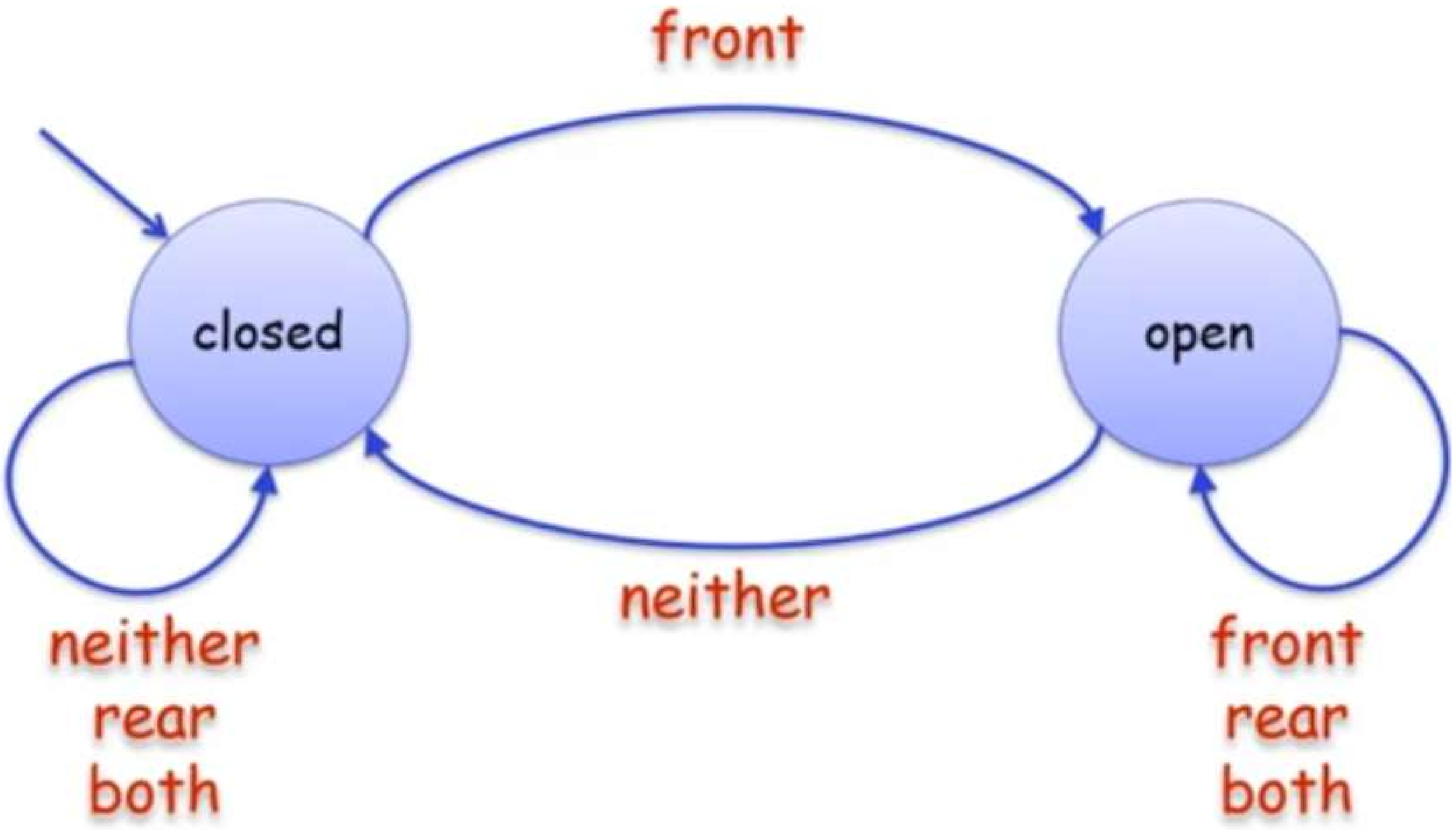


State transition diagram

- Depicts the movements of the controller from state to state depending upon the input it receives
- Start State=Closed

State transition diagram

- The state transition diagram of the controller,



State transition table

- The movement of the controller (and of the door) can also be represented by a table
- Rows are labelled by the states
- Columns are labeled by the input

Input signal

state		Neither	Front	Rear	Both
	Closed	closed	open	closed	closed
	Open	closed	open	open	open

Interpretation: T(state,input) = NewState

Exercise

- How should we change the state diagram if we have a sliding door?

- Try it Out!!

- This controller is a computer that has just a single bit of memory capable of recording the two states of the controller.

- Other common devices have larger memories.

Examples

- Other similar devices need controllers with larger memory.
- ??

Examples

Example-

- A state of the controller of an elevator may represent the floor the elevator is on and the inputs may be signals received from other floors

Examples of FSM

Example-

- Controllers of various household appliances such as
  - dishwashers,
  - electronic thermostats ,
  - parts of digital watches,
  - calculators
  - are computers with limited memories.

Examples of FSM

- A switching circuit, such as the control unit of a computer.
- A switching circuit is composed of a finite number of gates, each of which can be in one of two conditions, usually denoted 0 and 1.
- The state of a switching network with n gates is thus any one of the 2^n assignments of 0 or 1 to the various gates

Examples of FSM

- Although the voltages on each gate can assume any of an infinite set of values, the electronic circuits are so designed that only two of the voltages corresponding to 0 and 1 are stable and other voltages instantaneously adjust themselves to one of these voltages.

- Thus, It a FSM that separated the logical design of a computer from the electronic implementation

Design a FSM for JK Flip Flop

FSM for JK Flip Flop

TRUTH TABLE FOR JK F/F

J	K	QN	QN+1
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

NO CHANGE

J VALUE FORWARDED

TOGGLE CASE

FSM for JK Flip Flop

TRUTH TABLE FOR JK F/F

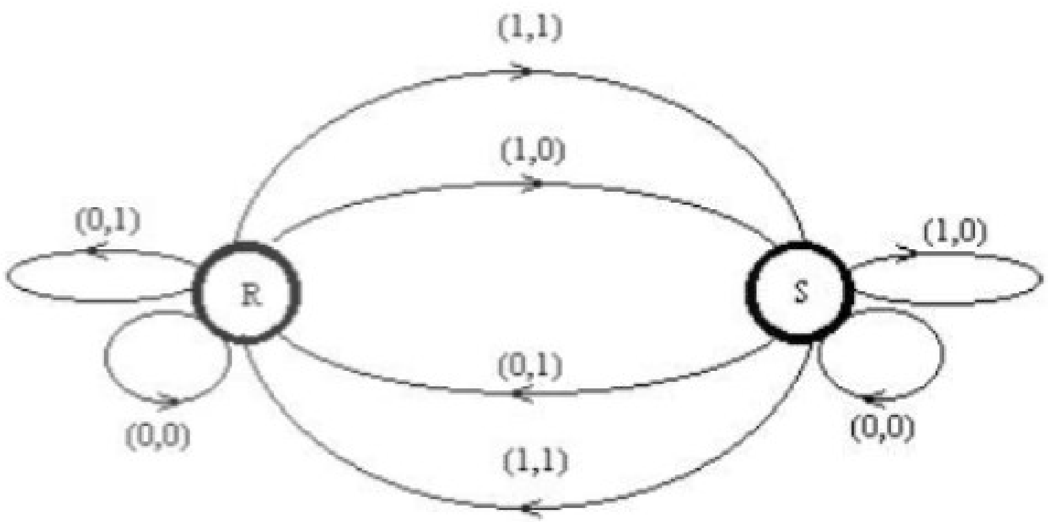
J	K	QN	QN+1
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

NO CHANGE

J VALUE FORWARDED

TOGGLE CASE

State Diagram for JK Flip Flop.



Examples of FSM

- Certain commonly used programs such as text editors and the lexical analyzers found in most compliers are often designed as finite state systems.

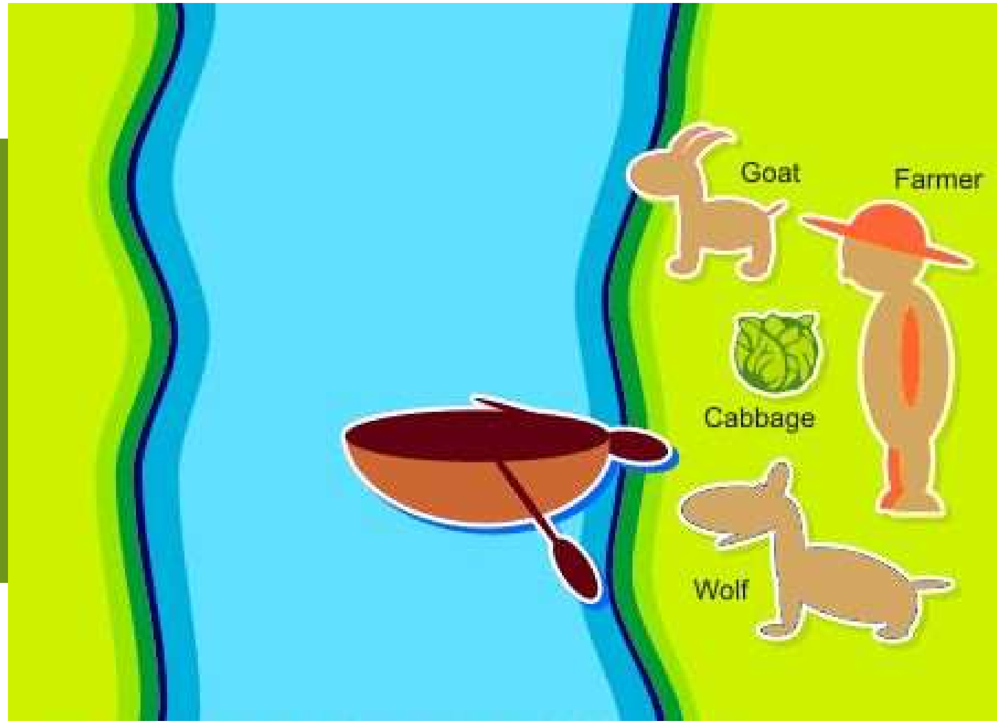
Examples of FSM

- For example, a lexical analyzer scans the symbols of a computer program to locate the strings of characters corresponding to
- **identifiers,**
- **numerical constants,**
- **reserved words**, and so on.

- In this process the lexical analyzer needs to remember only a finite amount of information,
  - **such as how long a prefix of a reserved word it has seen since startup.**

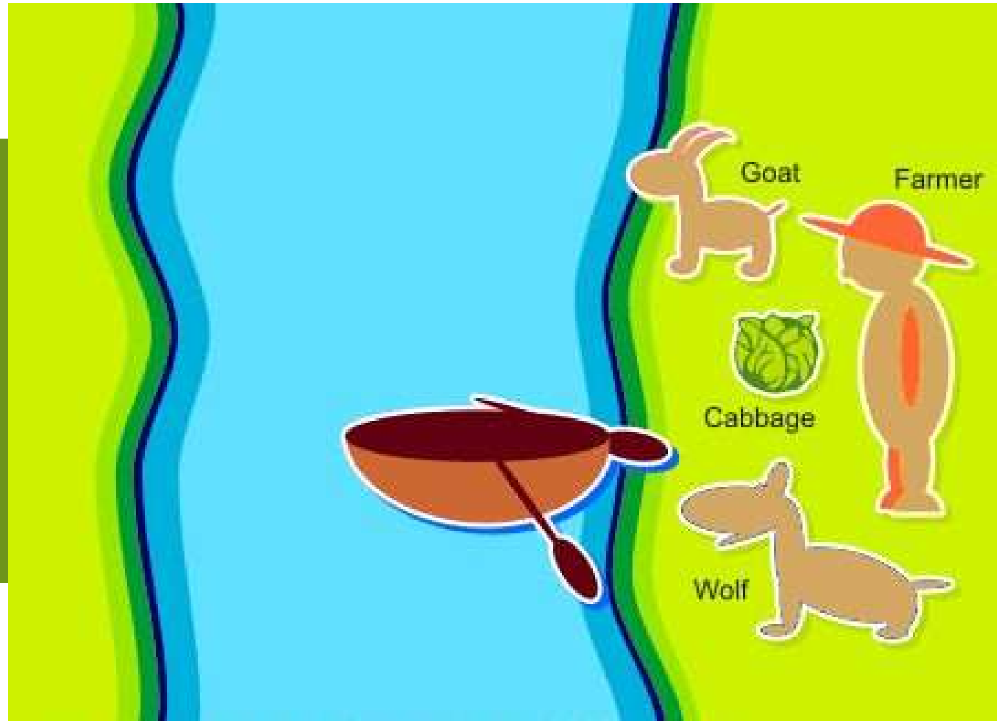
Wolf, Goat, and Cabbage problem

- A man with a wolf, goat, and cabbage is on the left bank of a river.
- There is a boat large enough to carry the man and only one of the other three.
- The man and his entourage wish to cross to the right bank, and the man can ferry each across, one at a time.



Wolf, Goat, and Cabbage problem

- If the man leaves the wolf and goat unattended on either shore, the wolf will surely eat the goat.
- If the goat and cabbage are left unattended, the goat will eat the cabbage.
- Is it possible to cross the river without the goat or cabbage being eaten?



Wolf, Goat, and Cabbage problem

- The problem is modelled by observing the occupants of each bank after a crossing.
- There are 16 subsets of the man (M), wolf (W), goat (G), and cabbage (C).

Wolf, Goat, and Cabbage problem

- States are labelled by hyphenated pairs
- **Left Bank Occupants-Right Bank Occupants**
- Symbols on left of the hyphen=subset on the left bank
- Symbols on right of the hyphen=subset on the right bank
- Eg: MG-WC

Wolf, Goat, and Cabbage problem

Fatal Combination-

- GC-MW
- GW-MC
- These are fatal and may never be entered by the system

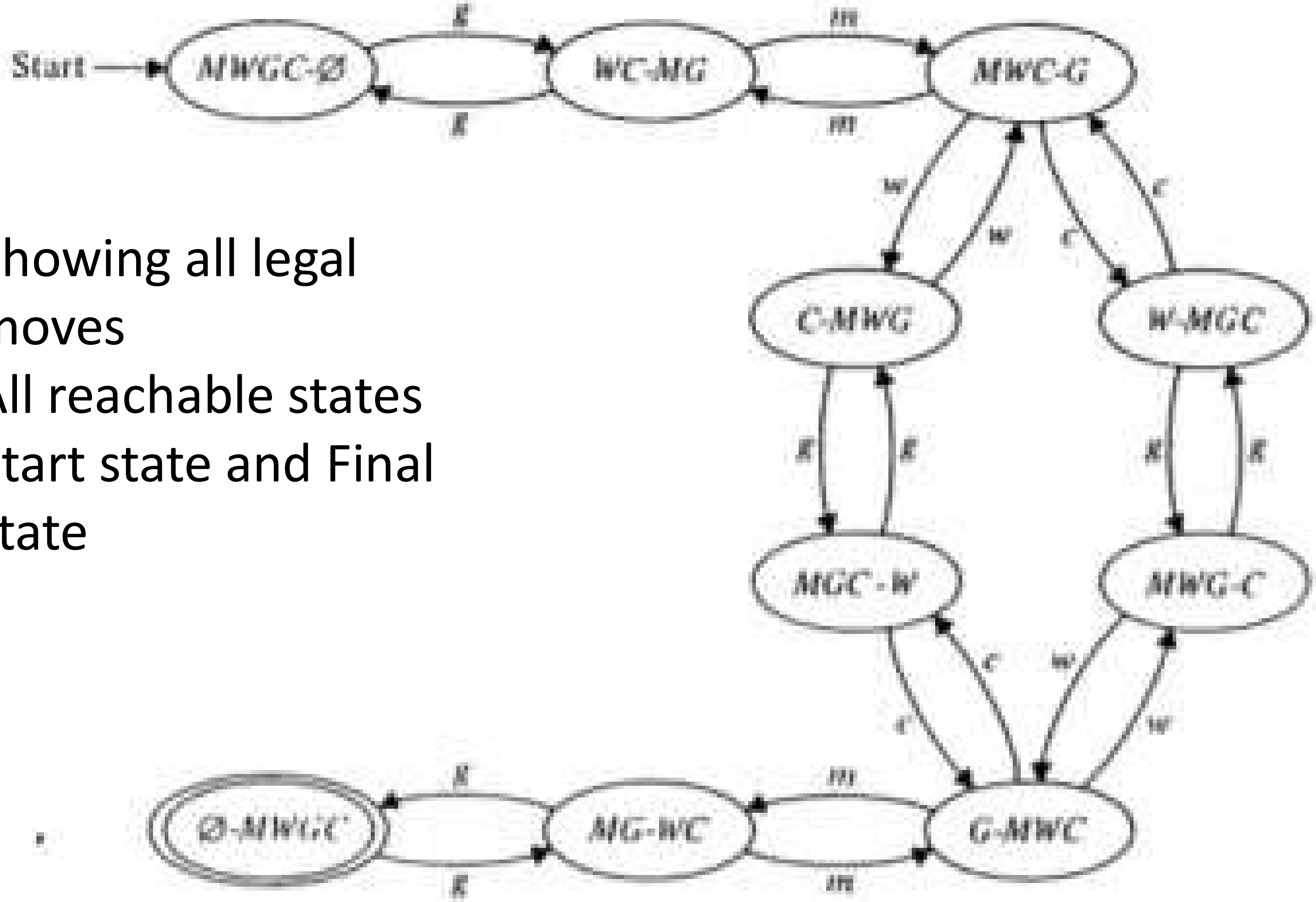
Wolf, Goat, and Cabbage problem

- The inputs are the action that the man takes
- Man crosses alone=Input m
- Man crosses with wolf=w
- Man crosses with goat=g
- Man crosses with cabbage=c

- Initial state=MWGC-ø
- Final state=ø-MWGC

Wolf, Goat, and Cabbage problem

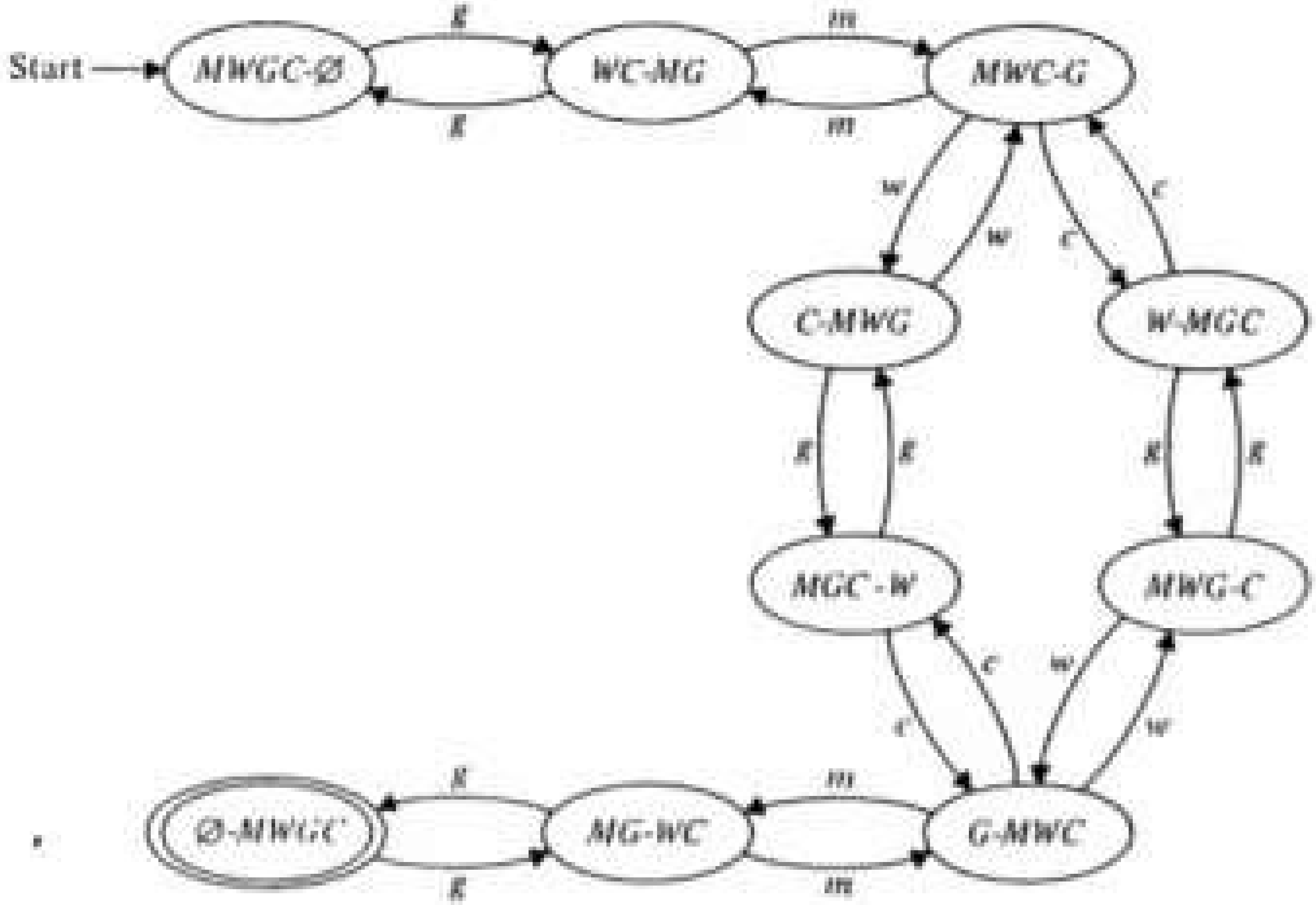
- State Transition Diagram-



- Showing all legal moves
- All reachable states
- Start state and Final state

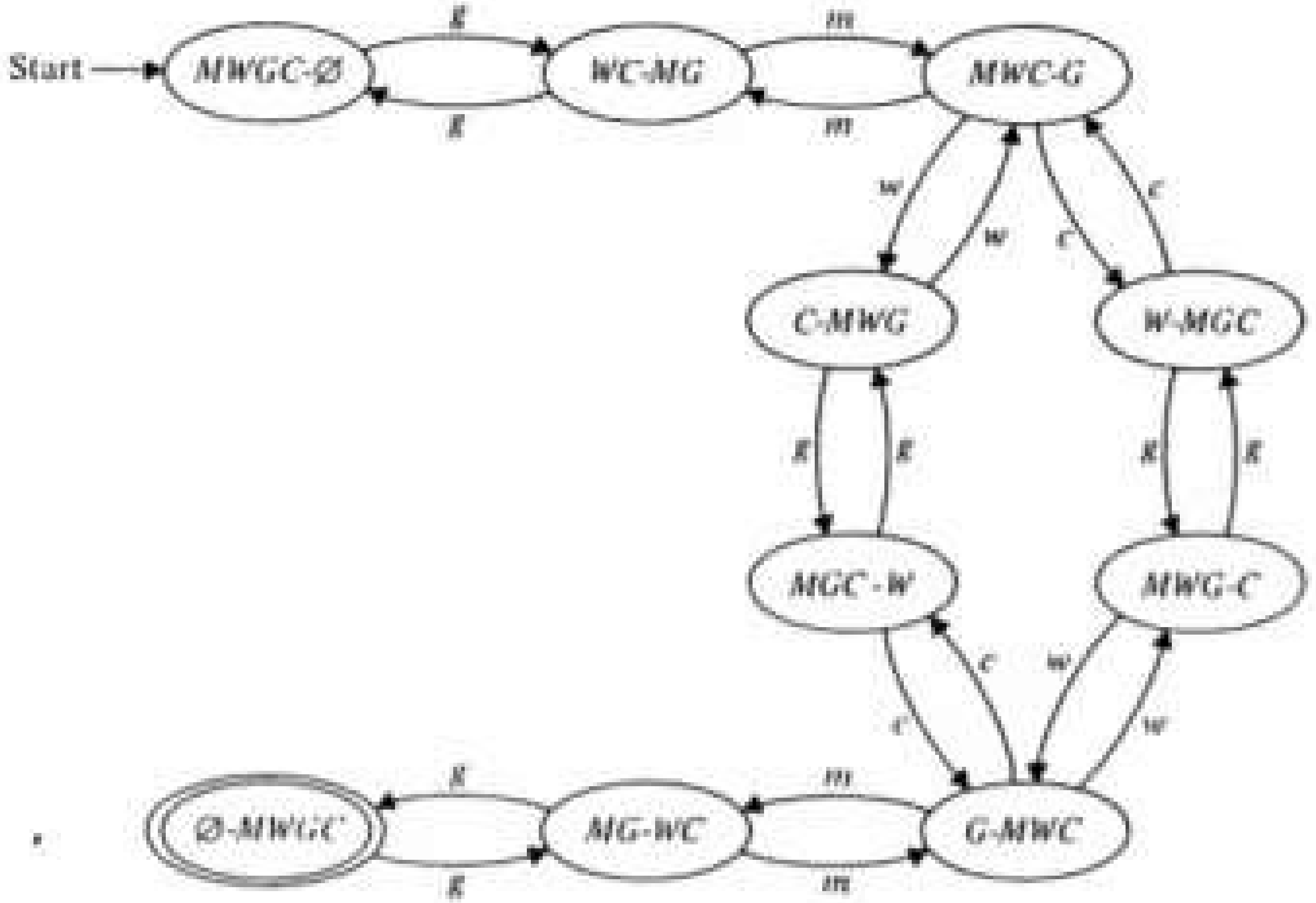
Wolf, Goat, and Cabbage problem

- There are **two equally short solutions** to the problem, as can be seen by searching for paths form the initial state to the final state.



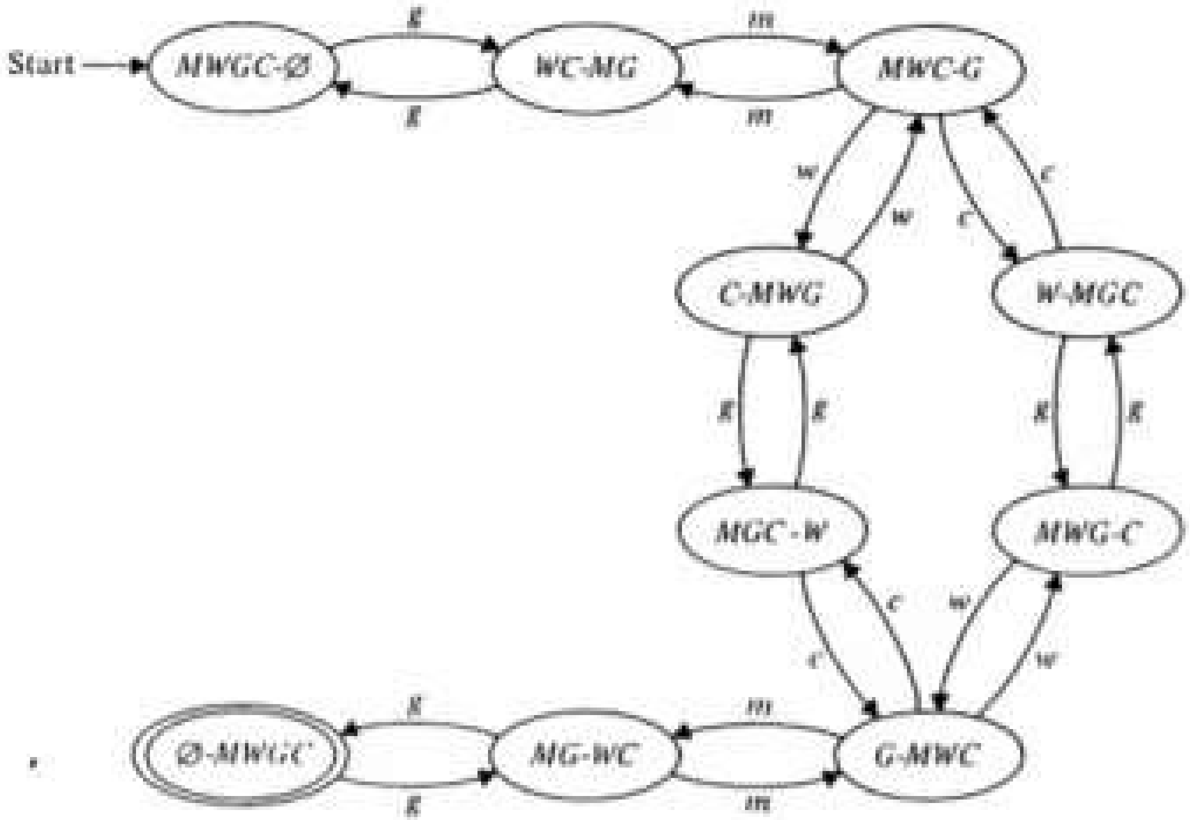
Wolf, Goat, and Cabbage problem

- Then a sequence of moves is a string,
- Such as the solution **gmwgcmg and gmcgwmg**



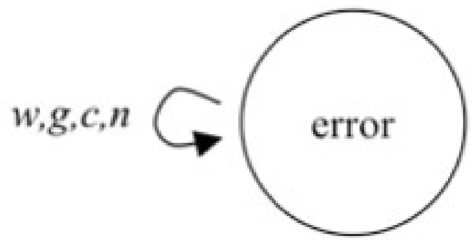
Wolf, Goat, and Cabbage problem

- What happens if we try a string that is not in the language?
- Consider *gmwm…*we get stuck
- with nowhere to go.



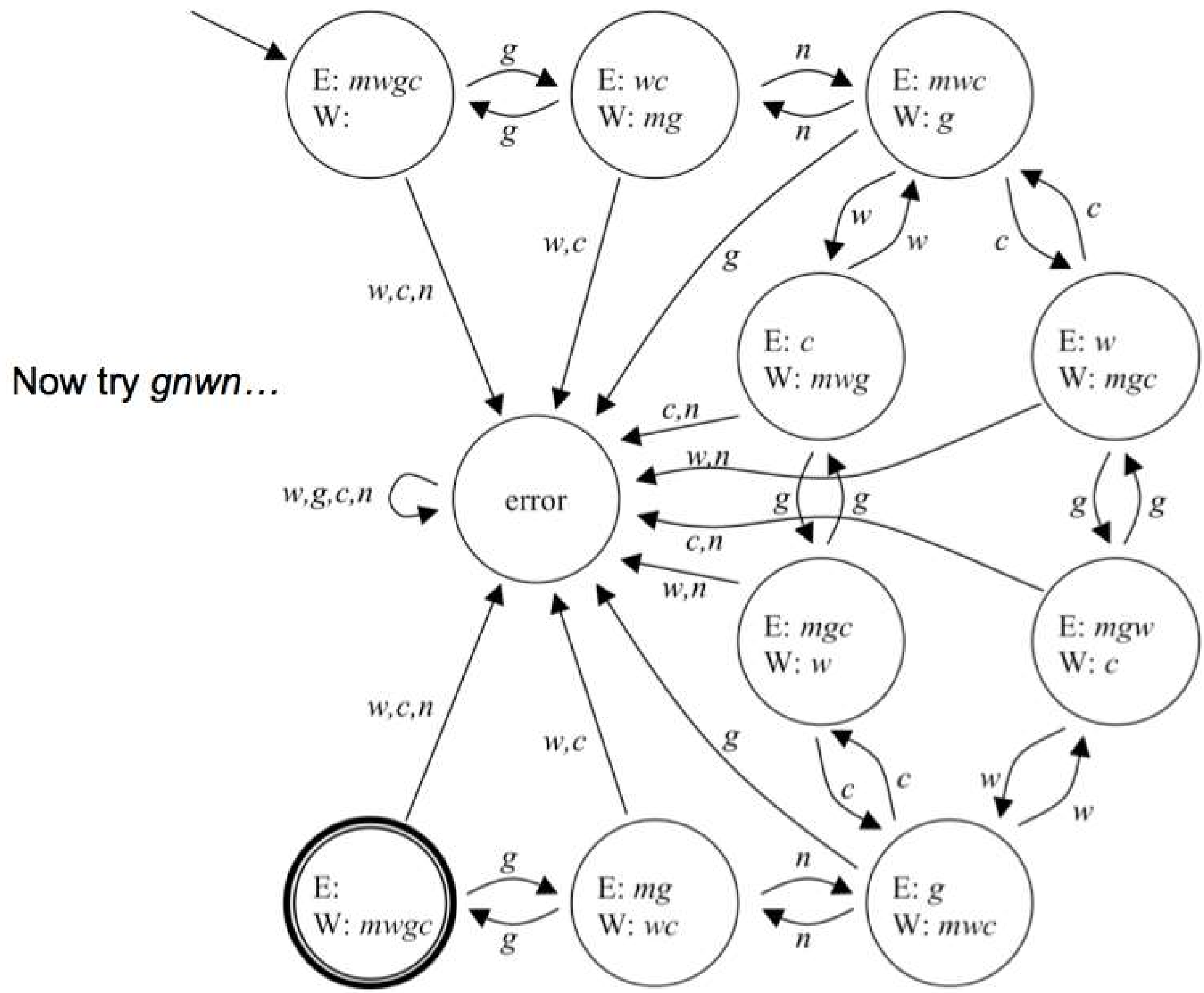
Wolf, Goat, and Cabbage problem

- On many strings that are not solutions, the previous diagram gets stuck
- Automata that never get stuck are easier to work with
- We'll need one additional state to use when an error has been found in a solution



Wolf, Goat, and Cabbage problem

- Here n=m



Set

- A group of objects represented as a unit
- Set may contain any type of object-numbers/symbols/other sets

-Michael Sipser

Set

- Well defined Collection of Objects and denoted as-
  1) Enumerating the member within { and }

     Eg-A={0,1,2}

  2) Using a set builder notation as:

     A={x| p(x)}

     A is set of all those x for which the predicate p(x) is true

     Eg- A set of all integers divisible by 3

     A={x| x is an integer and xmod3=0}

-Dr O.J Kakde

Alphabets

- An alphabet is a finite, non empty set of symbols
- Symbol for Alphabet= Σ
- $\Sigma=\{0, 1\}$, the **binary alphabet**, probably the most important alphabet in computer science.
- $\Sigma=\{a, b, c, ..., z\}$, The set of all lowercase letters

-Hopcraft

Strings

Strings-

- A finite sequence of symbols chosen from some Alphabet
- Eg- 01101 is a string from ∑={0, 1} , the binary alphabet

Empty String-

- String with zero occurrences of symbols
- Denoted by ε
- A string that may be chosen from any alphabet whatsoever.

-Hopcraft

Strings

Length of a String-

- Number of positions of symbols

Eg- 01101 has length 5

5 positions for symbols

But no of symbols are only 2, i.e. 0 and 1

- Denoted as |w|
- |011|=3
- | ε|=0

-Hopcraft

Strings

Concatenation of Strings-

- Let x and y be strings
- xy =Concatenation of x and y
- String formed by copy of x and following it by a copy of y

- If x=a1a2……ai
- If y=b1b2…..bj
- xy=a1a2….aib1b2….bj where length of string is i+j

-Hopcraft

Strings

Powers of an alphabet:

- Σ is an alphabet ,
- We can express the set of all strings of a certain length from that alphabet by using an exponential notation
- Σ^k=set of strings of length k, each of whose symbol is in Σ
- Σ^0= ε, no matter what alphabet is in it, ε is the only string of length 0

$\Sigma^{\text{exp}\ k}$ $\quad \Sigma^0 = \varepsilon$

-Hopcraft

Strings

Powers of an alphabet:

- If $\sum=\{a,b,c\}$ then
- $\sum^1=\{a,b,c\}$
- $\sum^2=\{aa,ab,ac,ba,bb,bc,ca,cb,cc\}$ and so on

Difference between $\sum$ and $\sum^1$

- $\sum$=alphabet with members a,b,c as symbols
- $\sum^1$=set of strings with members as the strings a,b and c of length 1

-Hopcraft

Strings

Powers of an alphabet:

$\sum^*$ =Set of all strings over an alphabet $\sum$

$\sum^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup ....$

Excluding Empty String ε, as $\Sigma^0 = \varepsilon$

$\sum^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup ....$

$\sum^* = \Sigma + \cup \{\varepsilon\}$

-Hopcraft

Language

- **Set of string formed by using the symbols belonging to some particularly chosen alphabet**

- **Common languages can be viewed as a set of strings**

- -Hopcraft

Language

- **English Language , the collection of legal English words is a set of strings over the alphabet that consists of all the letters**

- **C or any other programming Language, where the legal programs are a subset of the possible strings that can be formed from the alphabet of the language.**
  - **The alphabet is a subset of the ASCII characters-generally uppercase lower case letters,digits,punctuation and mathematical symbols.**

    - -Hopcraft

Language

Some abstract examples-

- The set of strings of 0s and 1s with an equal no of each-

{ε,01,10,0011,0101,1001,…..}

- The set of binary numbers whose value is prime:

{10,11,101,111,1011,…..}

Set Former Notation-

- {w|w consists of an equal number of 0s and 1s}

- <u>{w|w is a binary integer that is prime}</u>

- -Hopcraft

Operation on Languages

Union-

- Union of Two Languages L & M,
- Denoted as L U M
- The set of strings that are either L or M or both
- Eg-If L={001,10,111}, M={ε,001}
- L U M ={ε,10,001,111}

-Hopcraft

Operation on Languages

Concatenation-

- Concatenation of Languages L and M
- The set of strings that can be formed by taking any string in L and concatenating it with any string in M
- Eg-If L={001,10,111}, M={ε,001}
- L.M=LM={001,10,111,001001,10001,111001}

-Hopcraft

Finite Automata:

- FA is the mathematical model (or formalism) of an FSM
- The finite automaton is a mathematical model of a system, with discrete inputs and outputs.
- Mathematical Models are abstractions and simplifications of how certain machines actually work
- Help us understand specific properties of these machines

Finite Automata:

- The system can be in any one of a finite number of internal configurations or "states".

- The state of the system summarizes the information concerning past inputs that is needed to determine the behaviour of the system on subsequent inputs

Finite Automata: Formal Definition

- A list of five elements is called a 5-tuple,

- A finite automaton can be defined as a 5-tuple

Finite Automata: Formal Definition

FA is denoted by 5 Tuple:

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

1) Q = set of states

2) $\Sigma \Rightarrow$ Input alphabet

3) $\delta \Rightarrow$ Transition fn

4) $q_0 \Rightarrow$ Initial / start state

5) $F \Rightarrow$ Set of final State

Finite Automata: Formal Definition

FA is denoted by 5 Tuple: M=(Q, ∑, δ, q_0, F) where

- Q:Finite set of states
- ∑:Finite input alphabet
- δ:Q X ∑ ->Q   State Transition function(STF)
- q_0: Initial/Start state of FA, q0 □ Q
- F: Set of Final States/ Accept State, F □ Q

Finite Automata: Formal Definition

FA is denoted by 5 Tuple: $M=(Q, \sum, \delta, q_0, F)$ where

- This definition is for Deterministic FA or DFA
- The Transition Function for Non- Deterministic FA or NDFA is different
- NDFA will be discussed later

State Transition function(STF)

- Denoted by δ

- Define rules of moving

- We can denote the transition rules by a function called the transition function,

- δ : States × Alphabet → States

State Transition function(STF)

- If the FA has an arrow from a state x to state y labelled with input symbol 1
- If Automaton is in state x, when it reads 1, it then moves to state y
- $\delta(x,1)=y$
- Example: $\delta(q0, x) = q1$

Machine M1

- A FSM M1, that has 3 states, defined by the transition diagram in the figure 1

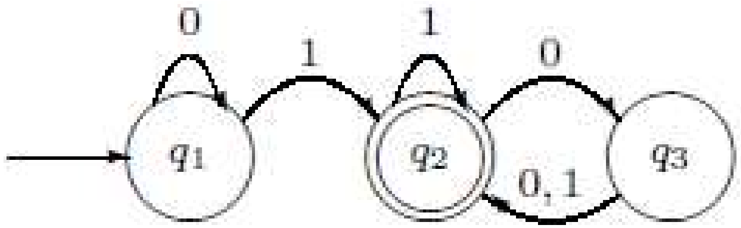


Figure 1: The finite automaton M_1

Machine M1

- **5 Tuple:-**

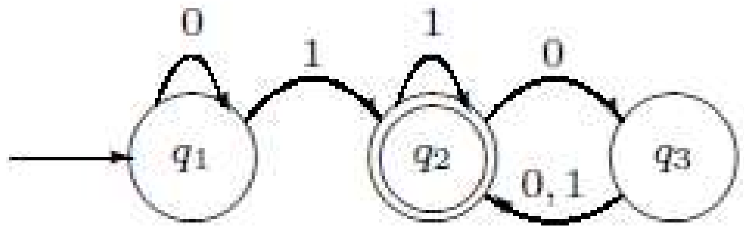


Figure 1: The finite automaton M_1

Formalizing M1

M1 = (Q, ∑, δ, q1, F) where

1. Q = {q1, q2, q3}
2. ∑ = {0, 1}
3. δ is described by the table:

δ	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

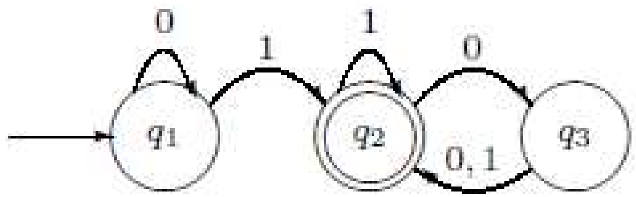


Figure 1: The finite automaton M_1

4. q1 is the start state,
5. F = {q2}

Acceptor/Rejector

- Upon reading the entire input string,
- If the machine resides in any of the final states then the input string is “Accepted” by the Machine
- If the machine resides in any of the non-final states then the input string is “Rejected” by the Machine
- FA acts as an Input Acceptor or Rejector

Language of a DFA

- If A is the set of all strings that a machine M accepts,
- We say that A is the language of the machine M and write L(M) = A.

Terminology

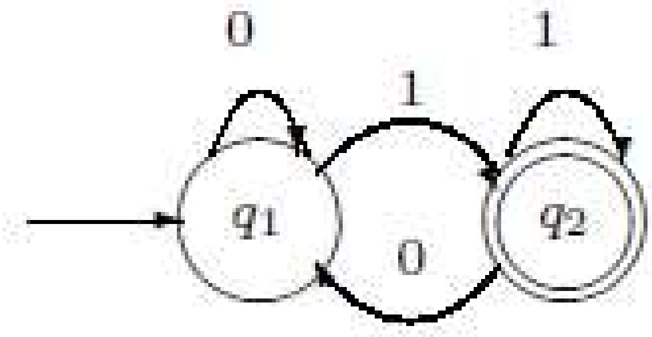
- In order to avoid confusion:
- Use accept when we refer to strings
- Use recognize when we refer to languages

Language of a DFA

- A machine may accept several strings, but it always recognizes only one language
- If a machine accepts no strings, it still recognizes one language, namely the empty language □

Machine M2

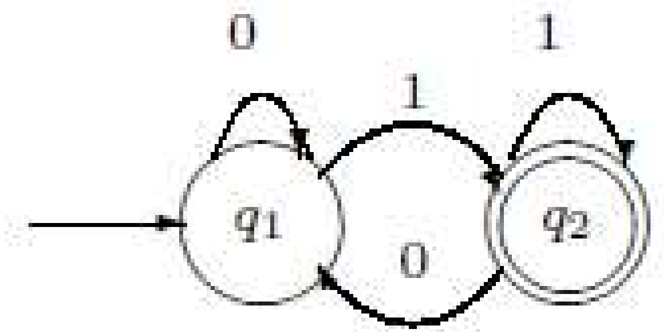
The state diagram in Figure 2 describes a machine M2



Write the 5 Tuple representation
Try finding the language of the string

Machine M2

5-Tuple Representation

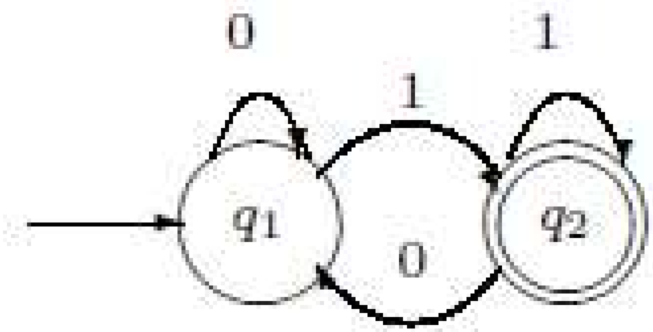


M2 = ({q1, q2}, {0, 1}, δ, q1, {q2}) where
δ(q1, 0) = q1, δ(q1, 1) = q2, δ(q2, 0) = q1, δ(q2, 1) = q2

Machine M2-Acceptor/Rejector

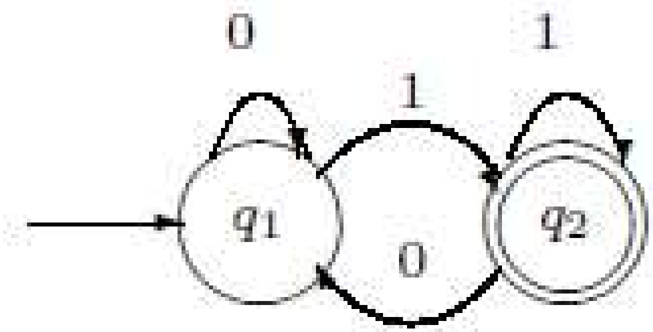
- A good way of understanding any machine is to try it on some sample input string

Machine M2-Acceptor/Rejector



String “1101”-Accepted/Rejected?

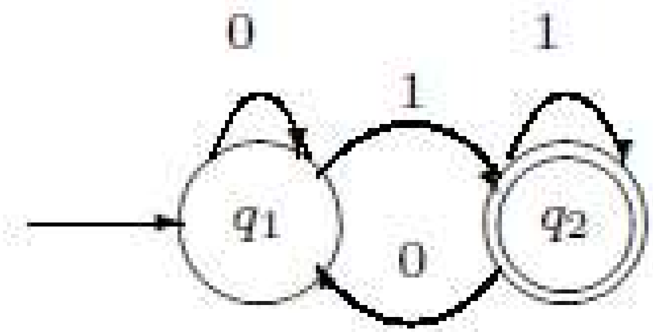
Machine M2-Acceptor/Rejector



String “1101”

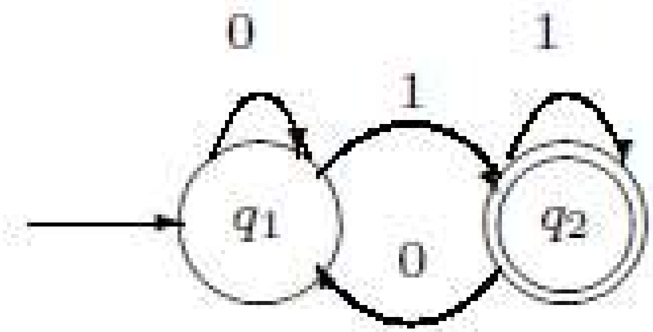
- Machine M2 starts in q1
- From q1,reads 1=>q2
- From q2 reads 1=>q2
- From q2 reads 0=>q1
- From q1 reads 1=>q2
- String ends here and q2 is Final state
- String is Accepted

Machine M2-Acceptor/Rejector



String “110”-Accepted/Rejected?

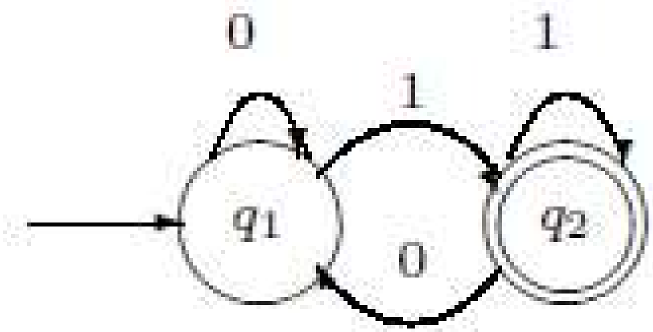
Machine M2-Acceptor/Rejector



String "110"

- Machine M2 starts in q1
- From q1,reads 1=>q2
- From q2 reads 1=>q2
- From q2 reads 0=>q1
- String ends here and q1 is Non-Final state
- String is Rejected

Language of DFA M2



- L(M2) = {w| w ends in a 1}
- M2 accepts all strings that end in a 1

Machine M3

Consider the finite automaton M3 in Figure 3

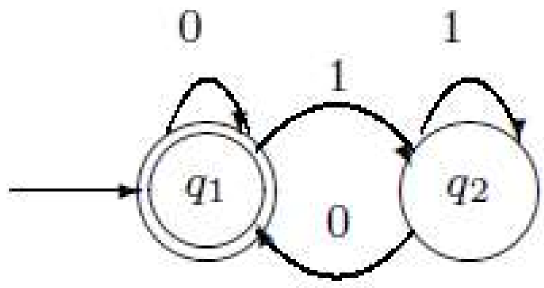
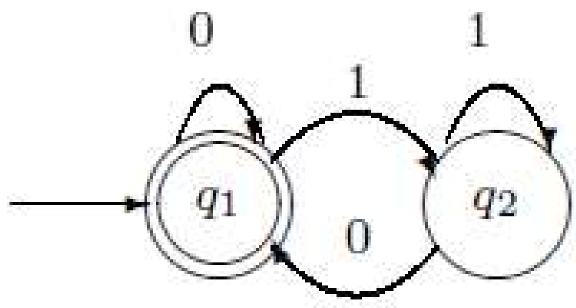


Figure 3: State diagram of the finite automaton M_3

- M3 is similar to M2, except for the location of the accept state
- Start state is also an accept state

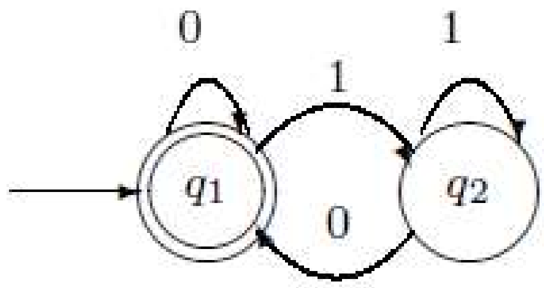
Machine M3-Acceptor/Rejector



State diagram of the finite automaton M_3

- As usual, M3 accepts all strings that leave it in an accept state when it has consumed the input.
- M3 also accepts the empty string ϵ.
- As soon as M3 begins the reading of ϵ. it is at the end,
- So if Start state is an Accept state, it accepts it

Language of DFA M3



State diagram of the finite automaton M_3

- M3 also accepts any string ending with a Zero
- L(M3) = {w|w is the empty string or ends in 0}

Machine M4

Consider the five-state machine M4, Figure 4

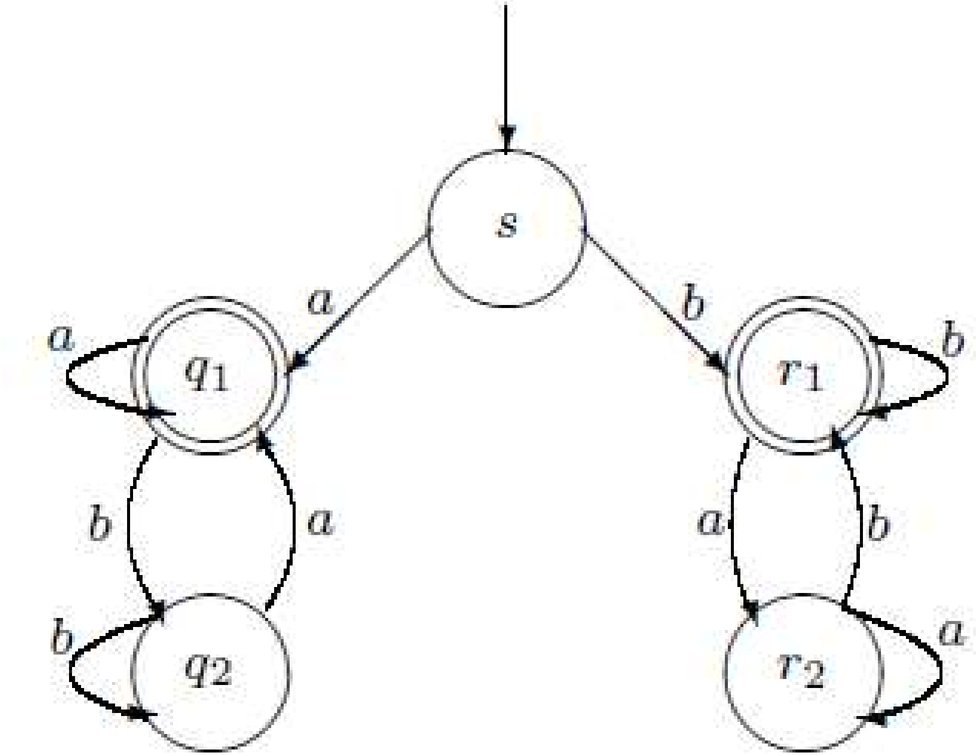


Figure 4: Finite automaton M_4

Machine M4-Acceptor/Rejector

- M4 has two accept states, q1 and r1
- M4 operates over the alphabet ∑ = {a, b}

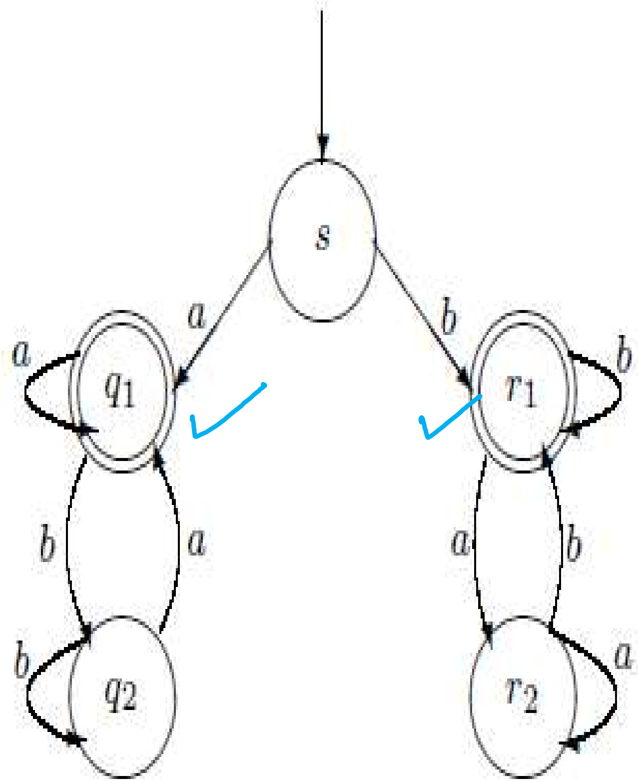


Figure 4: Finite automaton M_4

Language of DFA M4

- M4 begins in state s and after it reads the first symbol in the input it **either goes to the left to q states or to the right to r states**

- Once M4 goes to the left or to the right,**it can never return to the start state**

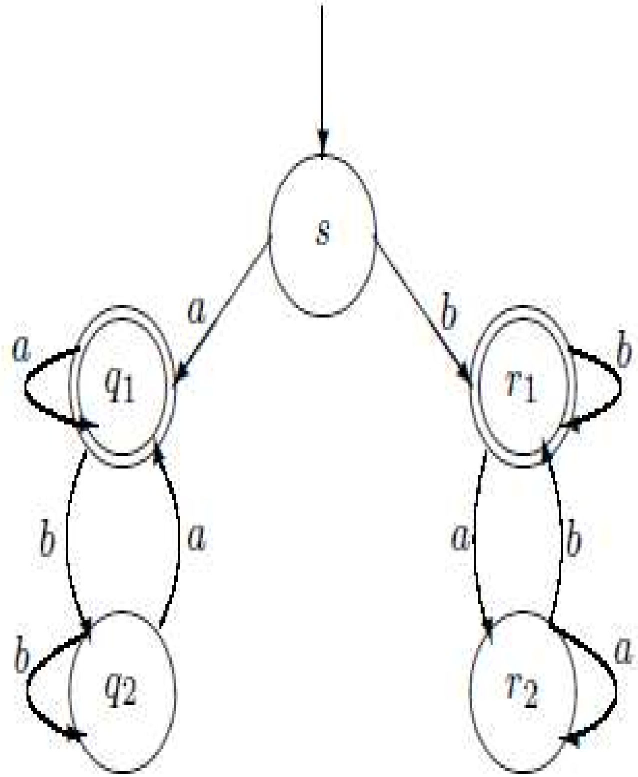


Figure 4: Finite automaton M_4

Language of DFA M4

Strings-

- a?
- aa?
- ab ?
- aba ?
- abbab ?
- b?
- bb?
- ba?
- bab?
- baaa?
- baabb?

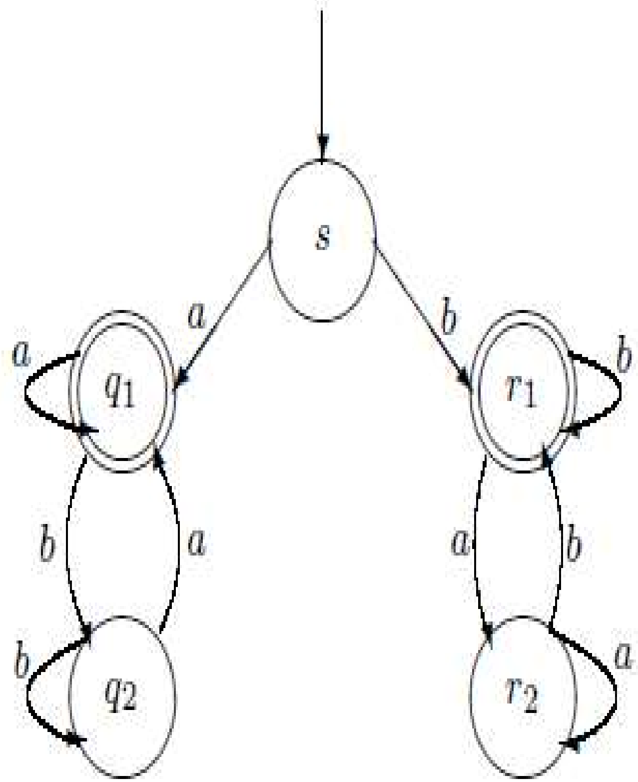


Figure 4: Finite automaton M_4

Language of DFA M4

- Some experimentation with M4 shows that -
  - Accepts strings as a, b, aa, bb, bab
  - Does not accept strings as ab, ba, bbba

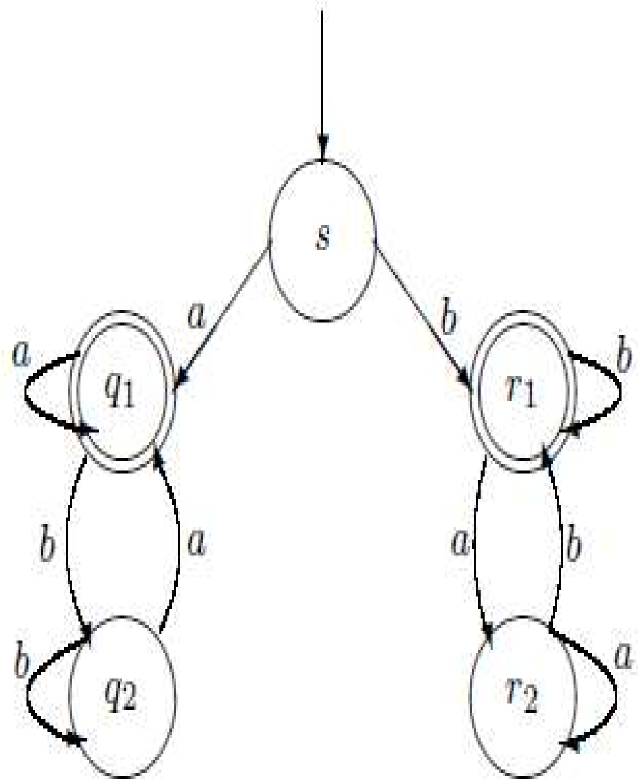


Figure 4: Finite automaton M_4

Language of DFA M4

- If the first symbol in the string is a then it goes to the left and accepts when the strings ends with an a

- If the first symbol in the string is b then it goes to the right and accept when the strings ends with a b

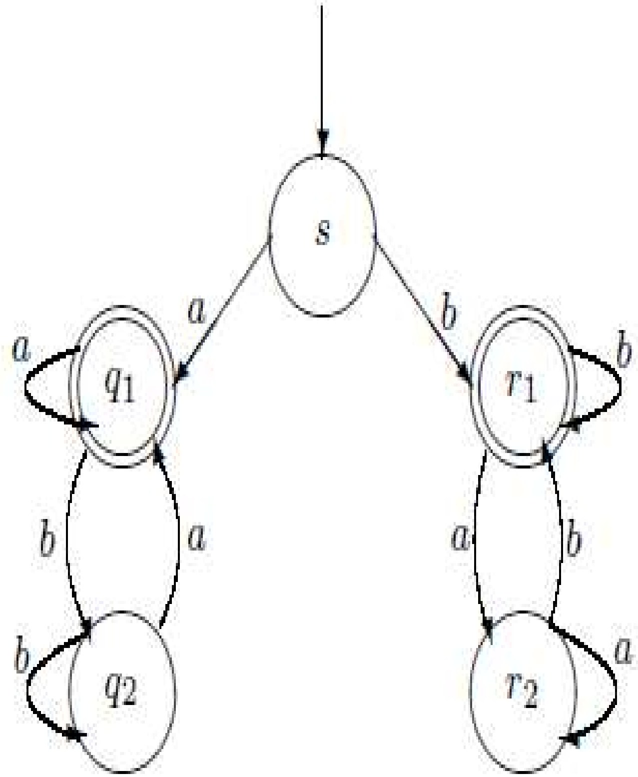


Figure 4: Finite automaton M_4

Language of DFA M4

- Accepts all strings that start and end with a or that starts and ends in b

- Conclusion: L(M4) = {w|w = axa} ∪ {w|w = byb} for x, y ∑*

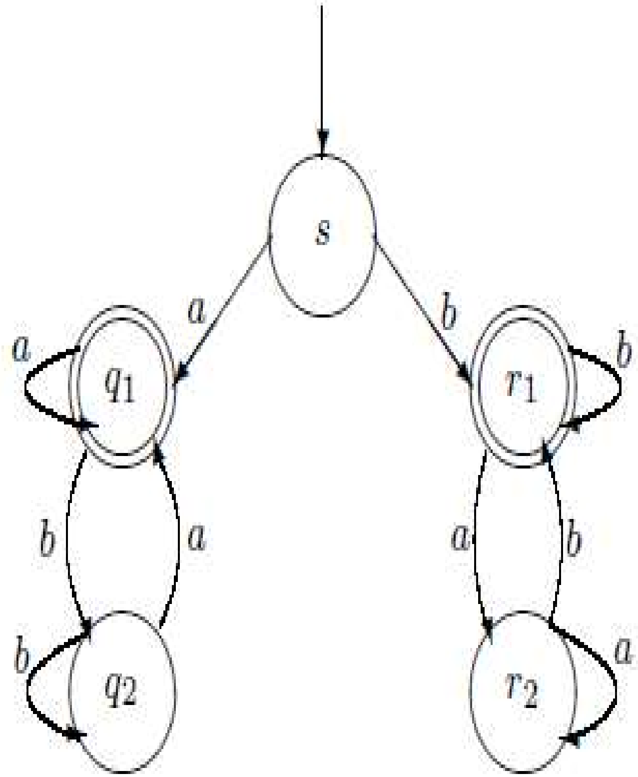


Figure 4: Finite automaton M_4

Exercise1 – Acceptor/Rejector

- Consider the transition table given below and check whether the string ‘110101’ lies in L(M) or not. Given Initial State=q0 and F={q0}.Also write the 5-Tuple representation.

	0	**1**
q0	q2	q1
q1	q3	q0
q2	q0	q3
q3	q1	q2

Exercise1 – Acceptor/Rejector

- Consider the transition table given below and check whether the string '110101' lies in L(M) or not. Given Initial State=q0 and F={q0}. Also write the 5-Tuple representation.

	0	1
q0	q2	q1
q1	q3	q0
q2	q0	q3
q3	q1	q2

Ans

$\delta(q_0, 110101) \vdash \delta(q_1, 10101)$

start

$\vdash \delta(q_0, 0101) \vdash \delta(q_2, 101)$

$\vdash \delta(q_3, 01) \vdash \delta(q_1, 1) \vdash q_0$

final state $\Rightarrow$ accepted

$M = (Q, \Sigma, \delta, \text{Initial state}, \text{Final state})$

1) $Q = \{q_0, q_1, q_2, q_3\}$

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

3) δ — already given in Q

4) Initial state $= q_0$

5) $F = \{q_0\}$

DESIGNING FINITE AUTOMATA

Design is a creative process!

DESIGNING FINITE AUTOMATA

Put yourself in the place of the machine you are trying to design.

DESIGNING FINITE AUTOMATA

Example 1

1) Suppose that the alphabet is {0,1} and that the language consists of all strings with **an odd number of 1s.**

2) You want to construct a finite automaton E1 to recognize this language.

DESIGNING FINITE AUTOMATA

Example 1

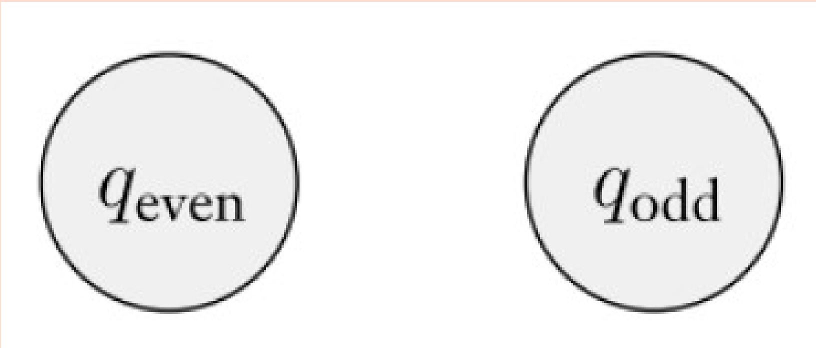
- Pretending to be the Automaton E1 recognize the language
- Start reading the input string of 0s and 1s symbol by symbol
- Simply remember whether the no of 1s seen so far is even or odd
- Keep track of this information
- Represent this information as finite set of possibilities

DESIGNING FINITE AUTOMATA

Example 1

- Possibilities so far would be-
  1) Even so far
  2) Odd so far

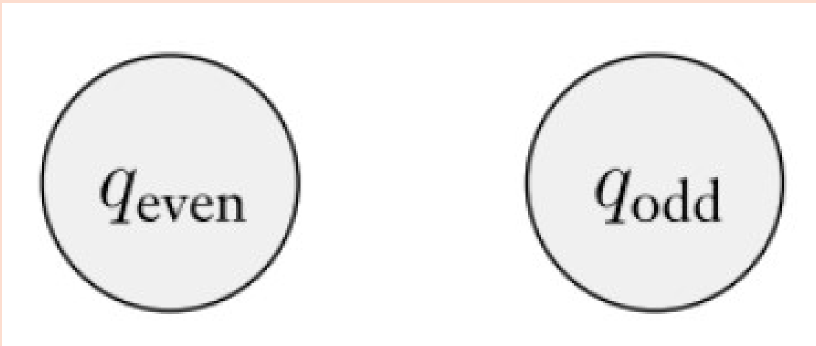
- **Assign states to these possibilities**



DESIGNING FINITE AUTOMATA

Example 1

- Assign the transitions by seeing how to go from one possibility to another upon reading a symbol

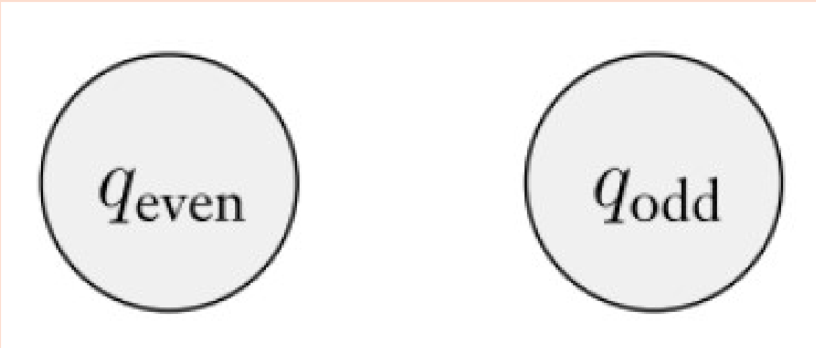


DESIGNING FINITE AUTOMATA

Example 1

- For Assigning the transitions by seeing how to go from one possibility to another upon reading a symbol

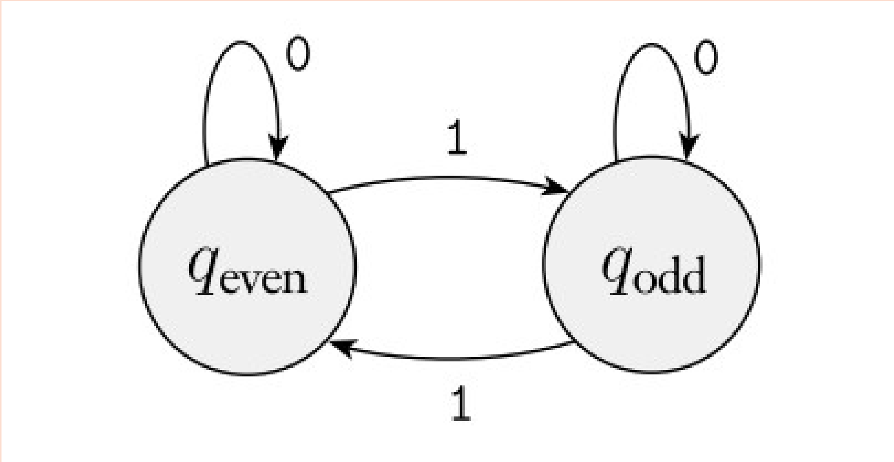
- Check for strings
- 0
- 00
- 001
- 010
- 01011
- 01011011



DESIGNING FINITE AUTOMATA

Example 1

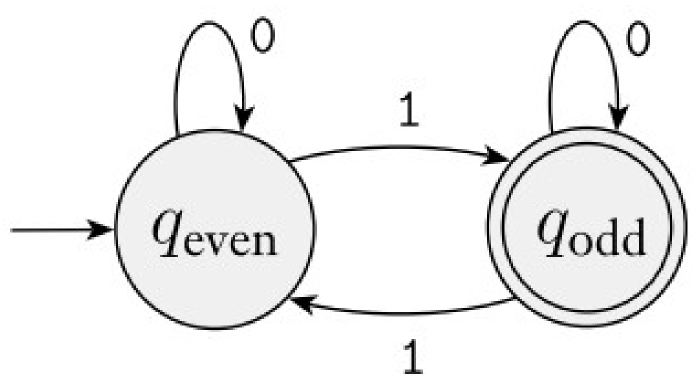
- Assign the transitions by seeing how to go from one possibility to another upon reading a symbol



DESIGNING FINITE AUTOMATA

Example 1

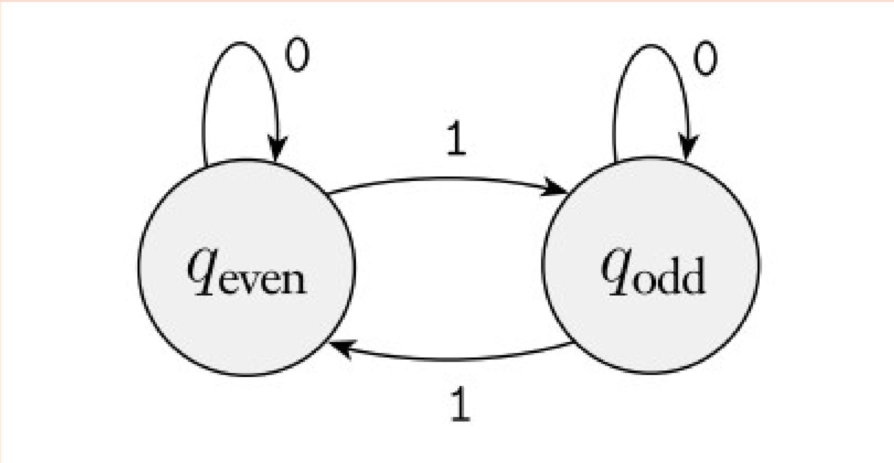
- Add the start state
- Set the start state to be the state corresponding to the possibility associated with having seen 0 symbols (Zero occurrences of 1s) so far (the empty string ε)
- As 0 is an even number, Start state corresponds to q_{even}



DESIGNING FINITE AUTOMATA

Example 1

- Add the accept state
- Accept state to be those corresponding to possibilities where you want to accept the input string.
- q_{odd} is set as an accept state



DESIGNING FINITE AUTOMATA

Example 2

1) Design a finite automaton E2 to recognize all strings that contain the string 001 as a substring.

2) For example, 0010, 1001, 001, and 1111111001111 are all in the language, but 11 and 0000 are not.

Tutorials

- (I) Given a Machine, State Diag / Transition Table → Write the 5-tuple representⁿ
- (II) Given a M/c, Check whether the given strings are accepted / rejected by the m/c
- (III) Design a FA / m/c from given language
  - Draw the state diag
  - Level 2
- (IV) M/c given, Identify the lang being recognized

Level 1

Σ Σ

CHARIOT

seq

Σ

HARBOUR

Σ

001

Σ

01010

DESIGNING FINITE AUTOMATA-

RECOGNIZER FA AS A SEQUENCE DETECTOR

understand

DESIGNING FINITE AUTOMATA

Example 2

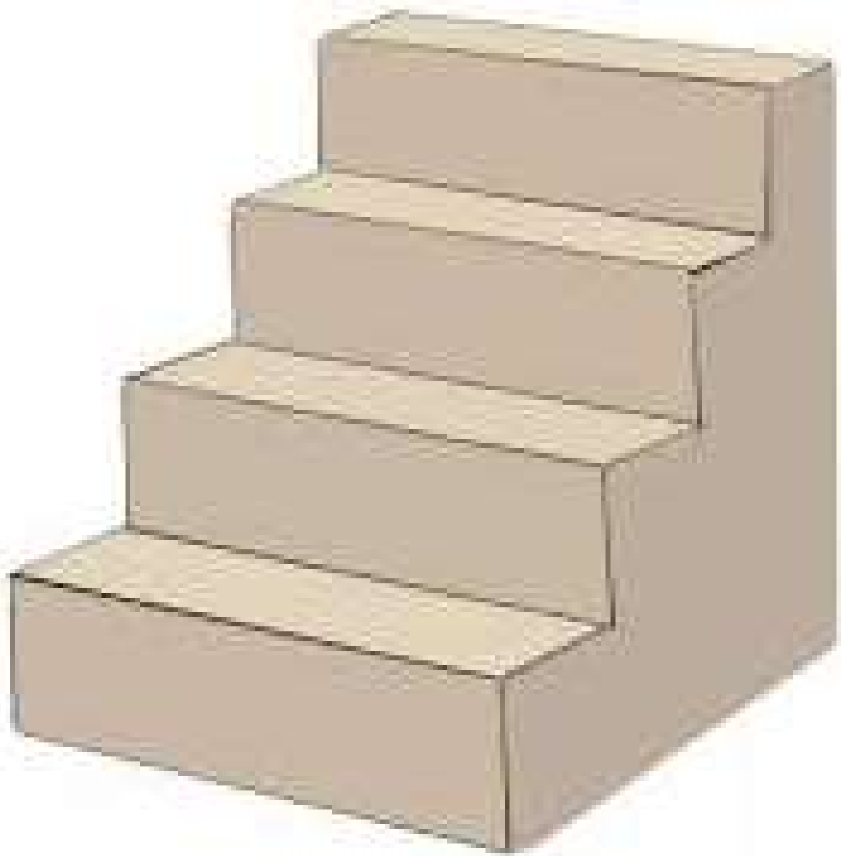
- Pretending to be the Automaton E2 recognize the language
- Start reading the input string of 0s and 1s symbol by symbol

DESIGNING FINITE AUTOMATA

Example 2-Design a finite automaton E2 to recognize all strings that contain the string 001 as a substring.

- As symbols come in, you would initially skip over all 1’s
- So the 4 possibilities are-
  - Haven’t seen any symbols of the pattern=> q state= start state
  - Have just seen a 0=>q_0 state
  - Have just seen 00=>q_{00} state
  - Have seen the entire pattern 001=>q_{001} state

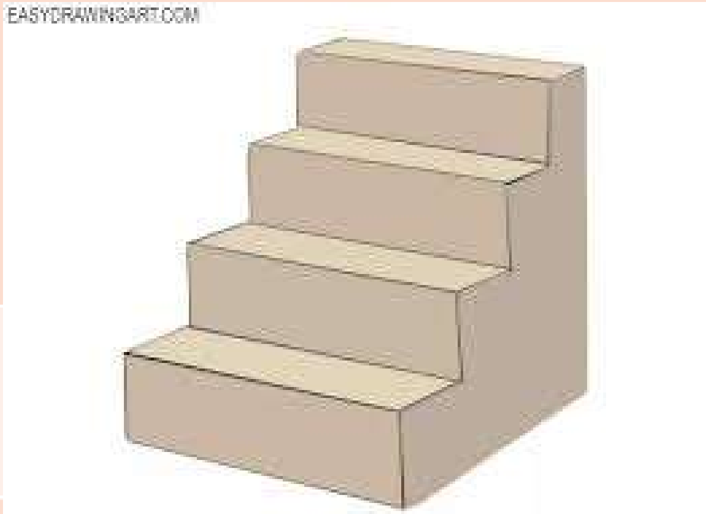
??



DESIGNING FINITE AUTOMATA

Example 2-Assign transitions

- FA moves from start state q to q_0, only if "0" is read, Otherwise remains in the same state
- q reading a 0, you move to q_0
- q reading a 1, you stay in q



- q_0 reading a 1, you return to q
- q_0 reading a 0, you move to q_{00}

- q_{00} reading a 1, you move to final state q_{001}
- q_{00} reading a 0, stays in q_{00}

- q_{001} reading a 0 or 1, stays in q_{001}
- If FA reads any symbol after reaching final state q_{001}, it does not make a transition to any other state as it has already recognized the string "001" by then

DESIGNING FINITE AUTOMATA

Example 2

- Assign transitions
- q reading a 1, you stay in q
- q reading a 0, you move to q_0
- q_0 reading a 1 you return to q
- q_0 reading a 0, you move to q_{00}
- q_{00} reading a 1, you move to q_{001}
- q_{00} reading a 0, stays in q_{00}
- q_{001} reading a 0 or 1, stays in q_{001}

1111001 ideal

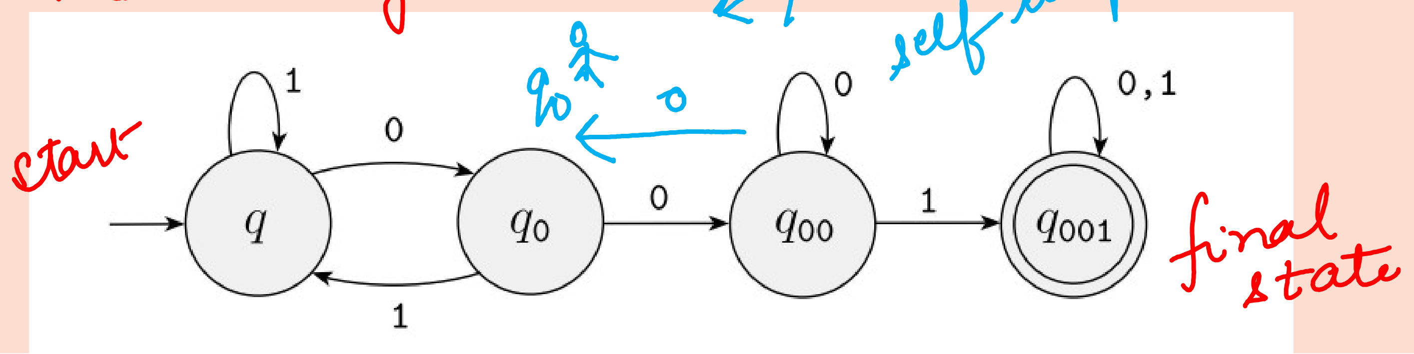
q

1111000 1 accepted

q q_0 q_{00}

q_0

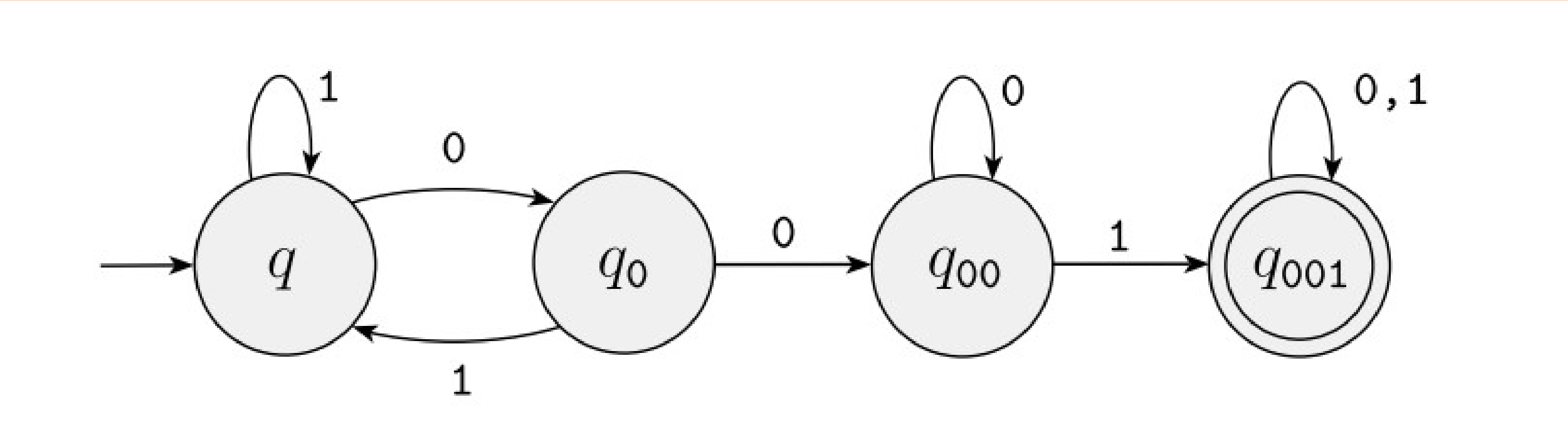
state diagram



DESIGNING FINITE AUTOMATA

Example 2

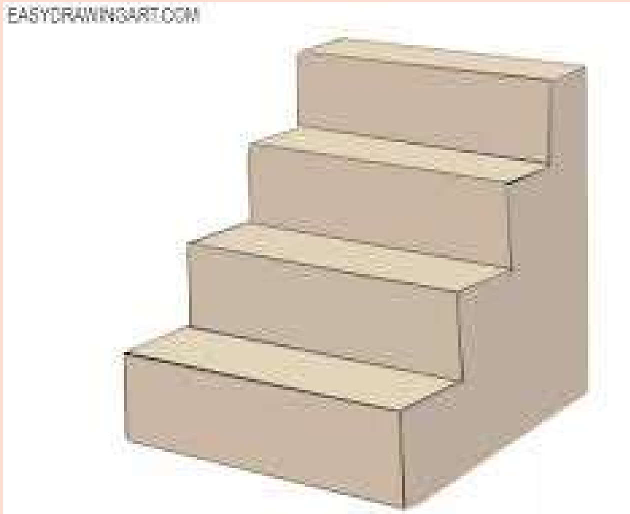
- Start state =q
- The only accept state =q_{001}



DESIGNING FINITE AUTOMATA

Exercise

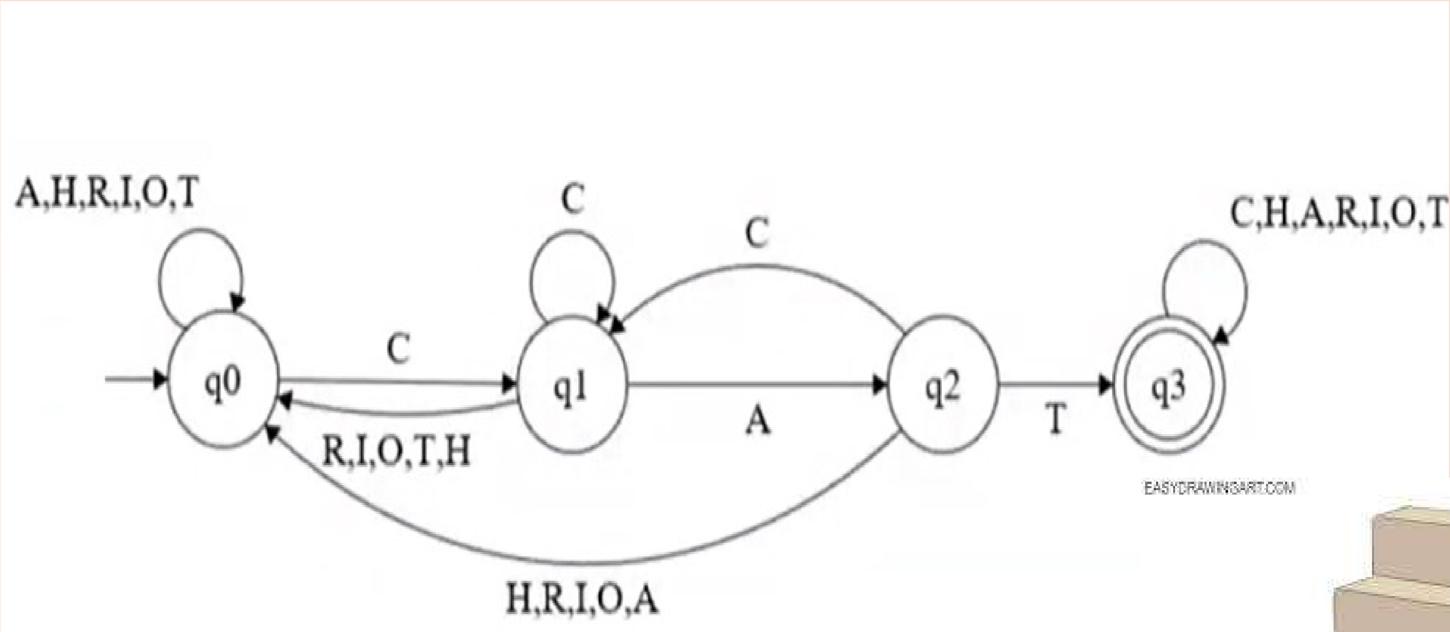
- Design a DFA with inputs as characters of “CHARIOT” which recognizes the string “CAT” as substring.

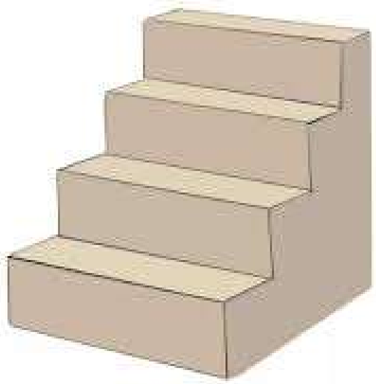


DESIGNING FINITE AUTOMATA

Exercise

- Design a DFA with inputs as characters of "CHARIOT" which recognizes the string "CAT" as substring.

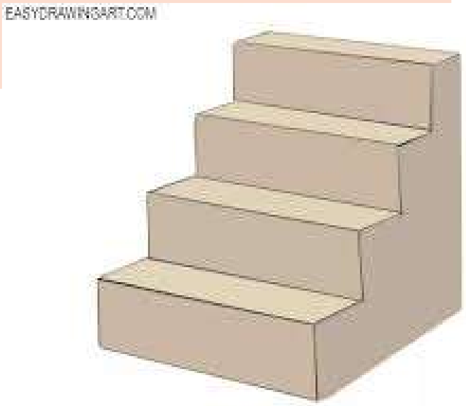




DESIGNING FINITE AUTOMATA

Example 3

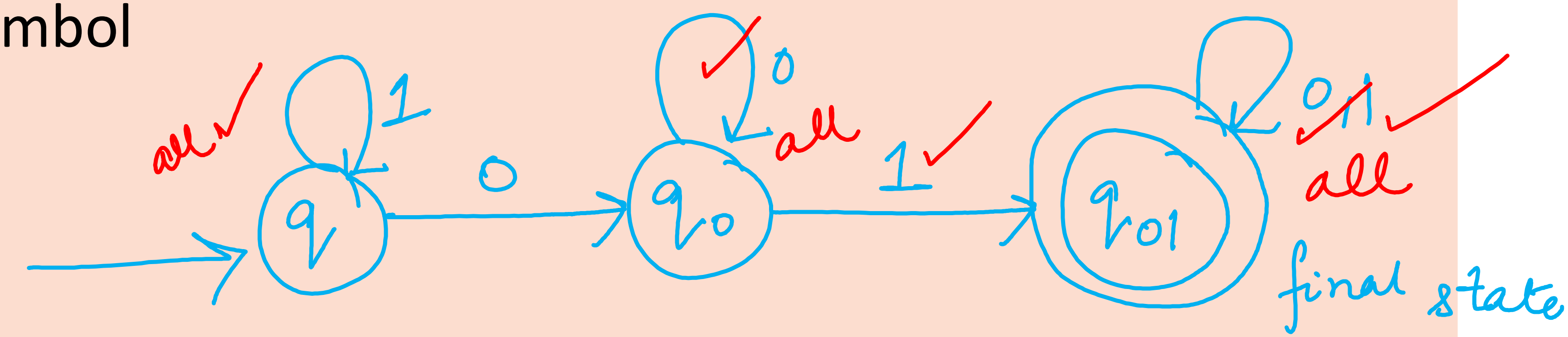
Let us formally specify a DFA that accepts all and only the strings of 0's and 1's that have the sequence 01 somewhere in the string. $\Sigma = \{0,1\}$



DESIGNING FINITE AUTOMATA

Example 3

- Pretending to be the Automaton E3 recognize the language
- Start reading the input string of 0s and 1s symbol by symbol



11101

q

start

00001 / 001

DESIGNING FINITE AUTOMATA

Example 3

- As symbols come in, you would initially skip over all 1's
- So the 3 possibilities are-
  - Haven't seen any symbols of the pattern=> q state
  - Have just seen a 0=>q_0 state
  - Have seen the entire pattern 01=>q_{01} state

001 → 01

sequence detector

DESIGNING FINITE AUTOMATA

Example 3

- Assign transitions
- q reading a 1, you stay in q
- q reading a 0, you move to q_0
- q_0 reading a 0, stays in q_0
- q_0 reading a 1, you move to q_{01}
- q_{01} reading a 0 or 1, stays in q_{01}

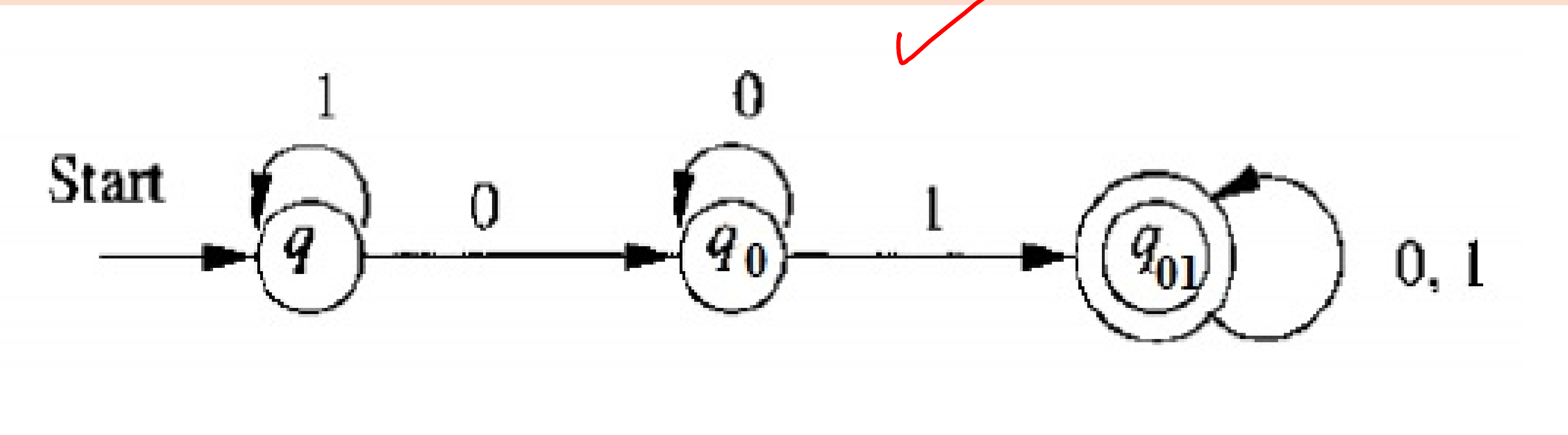
DESIGNING FINITE AUTOMATA

Example 3

Transitn table

	0	1
q		
q_0		
q_{01}		

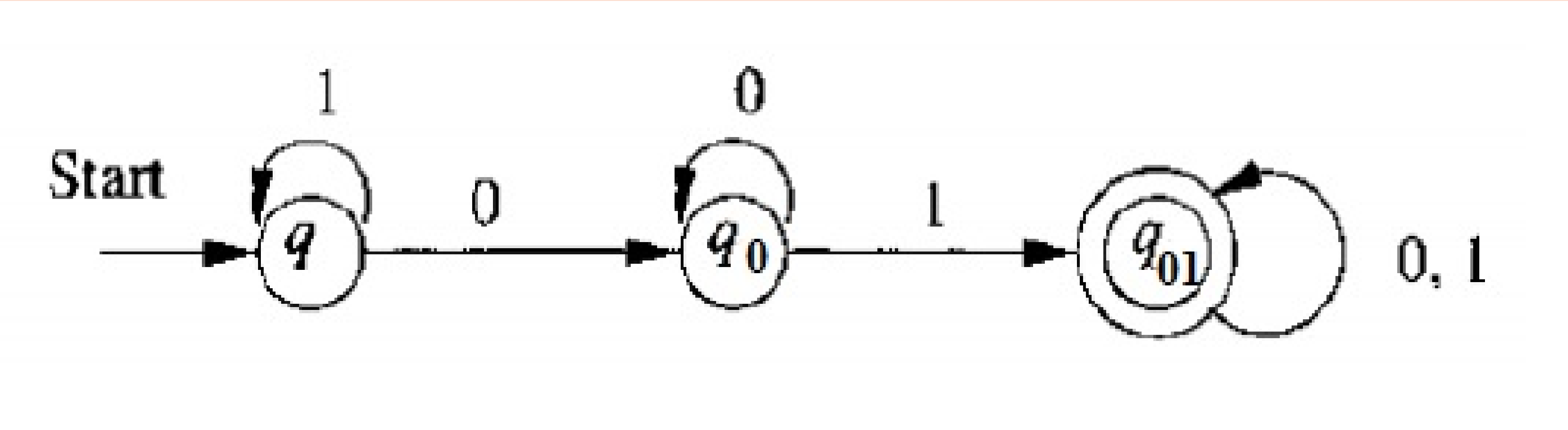
- Assign transitions
- q reading a 1, you stay in q
- q reading a 0, you move to q_0
- q_0 reading a 0, stays in q_0
- q_0 reading a 1, you move to q_{01}
- q_{01} reading a 0 or 1, stays in q_{01}



DESIGNING FINITE AUTOMATA

Example 3

- Start state =q
- The only accept state =q_{01}



DESIGNING FINITE AUTOMATA

Example 3

- We can write this language L as:

{w | w is of the form x01y for some strings x and y consisting of 0’s and 1’s only.}

- Another equivalent description, using parameters x and y to the left of the vertical bar, is:

{x01y | x and y are any strings of 0’s and 1’s}

DESIGNING FINITE AUTOMATA

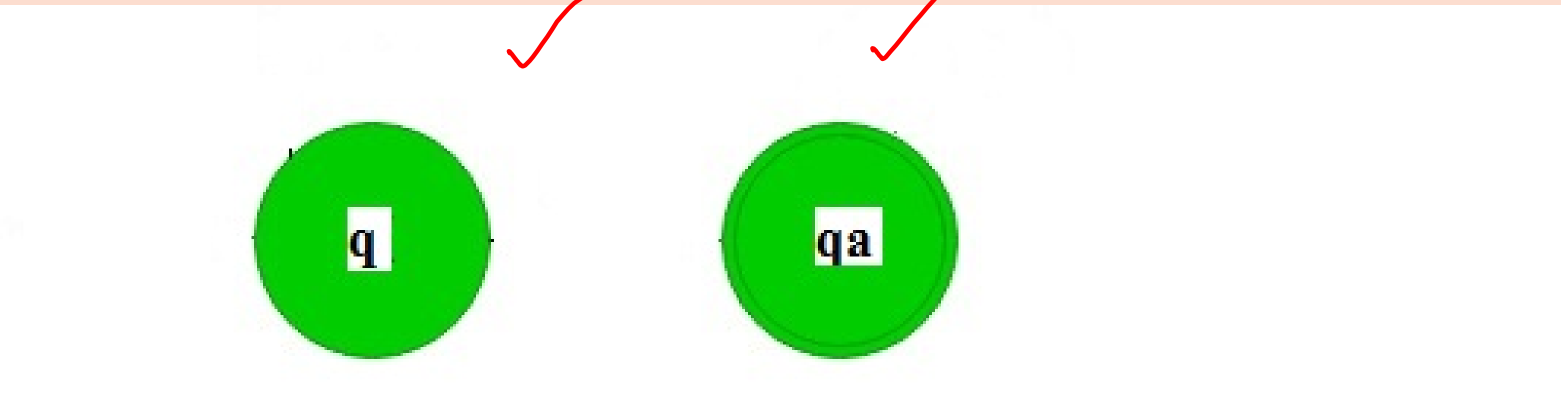
Exercise 1

Construct a DFA accepting set of string over {a, b} where each string containing ‘a’ as the substring.

DESIGNING FINITE AUTOMATA

Exercise 1

- As symbols come in, you would initially skip over all b's
- So the 2 possibilities are-
  - Haven't seen any "a" symbol => q state
  - Have just seen an "a"=>q_a state

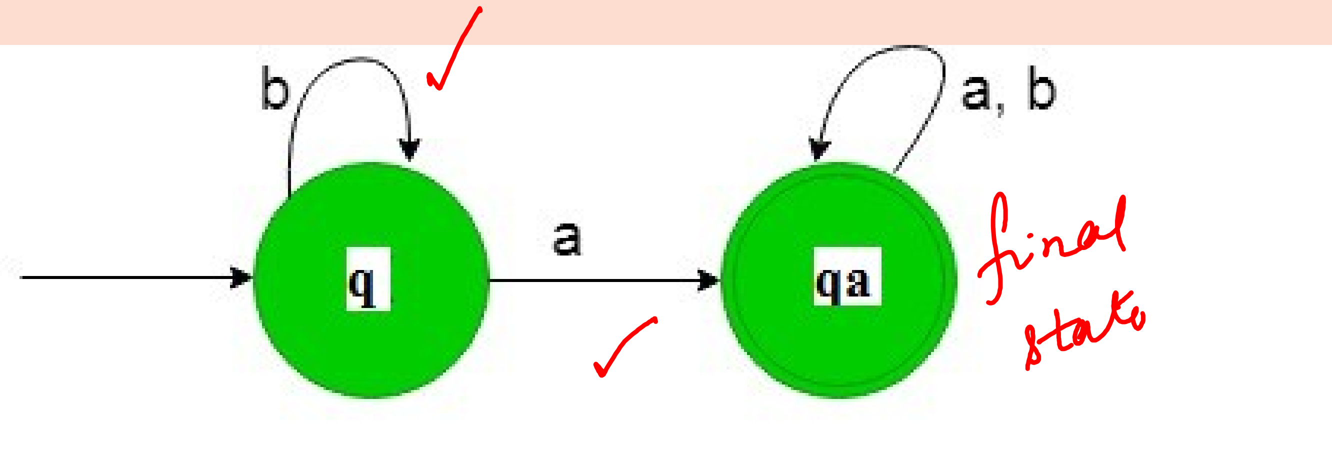


DESIGNING FINITE AUTOMATA

Exercise 1

Assign transitions

- q getting “b”, stays in state q itself
- q reading “a”, it transits to the final state q_a
- q_a On reading a or b , stays in the final state q_a itself.



Strings

a

aa

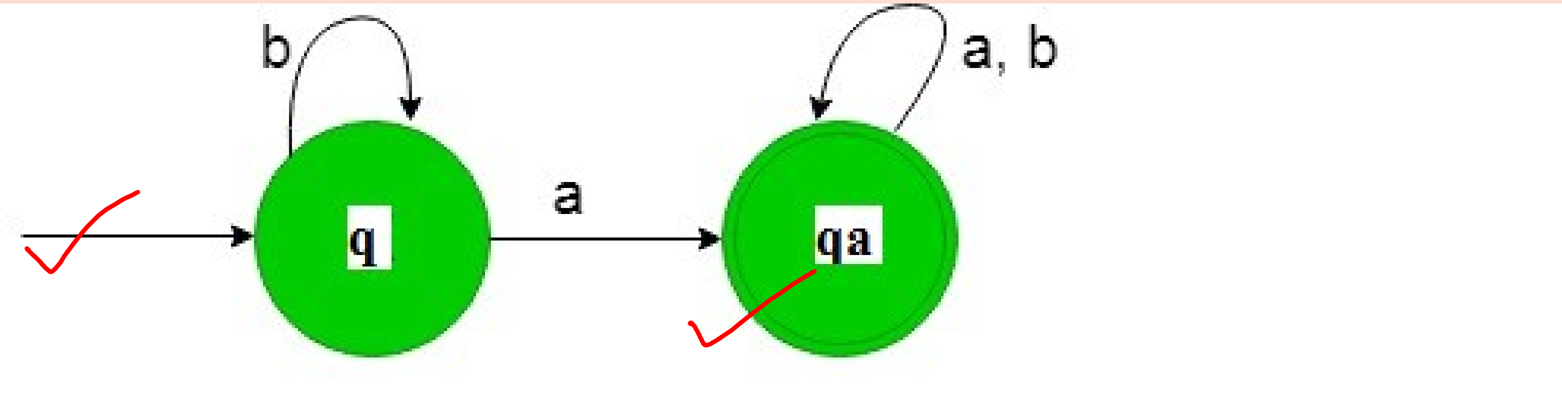
aab

aabb

DESIGNING FINITE AUTOMATA

Exercise 1

- Start state =q
- The only accept state =q_a



DESIGNING FINITE AUTOMATA

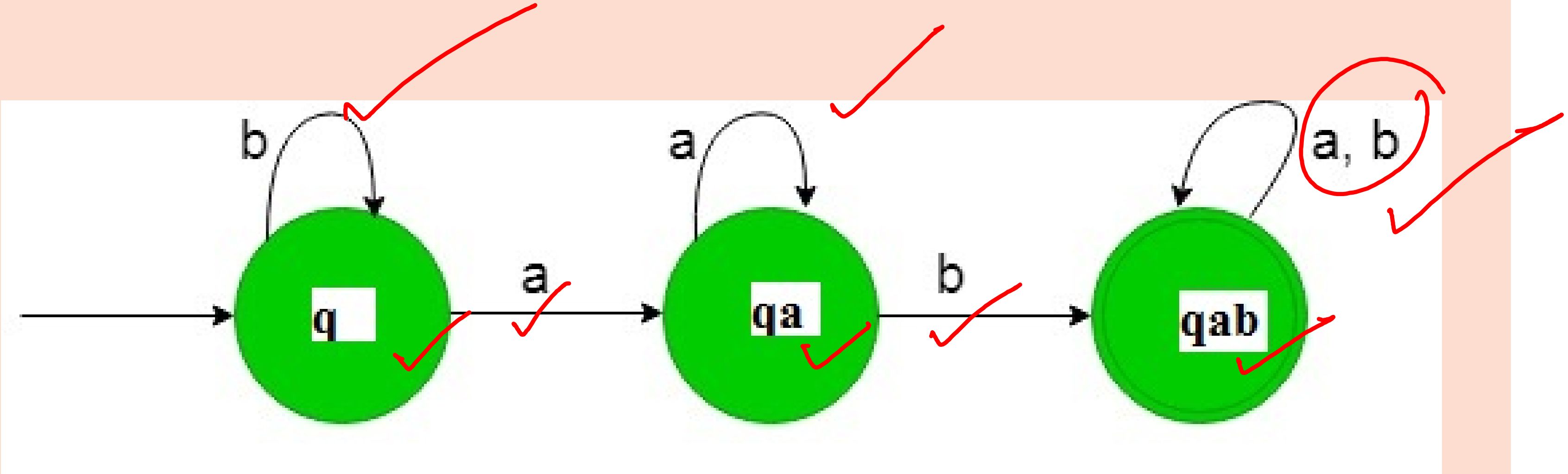
Exercise 2

Construction of a minimal DFA accepting set of string over {a, b} where each string containing ‘ab’ as the substring.

DESIGNING FINITE AUTOMATA

Exercise 2

Construction of a minimal DFA accepting set of string over {a, b} where each string containing 'ab' as the substring.



https://www.geeksforgeeks.org/designing-deterministic-finite-automata-set-3/

DESIGNING FINITE AUTOMATA

Example 4

Construction of a DFA to check whether given decimal number is divisible by three

DESIGNING FINITE AUTOMATA

Example 4

- The possible Remainders =0,1,2
- Remainder 0=>Divisible by 3 yes
- Group Digits according to their remainder
  - Initial State=A
  - (0,3,6,9,12,15…)mod 3=0, Remainder 0 group S0=>Final State
  - (1,4,7,10,13,16…)mod 3=1, Remainder 1 group S1
  - (2,5,8,11,14,17…)mod 3=2, Remainder 2 group S2

DESIGNING FINITE AUTOMATA

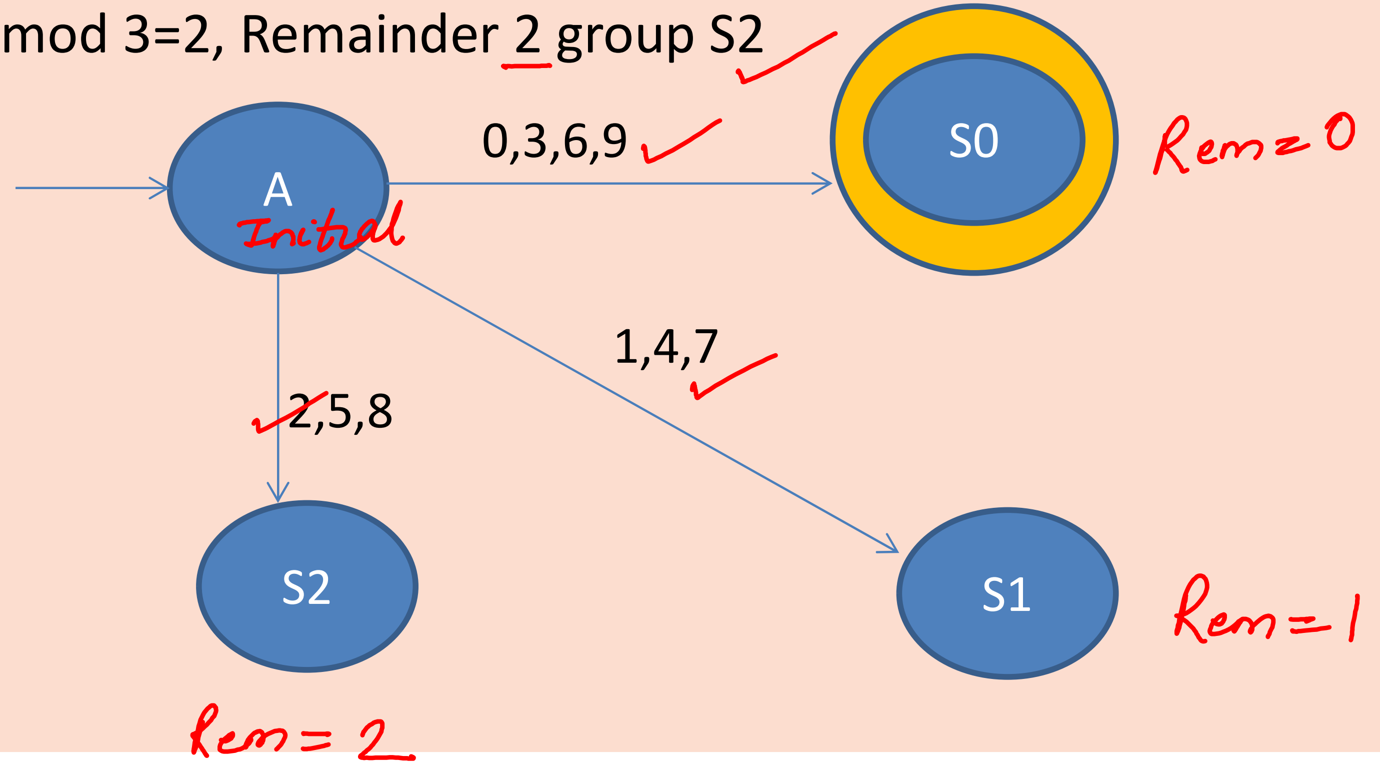
Example 4

3 5

Input tape

Single Digits

- (0,3,6,9...) mod 3=0, Remainder 0 group S0=>Final State
- (1,4,7) mod 3=1, Remainder 1 group S1
- (2,5,8) mod 3=2, Remainder 2 group S2



DESIGNING FINITE AUTOMATA

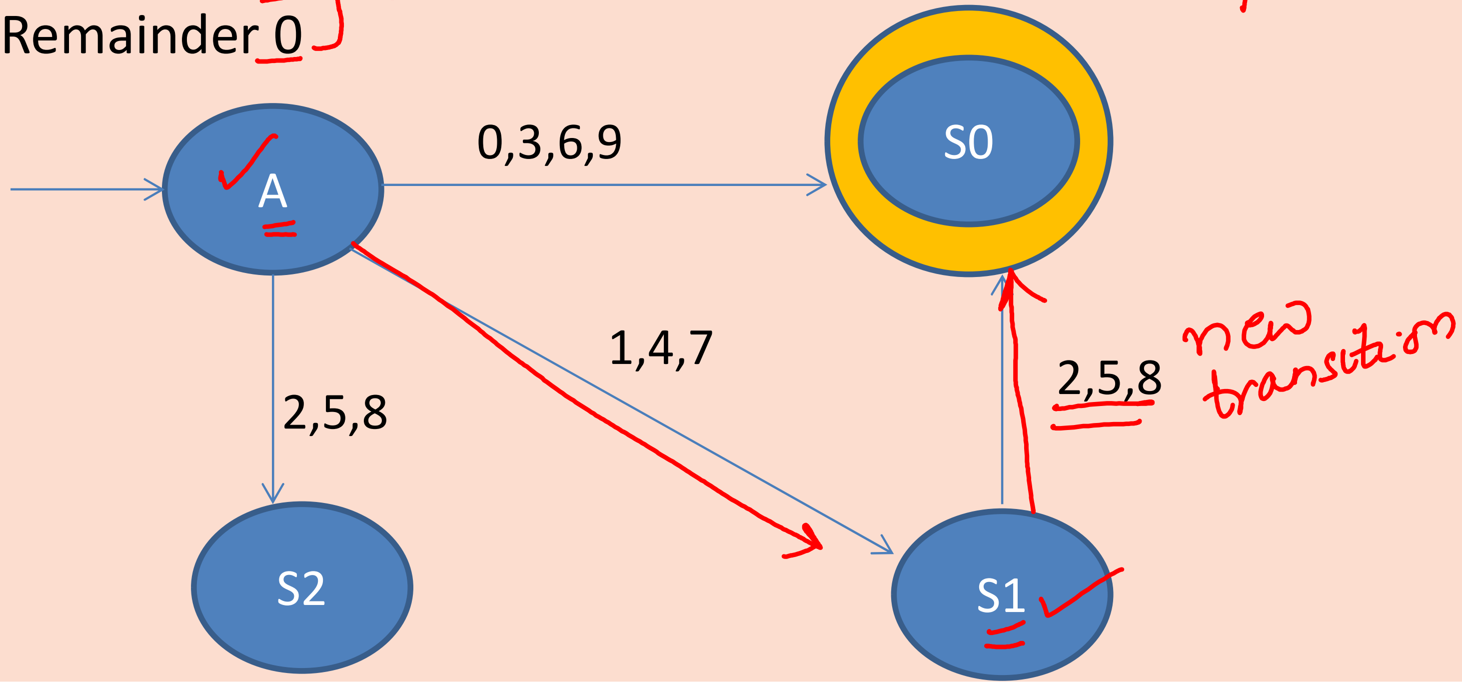
Example 4

Double Digits

- (0,3,6,9…) mod 3=0, Remainder 0 group S0=>Final State
- For 12, Remainder 0
- For 15, Remainder 0
- For 18, Remainder 0

S0

| 1 | 2 |

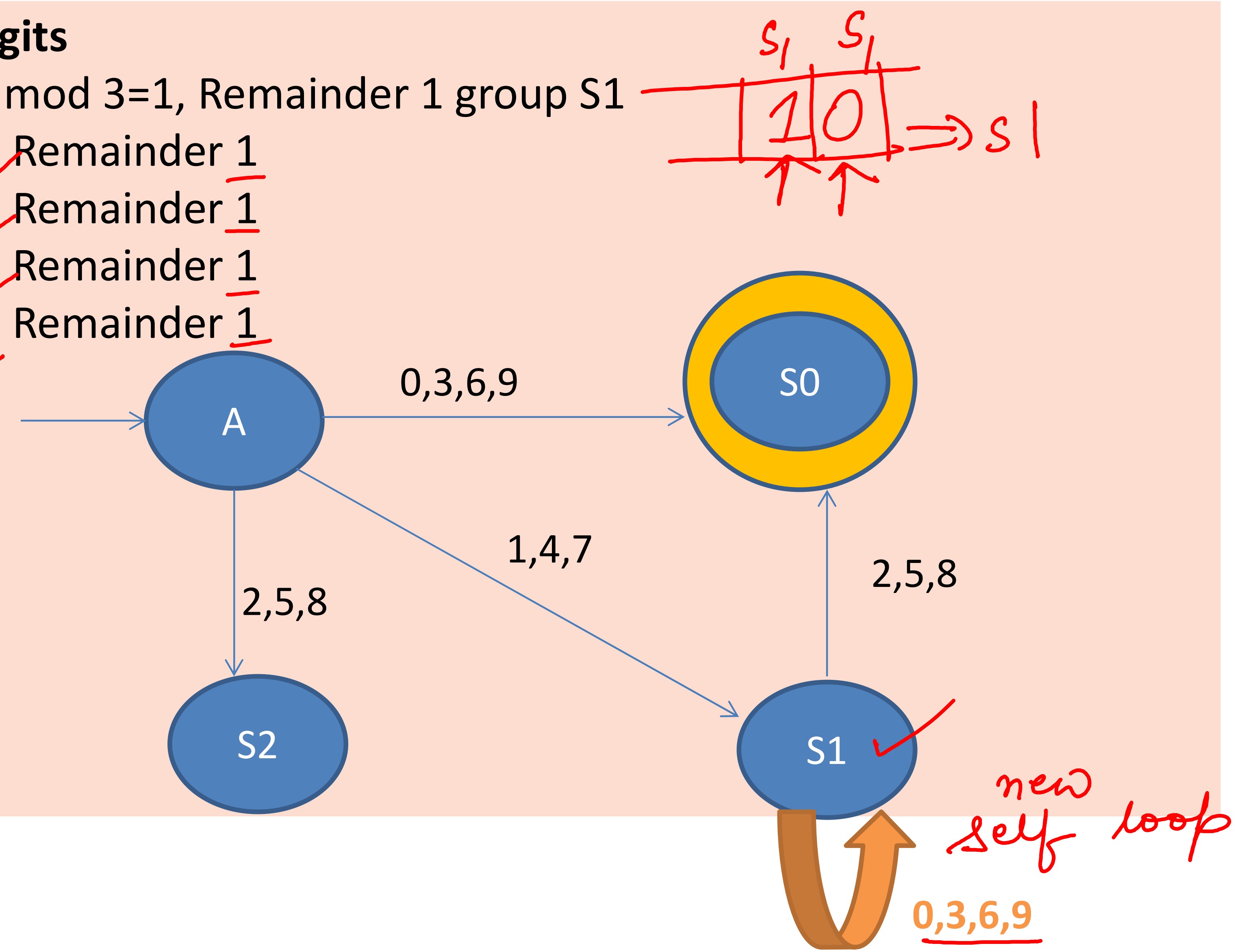


DESIGNING FINITE AUTOMATA

Example 4

Double Digits

- (1,4,7) mod 3=1, Remainder 1 group S1
- For 10, Remainder 1
- For 13, Remainder 1
- For 16, Remainder 1
- For 19, Remainder 1



DESIGNING FINITE AUTOMATA

Example 4

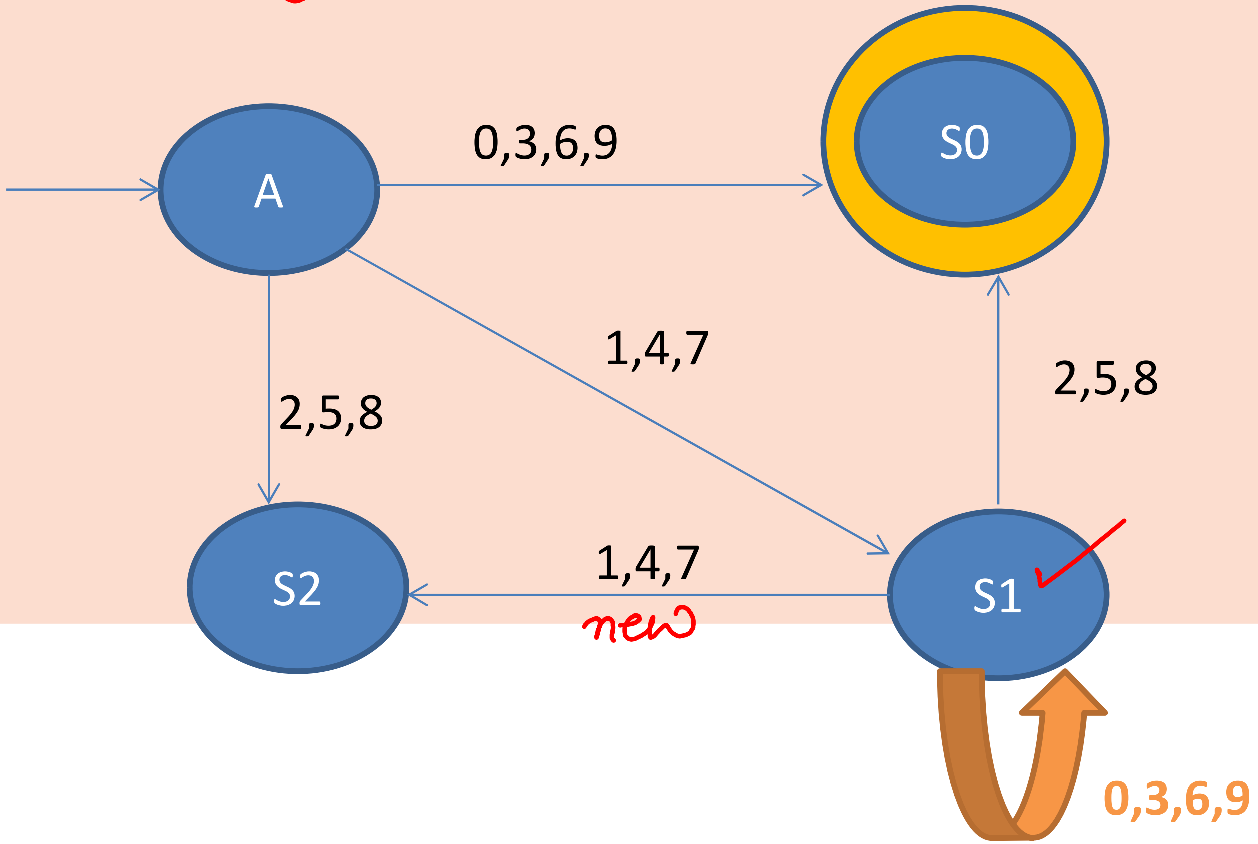
Double Digits

- (2,5,8,11,14,17…)mod 3=2, Remainder 2 group S2
- For 11, Remainder 2
- For 14, Remainder 2
- For 17, Remainder 2

1,4,7 ⇒ S2

S1

| 1 | 1 |



DESIGNING FINITE AUTOMATA

Example 4

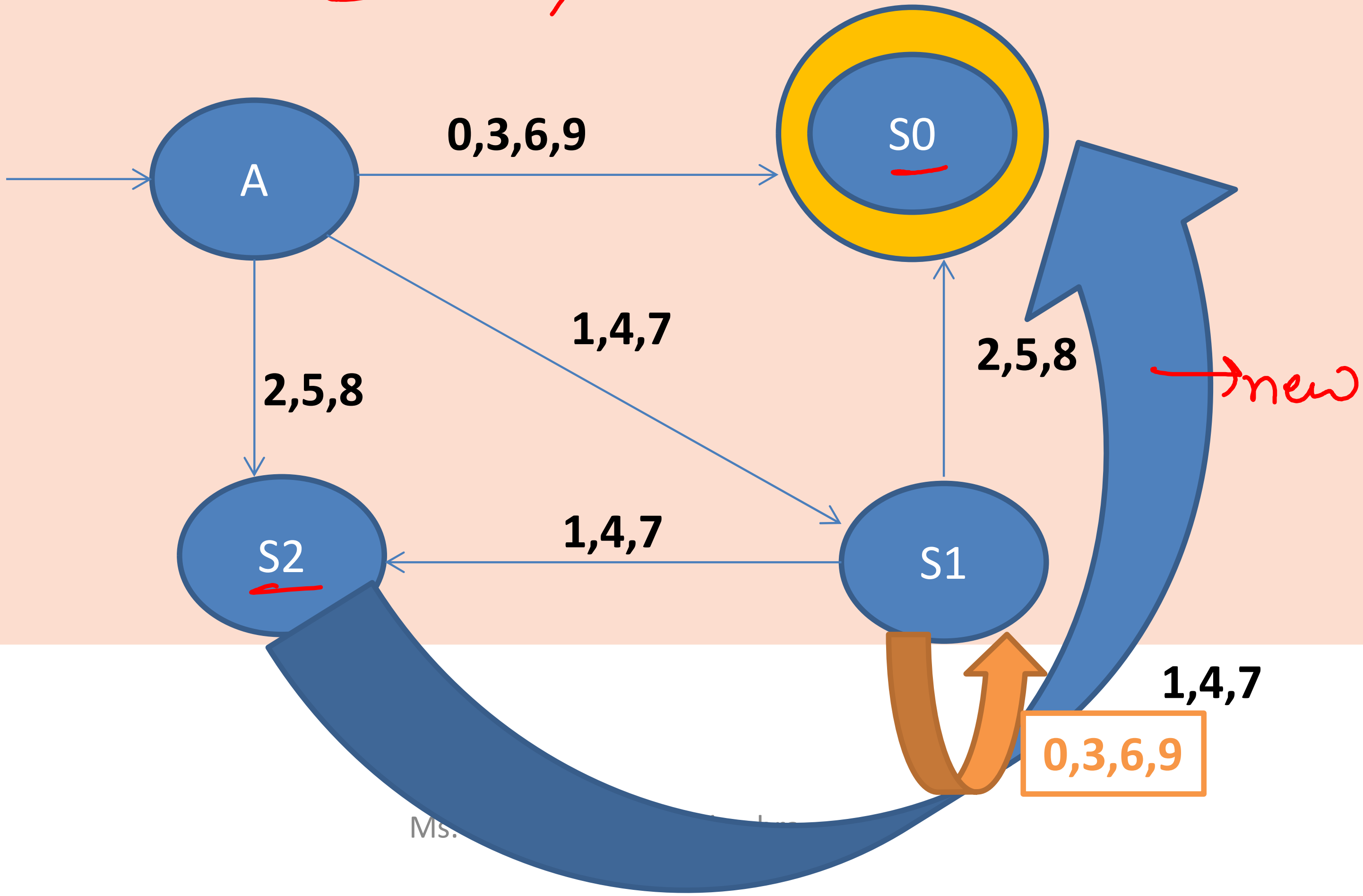
Double Digits

- For 21, Remainder 0
- For 24, Remainder 0
- For 27, Remainder 0

$\xrightarrow{1,4,7}$

S2 S0

2	1



DESIGNING FINITE AUTOMATA

Example 4

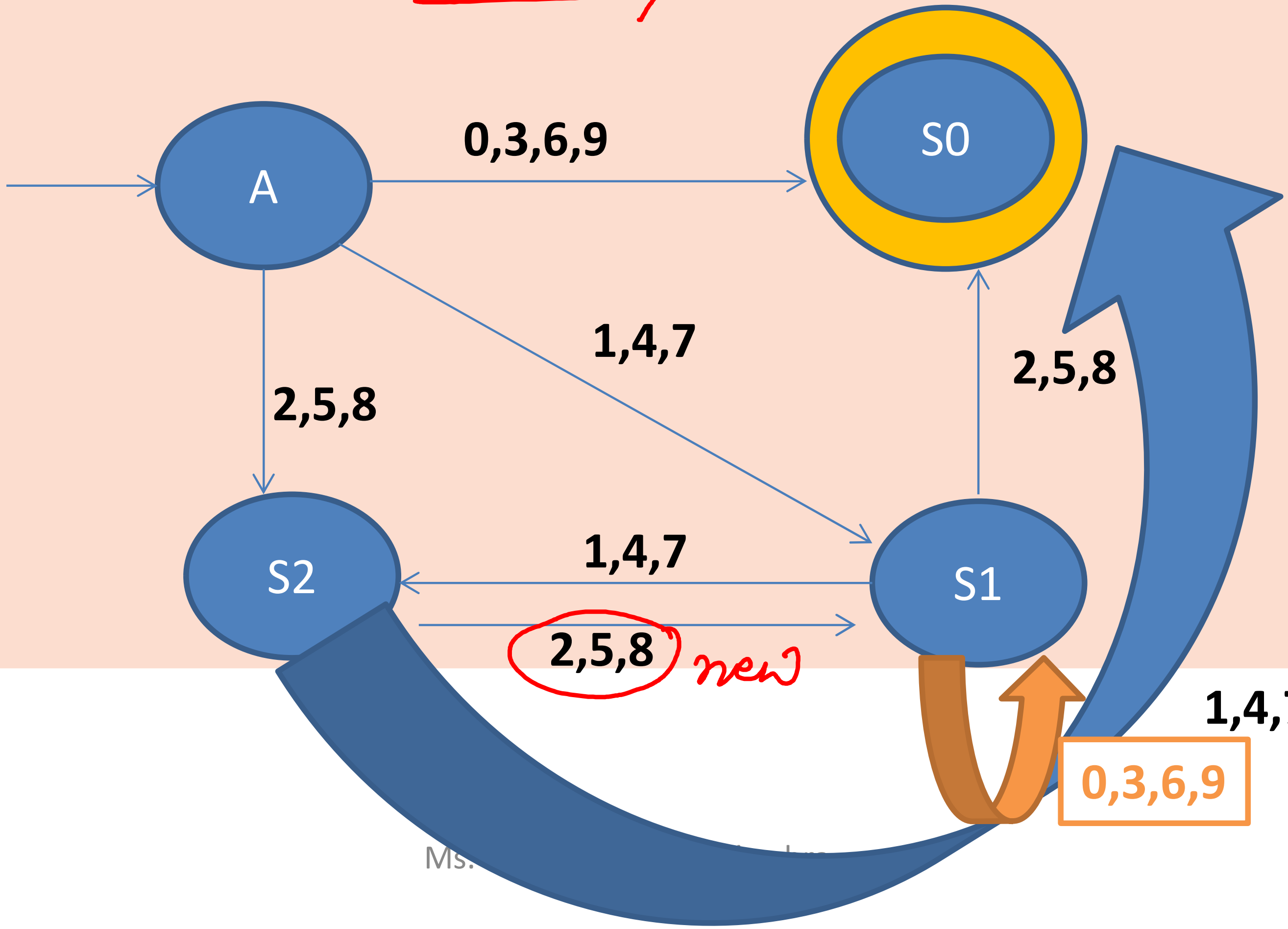
Double Digits

S2 S1

| 2 | 2 |

- For 22, Remainder 1
- For 25, Remainder 1 ⇒ S1
- For 28, Remainder 1

2,5,8 →



DESIGNING FINITE AUTOMATA

Example 4

Double Digits

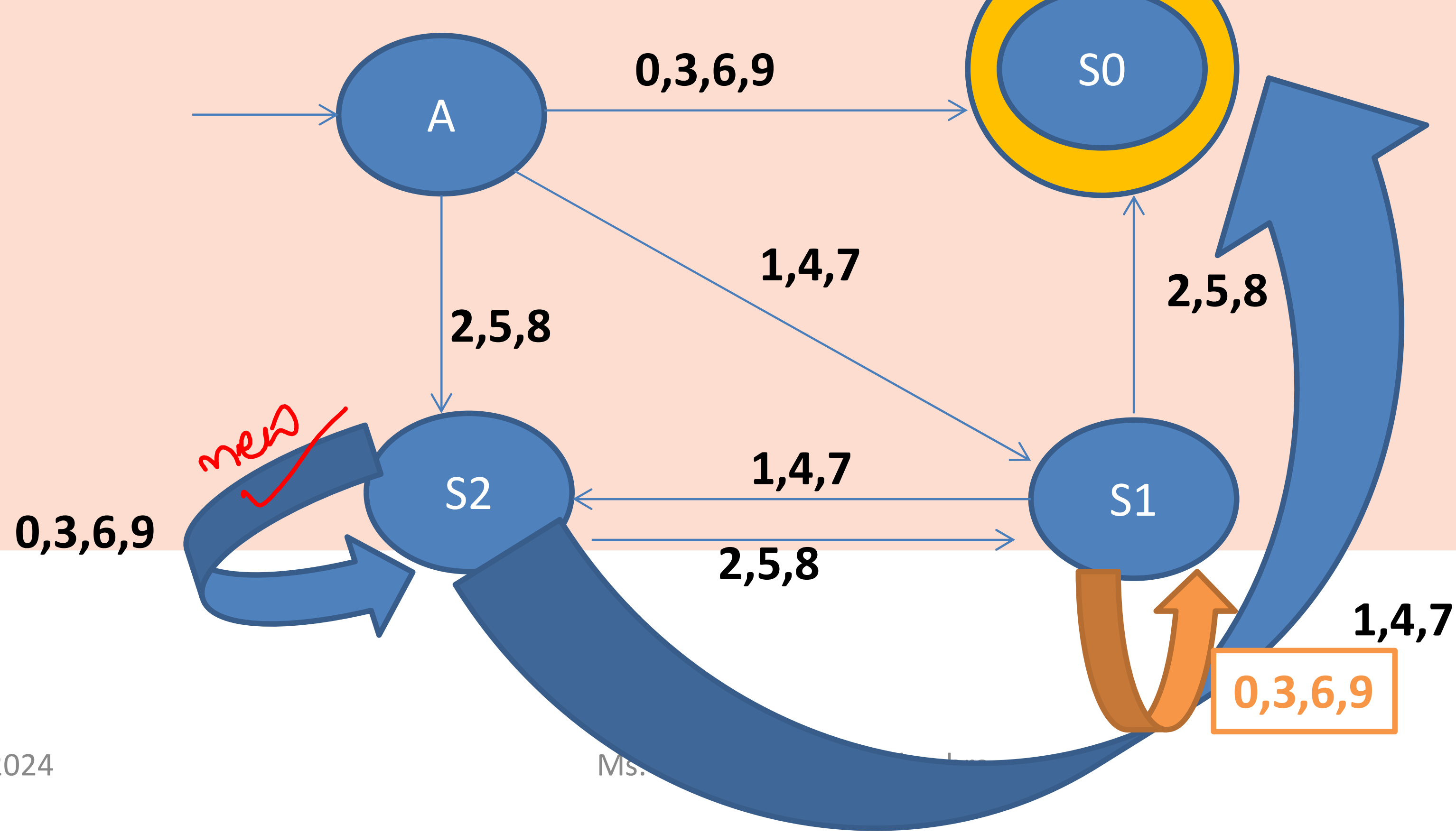
- For 20, Remainder 2 ✓
- For 23, Remainder 2 ✓
- For 26, Remainder 2 ✓
- For 29, Remainder 2 ✓

S2 S2

7 | 2 | 0 |

self loop

0,3,6,9 →



DESIGNING FINITE AUTOMATA

Example 4

Double Digits

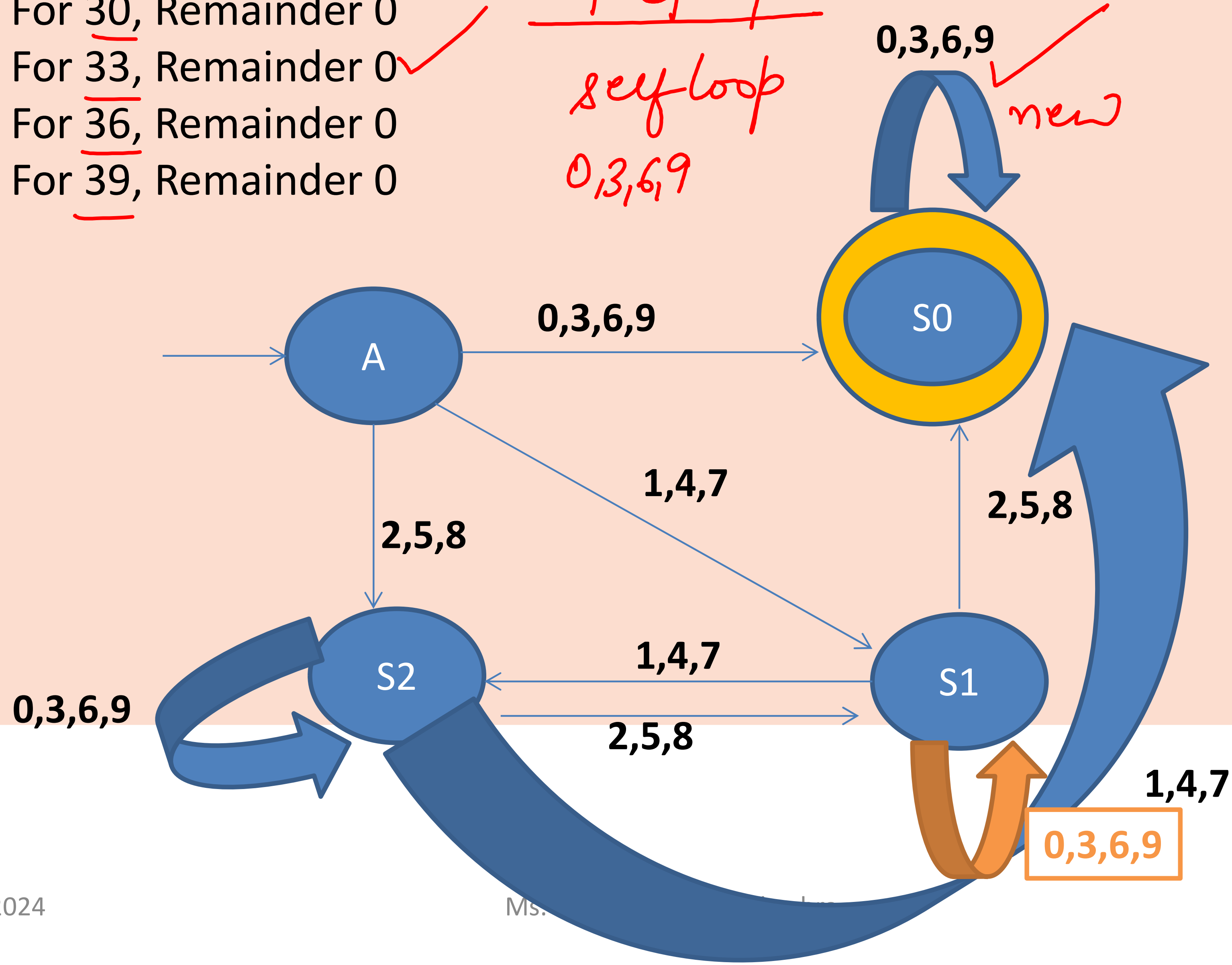
- For 30, Remainder 0
- For 33, Remainder 0 ✓
- For 36, Remainder 0
- For 39, Remainder 0

so so

| 3 | 0 |

selfloop

0,3,6,9

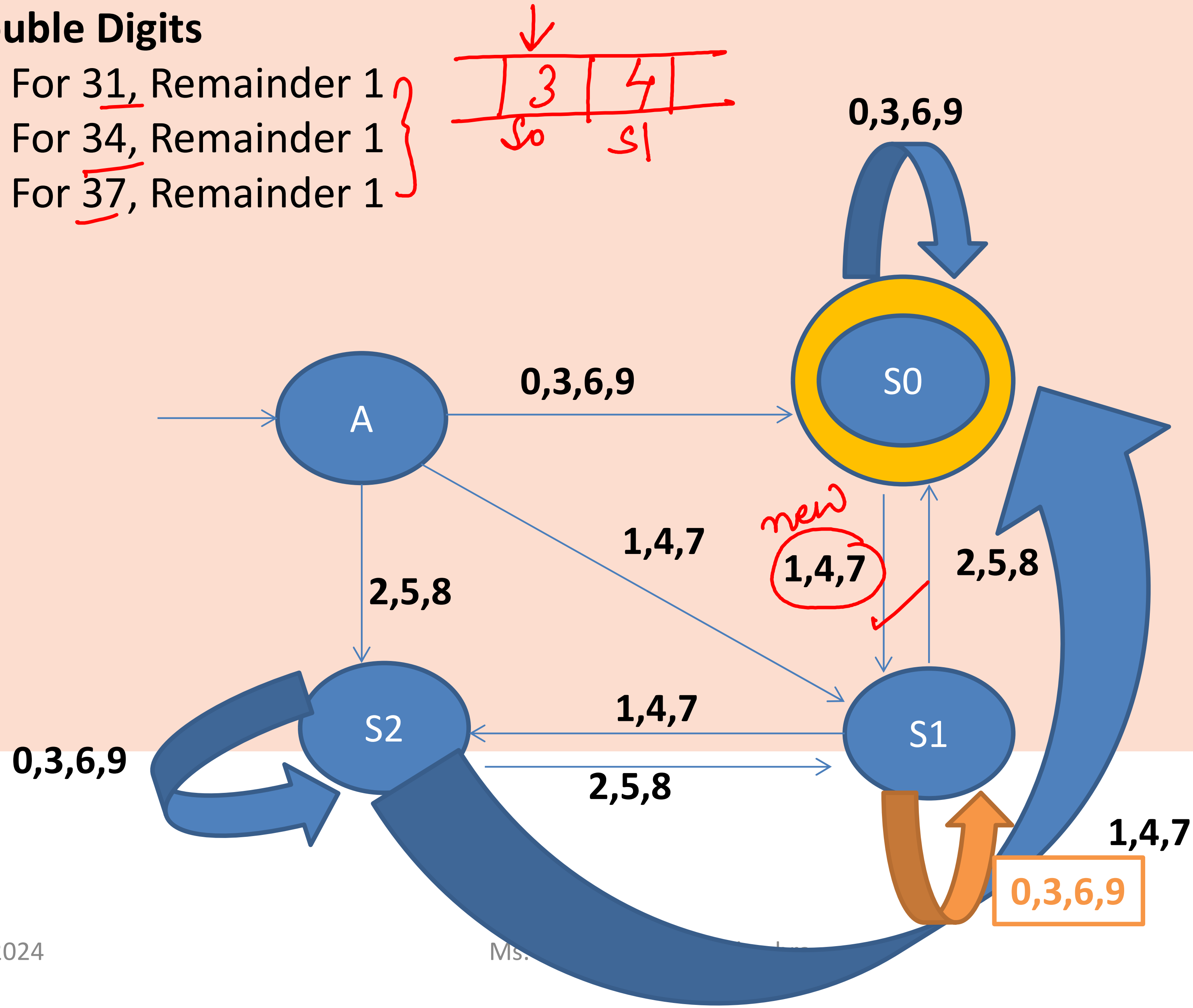


DESIGNING FINITE AUTOMATA

Example 4

Double Digits

- For 31, Remainder 1
- For 34, Remainder 1
- For 37, Remainder 1



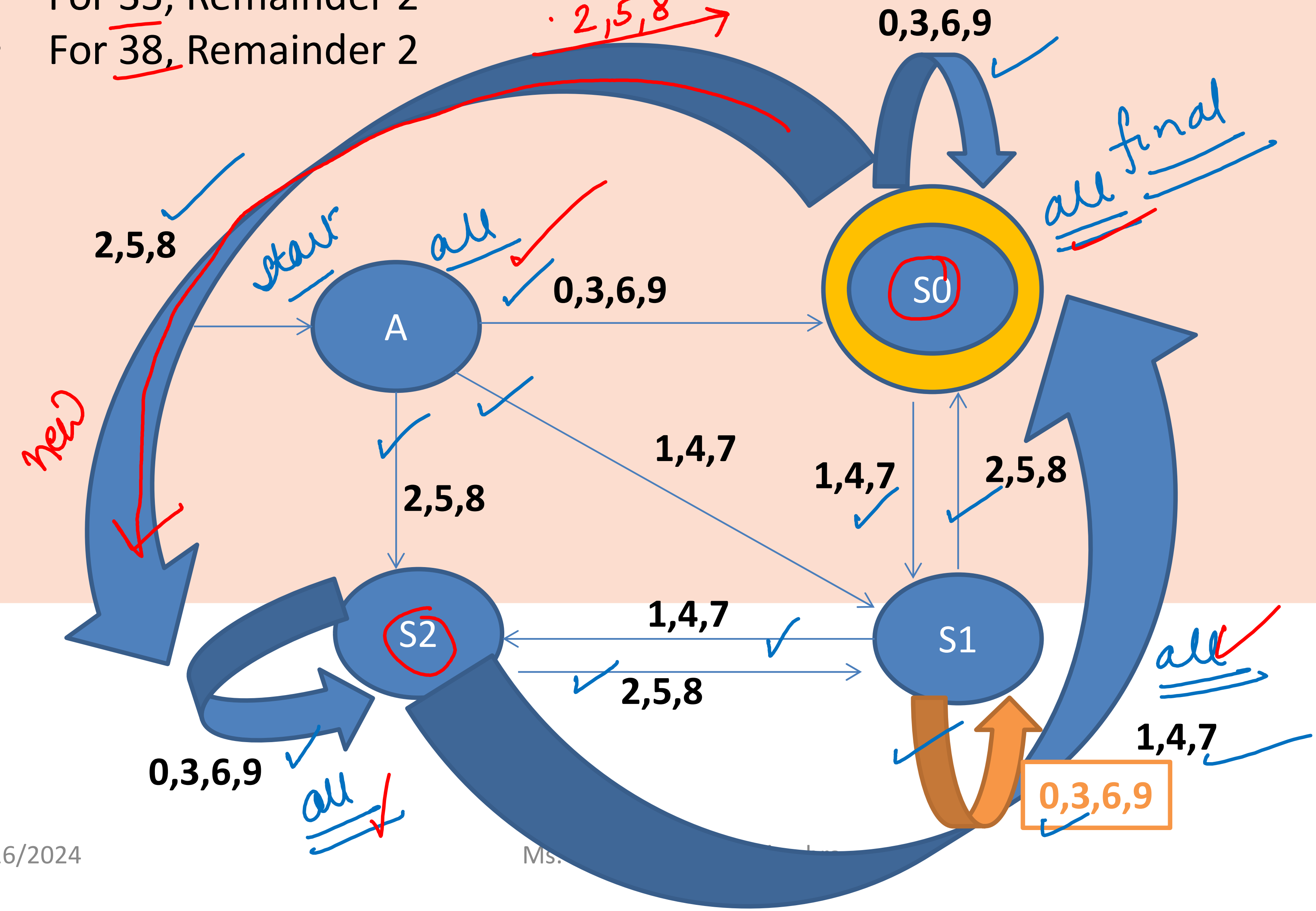
DESIGNING FINITE AUTOMATA

Example 4

Double Digits

- For 32, Remainder 2
- For 35, Remainder 2
- For 38, Remainder 2

S0 S2
3 5
2,5,8

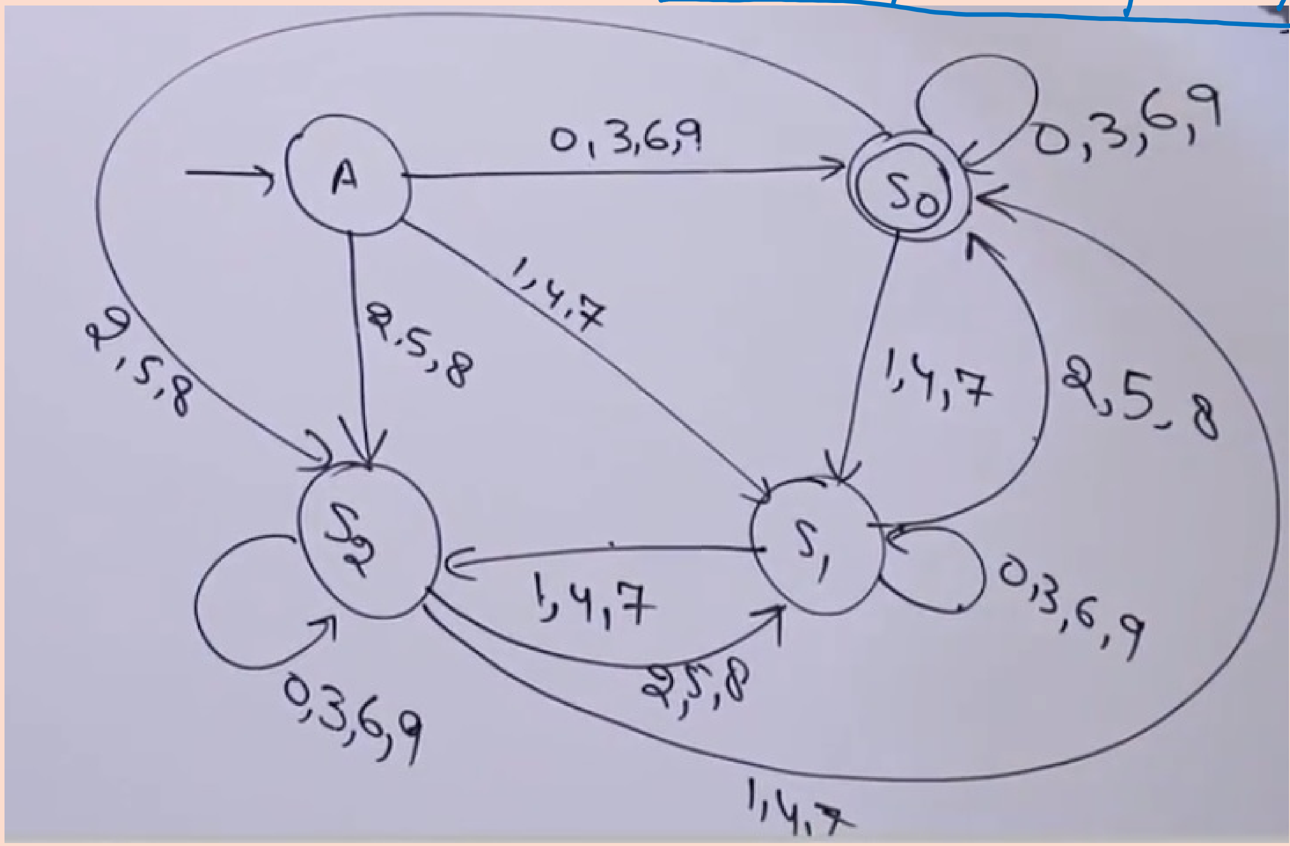


DESIGNING FINITE AUTOMATA

Example 4

*	0,3,6,9	1,4,7	2,5,8
0,3,6,9			
1,4,7			
2,5,8			

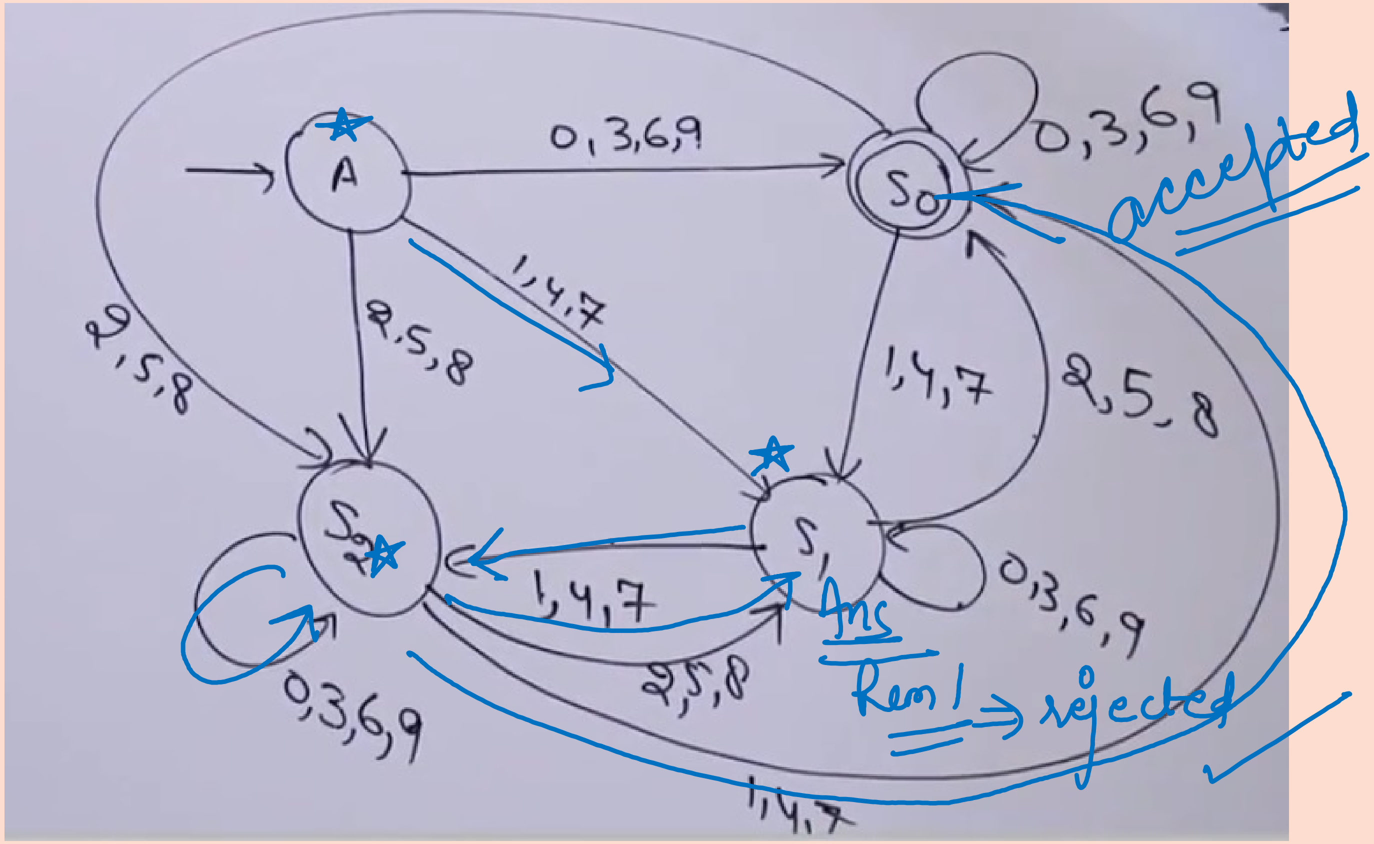
- Complete DFA

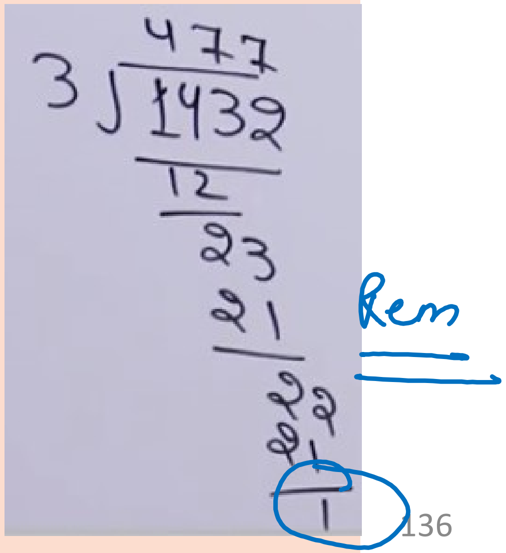


DESIGNING FINITE AUTOMATA

Example 4

- Lets check for 1432,1431
- On traversing for 1432, finally we reach state s1 i.e. Remainder 1,
- On traversing for 1431, finally we reach state s0 i.e. Remainder 0





DESIGNING FINITE AUTOMATA

Example 5

Design a DFA which checks whether the given binary number is even

DESIGNING FINITE AUTOMATA

Example 5

- Binary number that ends with zero=Even
- Binary number that ends with one=Odd

poss Σ states

BINARY NUMBERS

1	1	14	1110
2	10	15	1111
3	11	16	10000
4	100	17	10001
5	101	18	10010
6	110	19	10011
7	111	20	10100
8	1000	21	10101
9	1001	22	10110
10	1010	23	10111
11	1011	24	11000
12	1100	25	11001
13	1101		

ends with 0

q_0

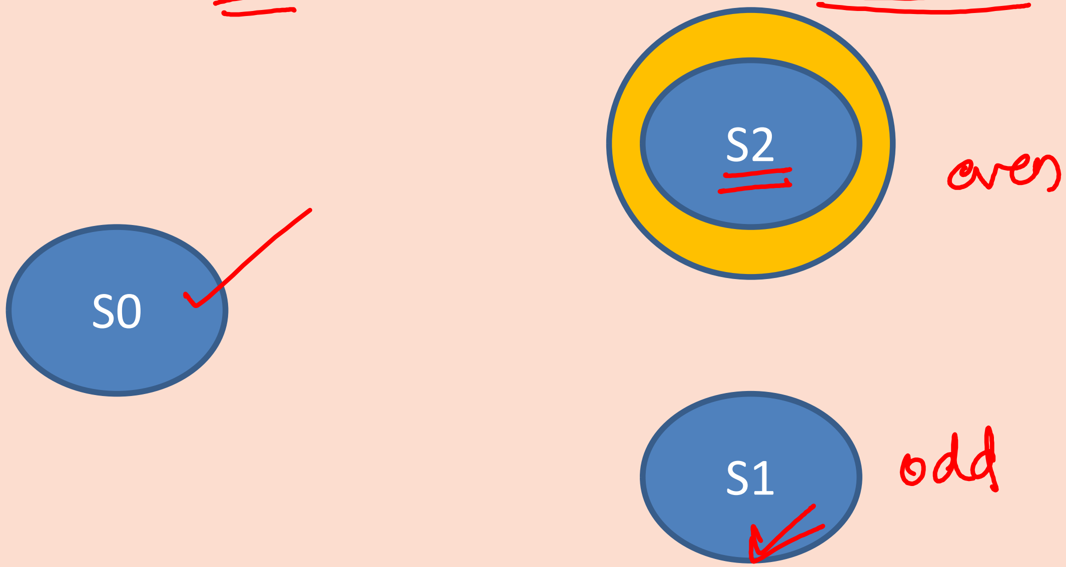
ends with 1

DESIGNING FINITE AUTOMATA

Example 5

Possibilities:

- Start state=S0
- State ending in 1=S1
- State ending in 0=Even numbers=Final state=S2



DESIGNING FINITE AUTOMATA

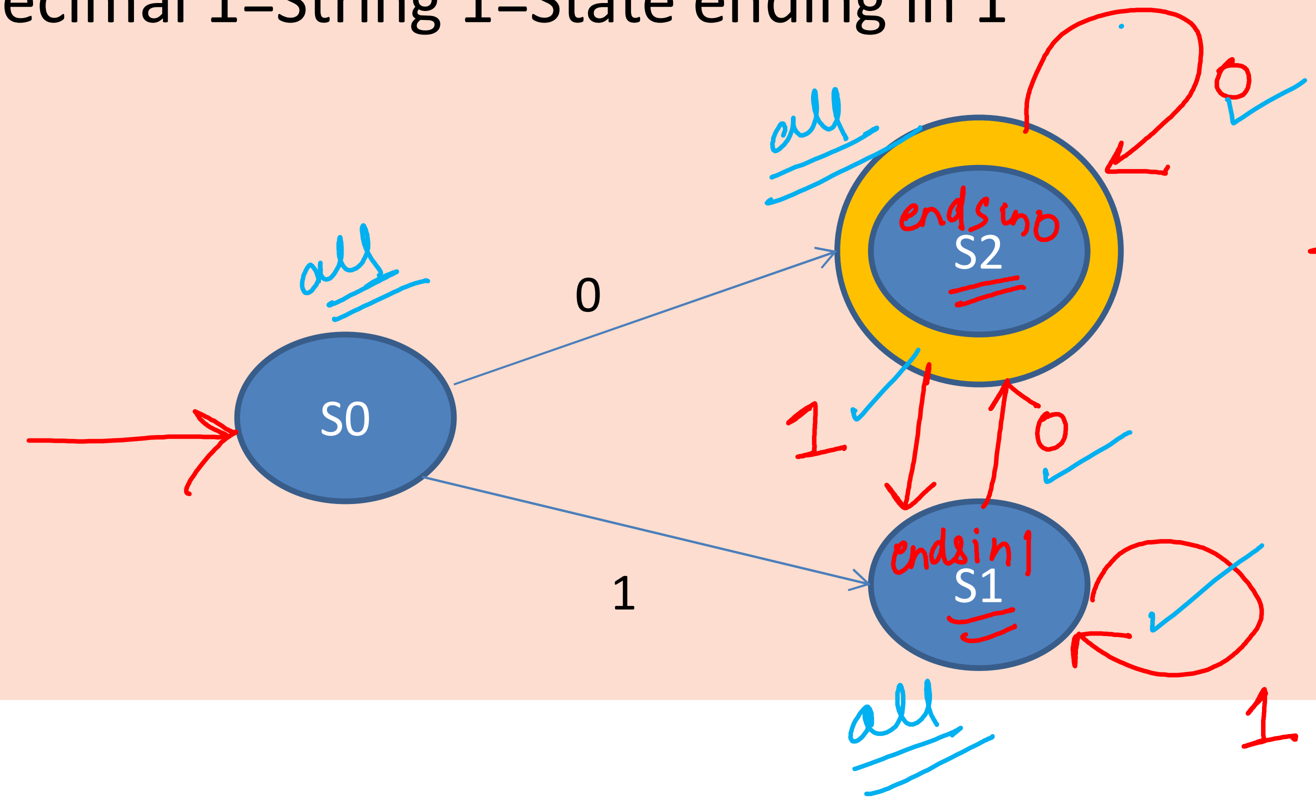
Example 5

String –

Decimal 0=String 0=State ending in 0

Decimal 1=String 1=State ending in 1

String 0 ⇒ ending 0
1



10 = Dec 2

11 = Dec 3

100 = Dec 4

101 = Dec 5

DESIGNING FINITE AUTOMATA

Example 5

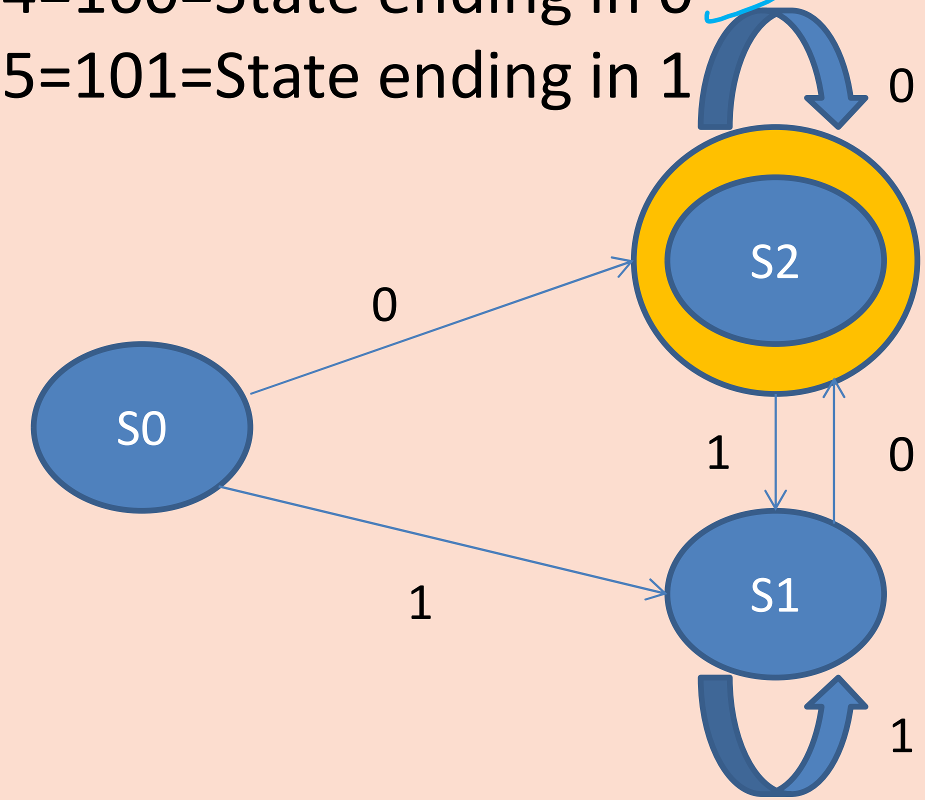
String –

Decimal 2=10=State ending in 0

Decimal 3=11=State ending in 1

Decimal 4=100=State ending in 0

Decimal 5=101=State ending in 1



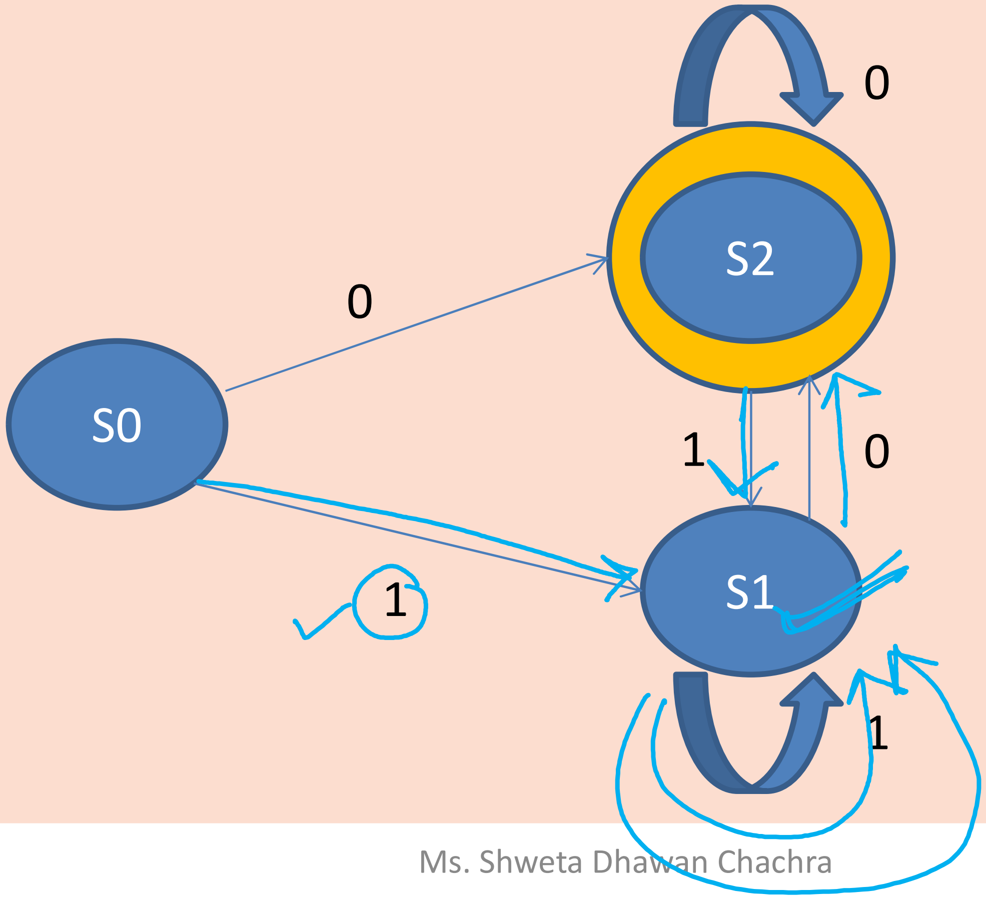
DESIGNING FINITE AUTOMATA

Example 5

Check for bigger numbers

Decimal 64= 1000000=State ending in 0

Decimal 27=11011=State ending in 1



Done

18/1/22

DESIGNING FINITE AUTOMATA

Example 6

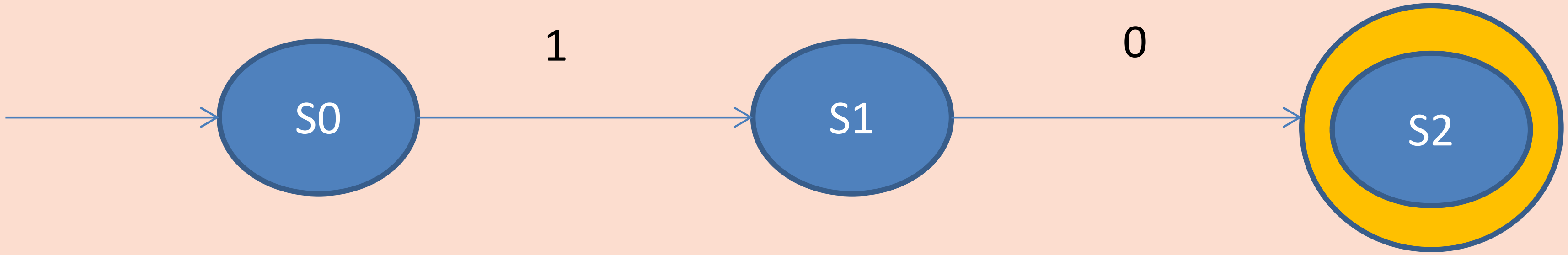
Design a DFA which accepts only those binary numbers which start with 1 and ends with 0.

DESIGNING FINITE AUTOMATA

Example 6

Design a DFA which accepts only those binary numbers which start with 1 and ends with 0.

Initial state S0
S0 reads 1, moves to S1
String 10
S1 reads 0, moves to S2=Final state



DESIGNING FINITE AUTOMATA

Example 6

String 1010

Also check for strings-

10

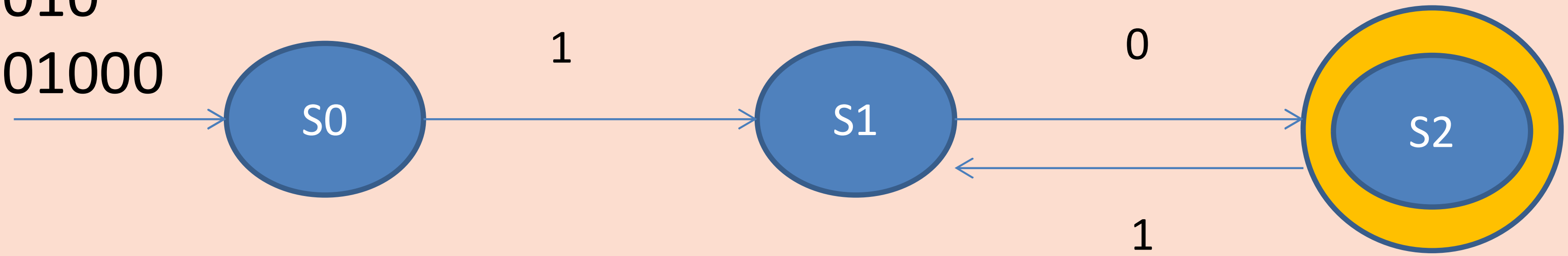
110

1110

1010

101010

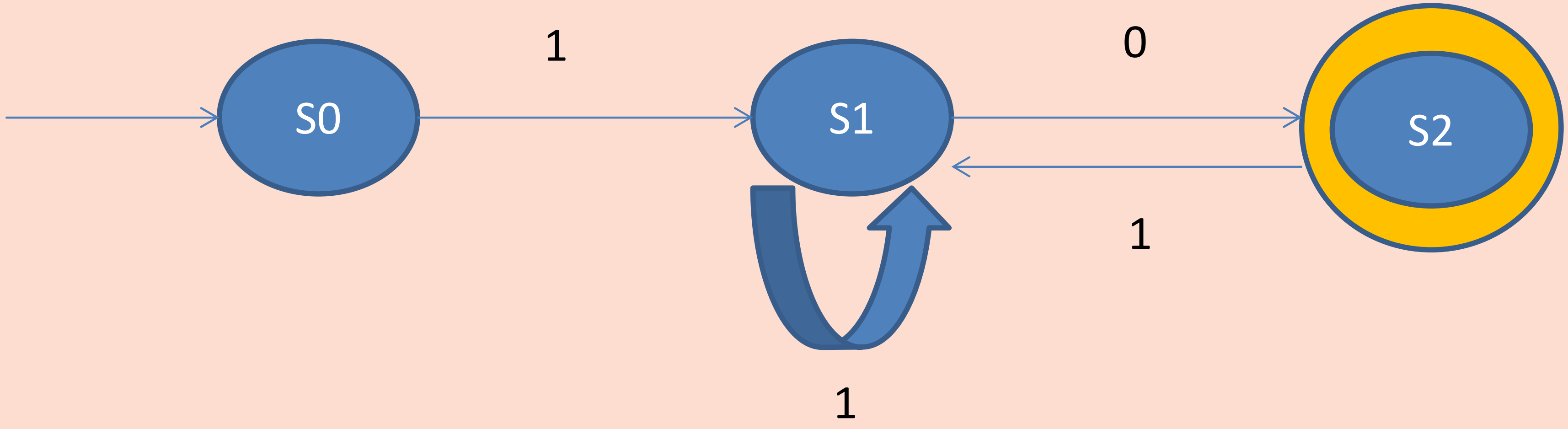
10101000



DESIGNING FINITE AUTOMATA

Example 6

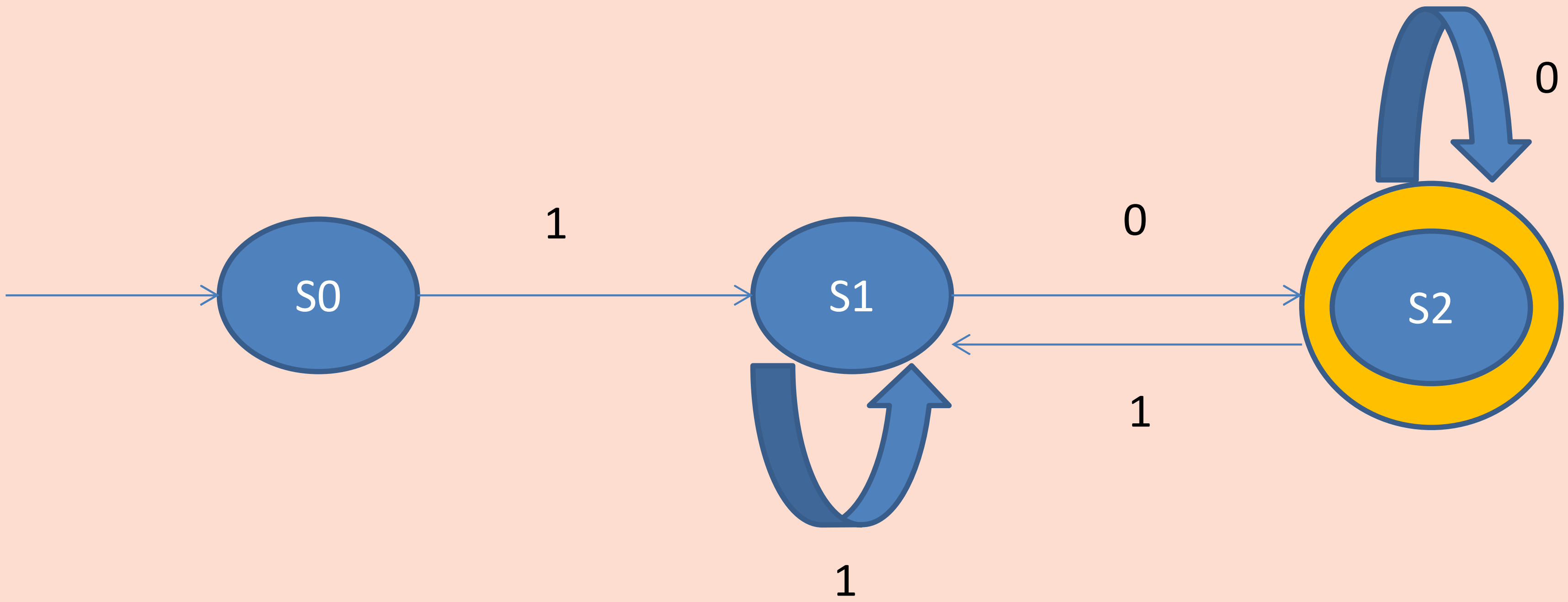
String 10110



DESIGNING FINITE AUTOMATA

Example 6

String 1000



DESIGNING FINITE AUTOMATA

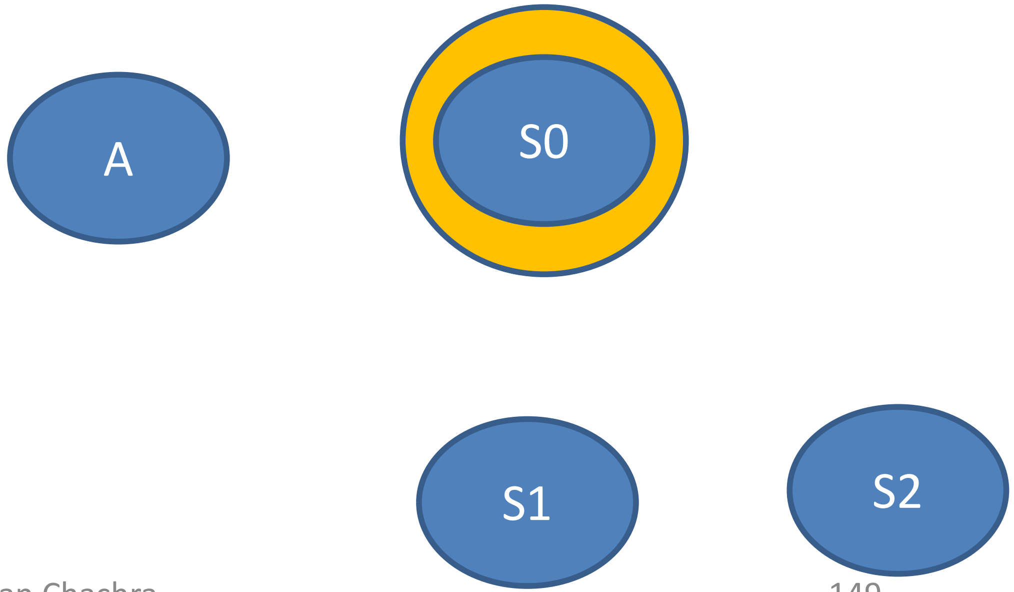
Example 7

Design a DFA which checks whether a given binary number is divisible by three

Example 7

- The possible Remainders =0,1,2
- Remainder 0=>Divisible by 3

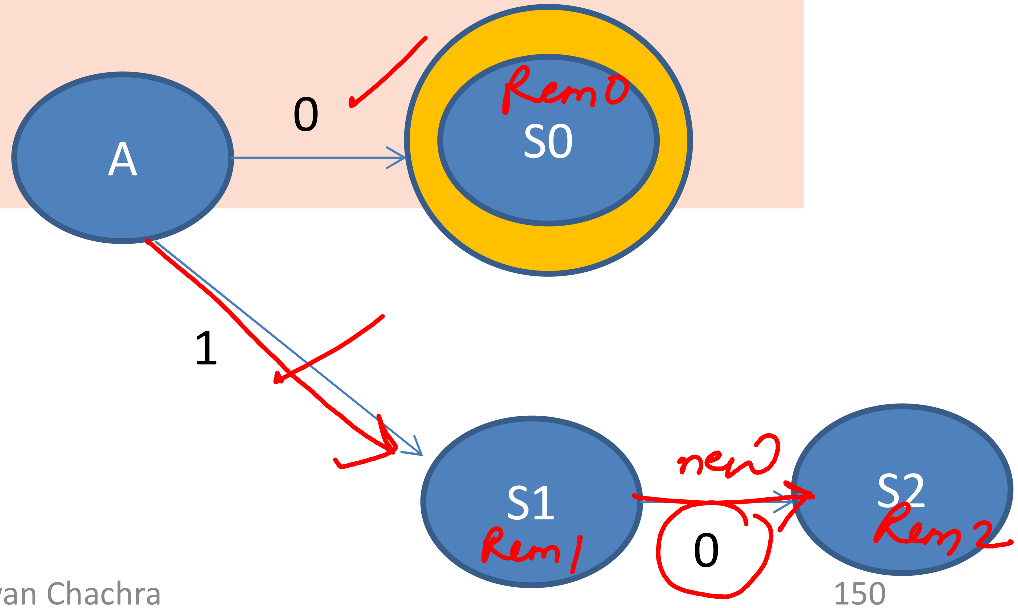
- Group Digits according to their remainder
  - Initial State=A
  - Remainder 0 group S0=>Final State
  - Remainder 1 group S1
  - Remainder 2 group S2



DESIGNING FINITE AUTOMATA

Example 7

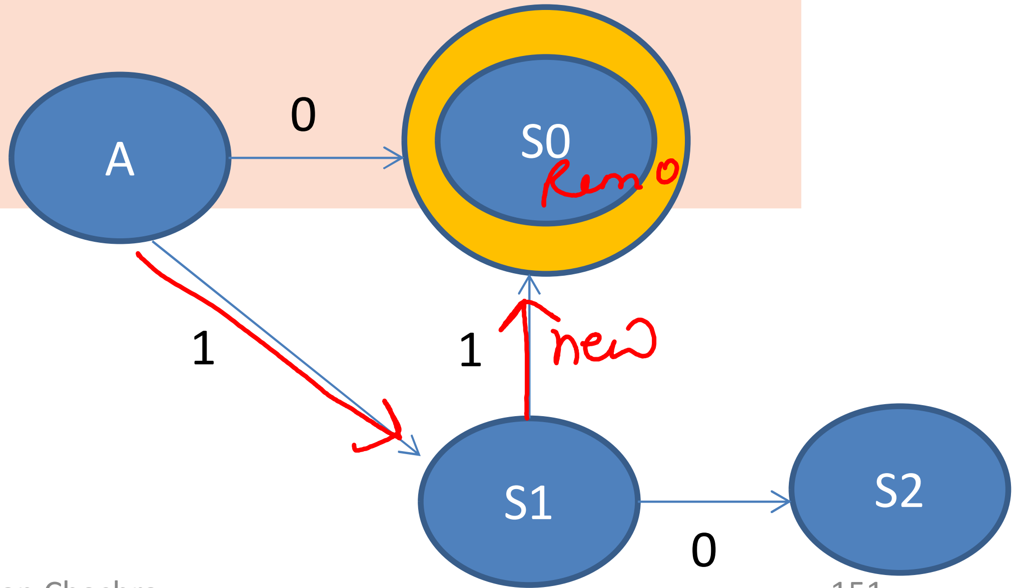
- Binary –Decimal =>
- 0-0 , Rem=0,State S0
- 01-1, Rem=1, State S1
- 10-2 , Rem=2, State S2



DESIGNING FINITE AUTOMATA

Example 7

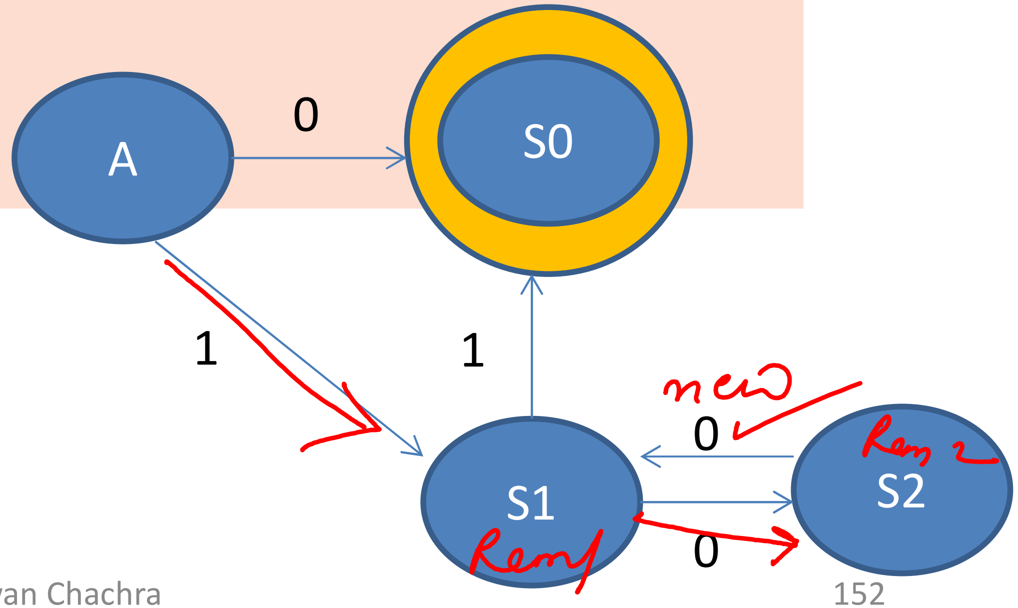
- Binary –Decimal =>
- 11-3 , Rem=0, State S0



DESIGNING FINITE AUTOMATA

Example 7

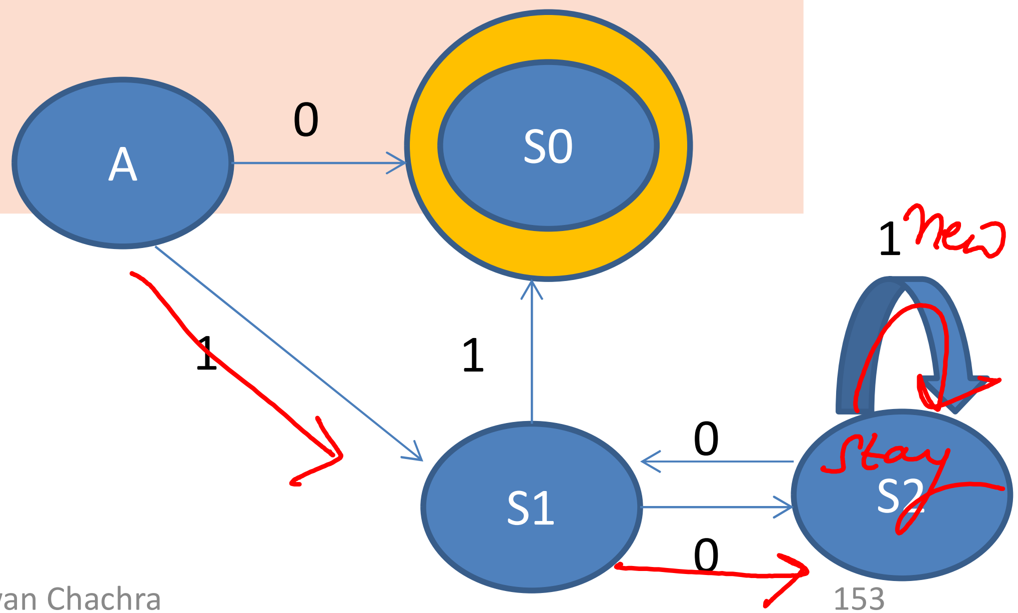
- Binary –Decimal =>
- 100-4 , Rem=1, State S1



DESIGNING FINITE AUTOMATA

Example 7

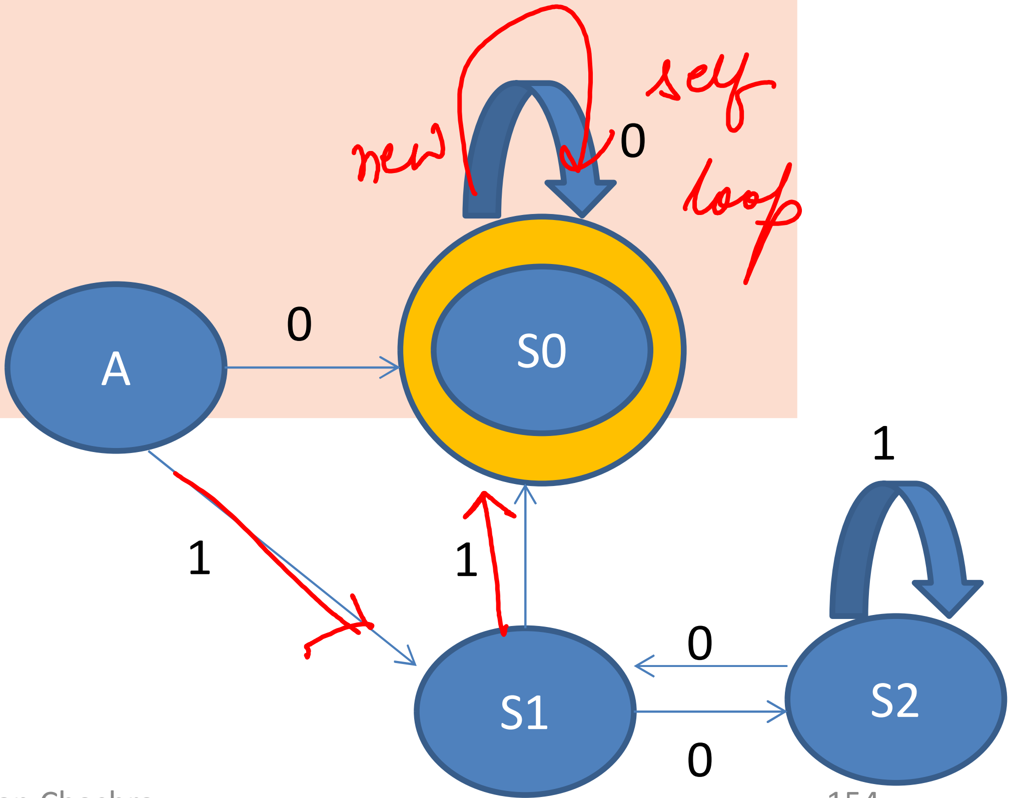
- Binary –Decimal =>
- 101-5 , Rem=2, State S2



DESIGNING FINITE AUTOMATA

Example 7

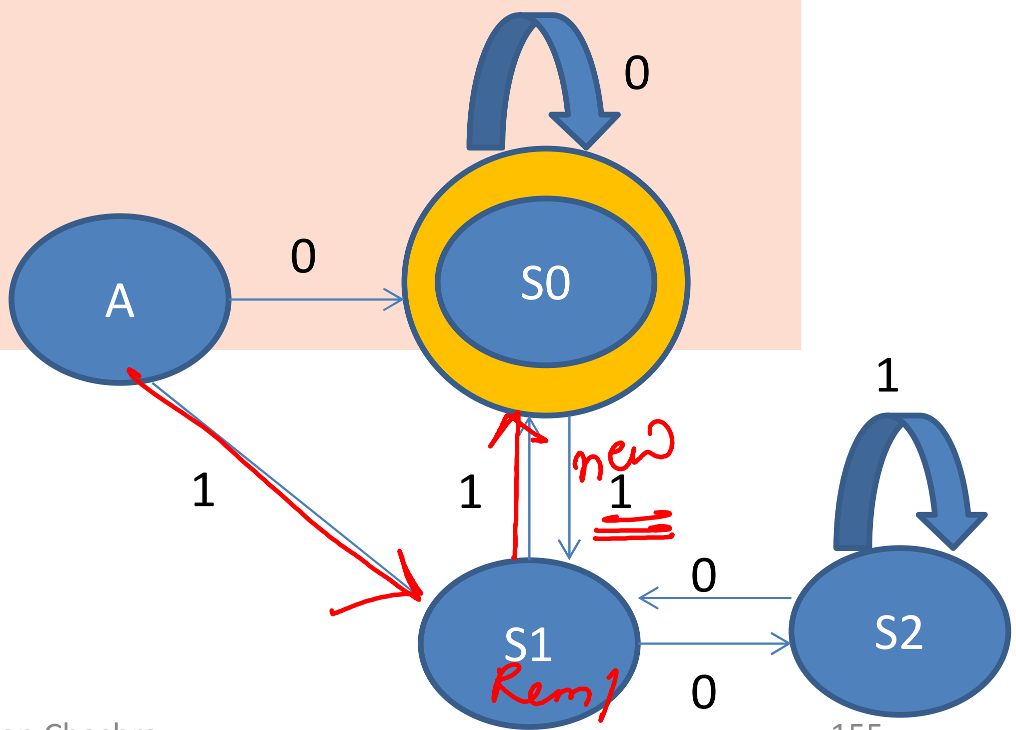
- Binary –Decimal =>
- 110-6 , Rem=0, State S0



DESIGNING FINITE AUTOMATA

Example 7

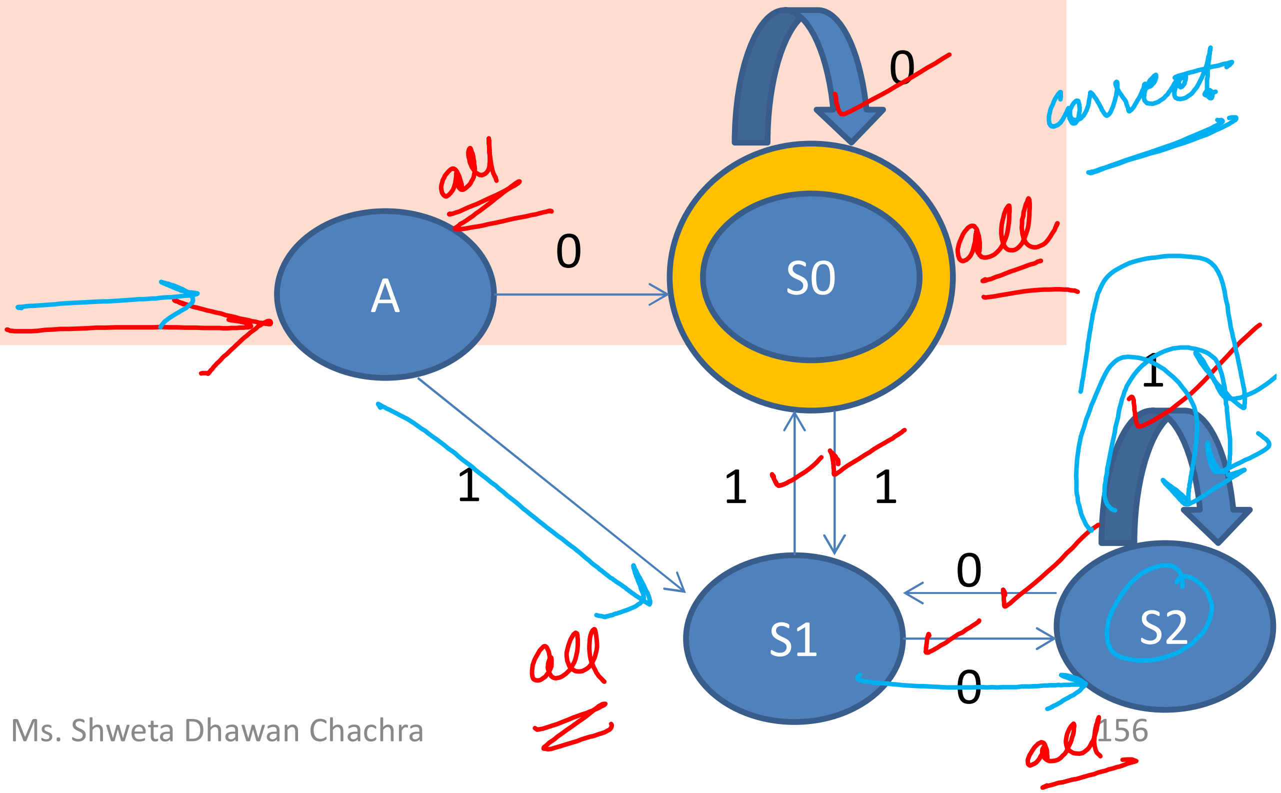
- Binary –Decimal =>
- 111-7 , Rem=1, State S1



DESIGNING FINITE AUTOMATA

Example 7

- For any Bigger Number
- Binary –Decimal =>
- 10111-23 , Rem=2, State S2
- Yes, Satisfied



DESIGNING FINITE AUTOMATA

Example 8

Design a DFA to accept a Language L where all strings in L are such that total number of a’s in them are divisible by 3

DESIGNING FINITE AUTOMATA

Example 8

- There is no condition that a should be in clumps
- But Total number of a's should be divisible by 3
- No conditions on b's
- b's are allowed in between

ababa ✓

count/3

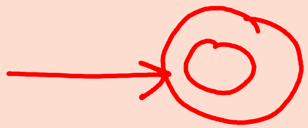
aaa

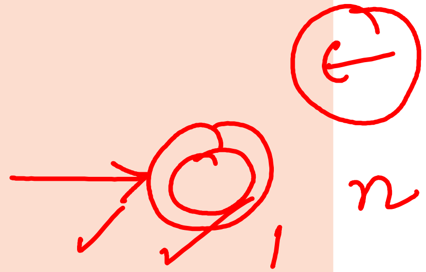
aaabaaabaaa

DESIGNING FINITE AUTOMATA

Example 8

- Total no of a’s should be multiples of 3 i.e
- Total no of a’s can be 0=>Initial state can be final state





$n \geq 0$

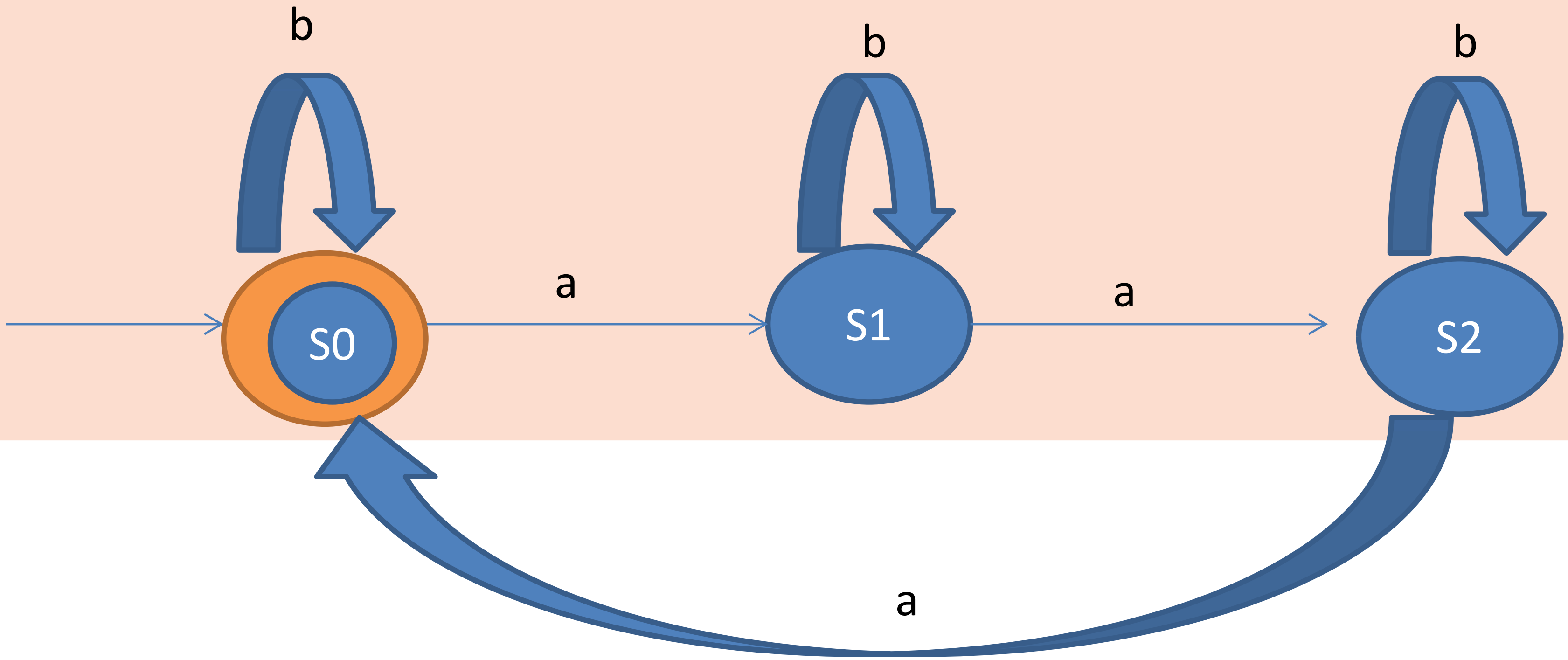
- Total no of a’s can be 3
- Total no of a’s can be 6
- Total no of a’s can be 9
- and so on…

- And whenever count of no of a’s is 0/3/6/9/12/15…so on
- The string should be accepted i.e. it should be final state

DESIGNING FINITE AUTOMATA

Example 8

Design a DFA to accept a Language L where all strings in L are such that total number of a's in them are divisible by 3



DESIGNING FINITE AUTOMATA

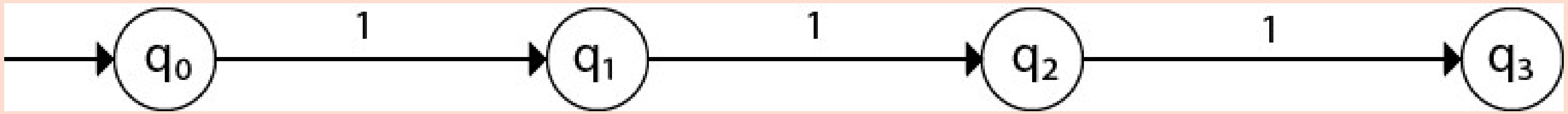
Example 9

Design a DFA to accept the language L(M) = {w | w ε {0, 1}*} and W is a string that does not contain three consecutive 1's.

DESIGNING FINITE AUTOMATA

Example 9

- When three consecutive 1's occur the DFA will be:

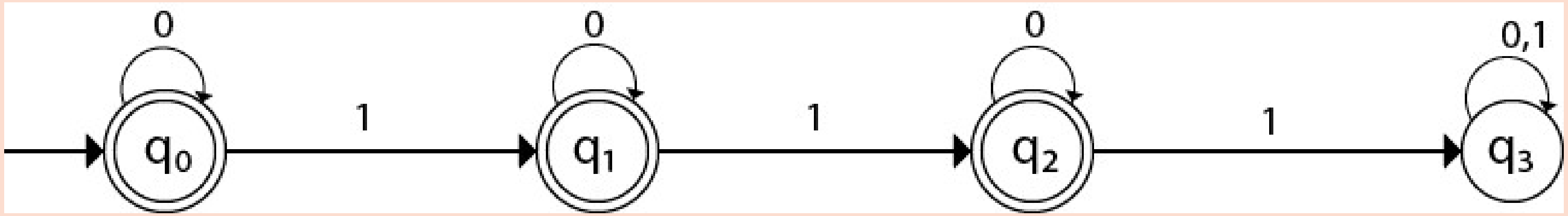


- Here two consecutive 1's or single 1 is acceptable, hence

- The stages q0, q1, q2 are the final states.

DESIGNING FINITE AUTOMATA

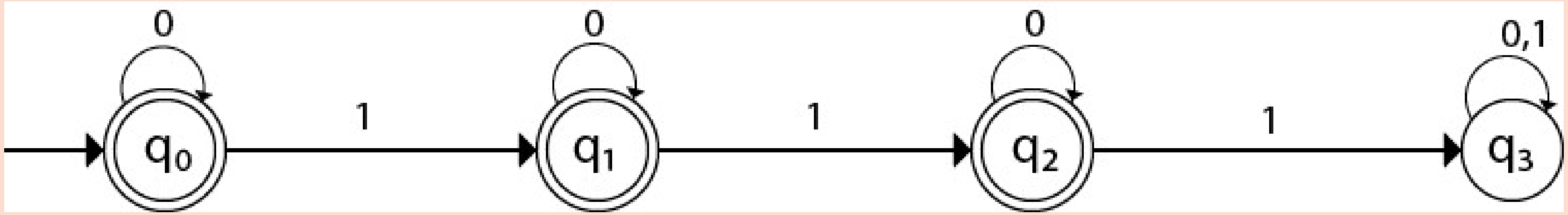
Example 9



- String 111 will be in q3 state=Non final state=Rejected
- String 10110111 will be in q3 state=Non final state=Rejected

DESIGNING FINITE AUTOMATA

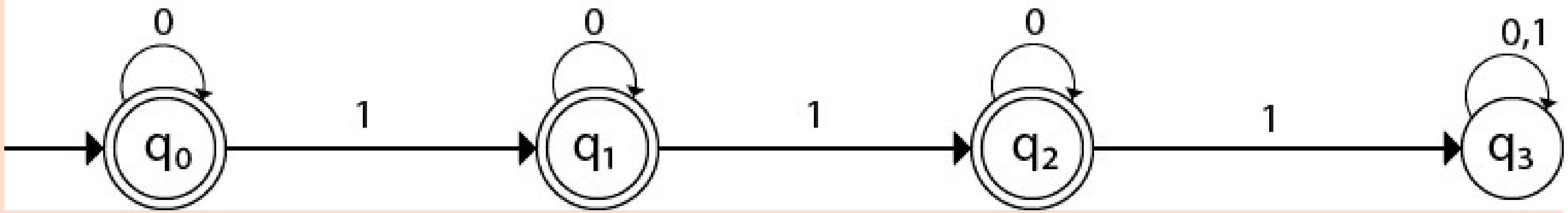
Example 9



- String 0110 accepted
- Strings 1011101 rejected
- Correct till now……

DESIGNING FINITE AUTOMATA

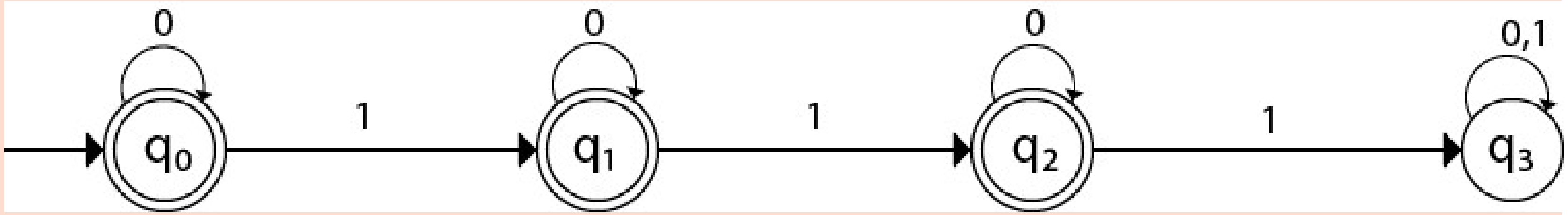
Example 9



- Exactly three 1’s but not consecutive giving Error-
- String 1101 rejected
- String 11010 rejected
- String 10110 rejected
- String 1011 rejected
- The above strings should be accepted as the 1’s are not consecutive
- Strings 110110 rejected-No of 1’s increases to 4=>Works perfectly

DESIGNING FINITE AUTOMATA

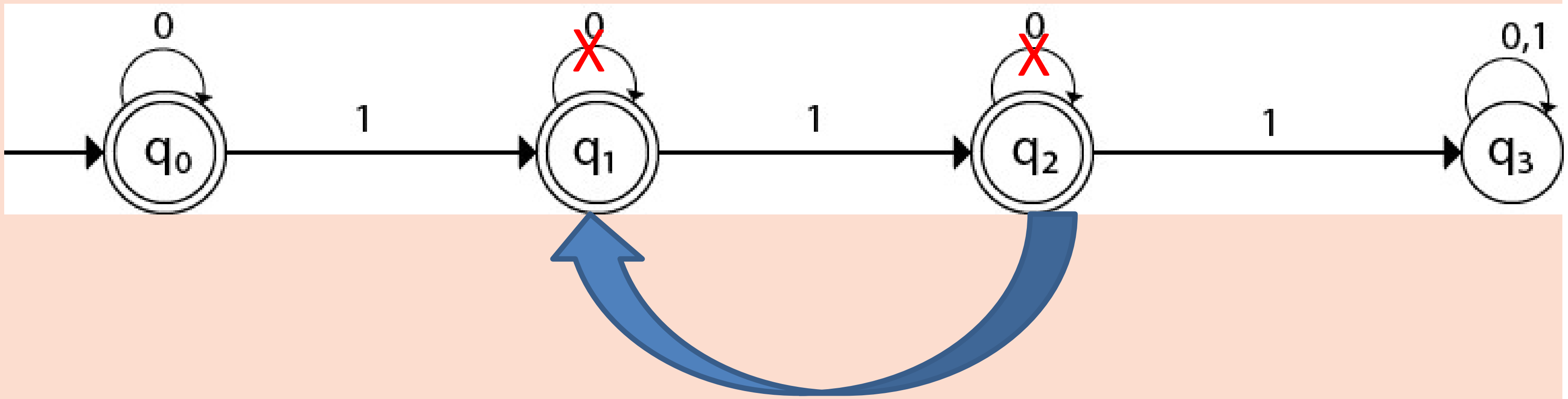
Example 9



- Change to be done-
- Change Outgoing transition from q1 and q2

DESIGNING FINITE AUTOMATA

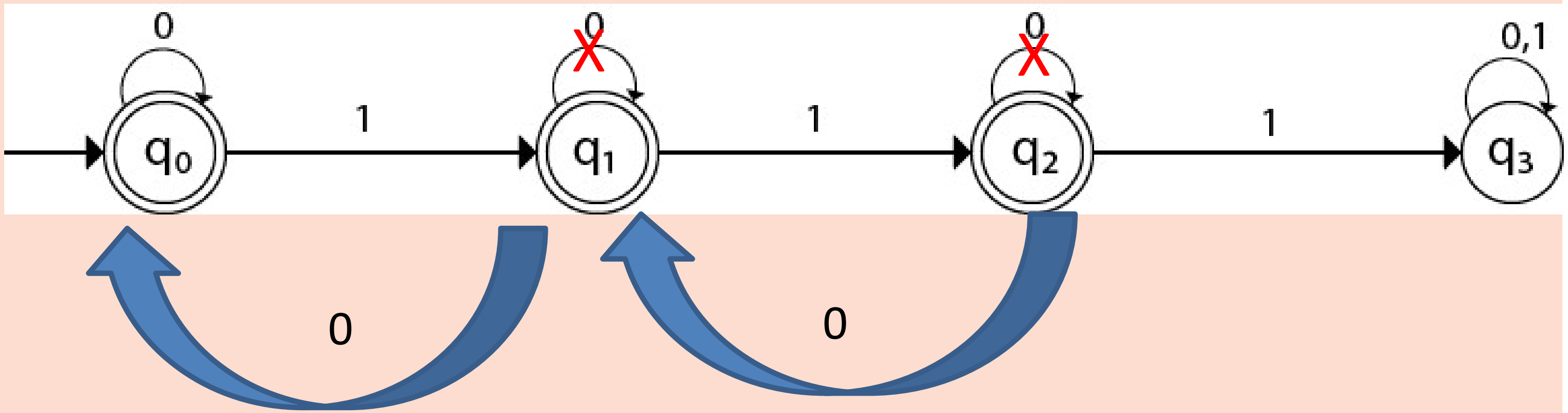
Example 9



- Change to be done-
- Change Outgoing transition from q1 and q2
- For Strings 1101,11010 –
- Outgoing transition for 0 from q2 should go to q1
- so that strings like 11011 should be rejected

DESIGNING FINITE AUTOMATA

Example 9



- Change to be done-
- Change Outgoing transition from q1 and q2
- For Strings like 10110, 1011
- Outgoing transition for 0 from q1 should go to q0
- so that strings like 101101 should be rejected

DESIGNING FINITE AUTOMATA

Example 10

Design a DFA to accept the language $L(M) = \{0^m 1^n \mid m>=0 \text{ and } n>=1\}$

DESIGNING FINITE AUTOMATA

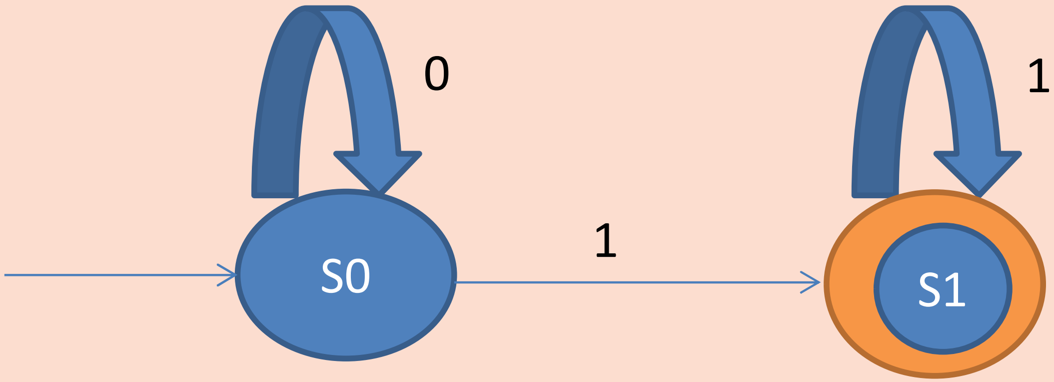
Example 10-Design a DFA to accept the language L(M) = $\{0^m 1^n \mid m>=0 \text{ and } n>=1\}$

- All 0s are followed by 1s
- No of zeros can be 0
- No of ones has to be minimum 1, so one transition for 1 has to be there

DESIGNING FINITE AUTOMATA

Example 10-Design a DFA to accept the language L(M) = $\{0^m1^n \mid m>=0 \text{ and } n>=1\}$

- No of zeros can be 0=>can be associated with start state
- No of ones has to be minimum 1, so one transition for 1 has to be there=>final state

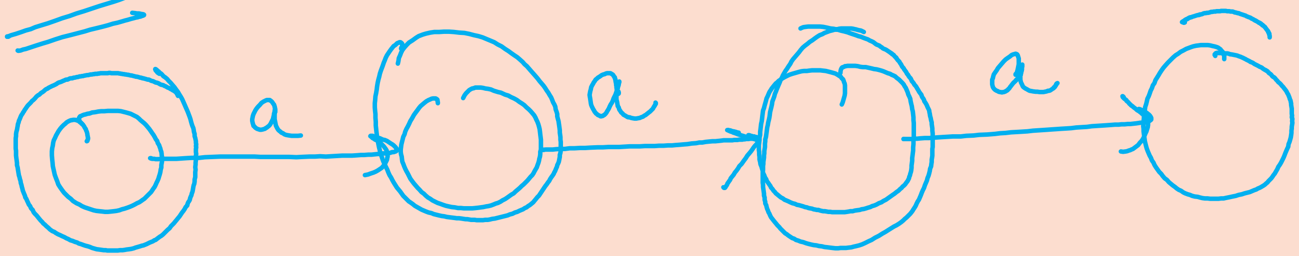


DESIGNING FINITE AUTOMATA

Example 11

Design a DFA which accepts all the strings not having more than 2 a's over ∑={a,b}

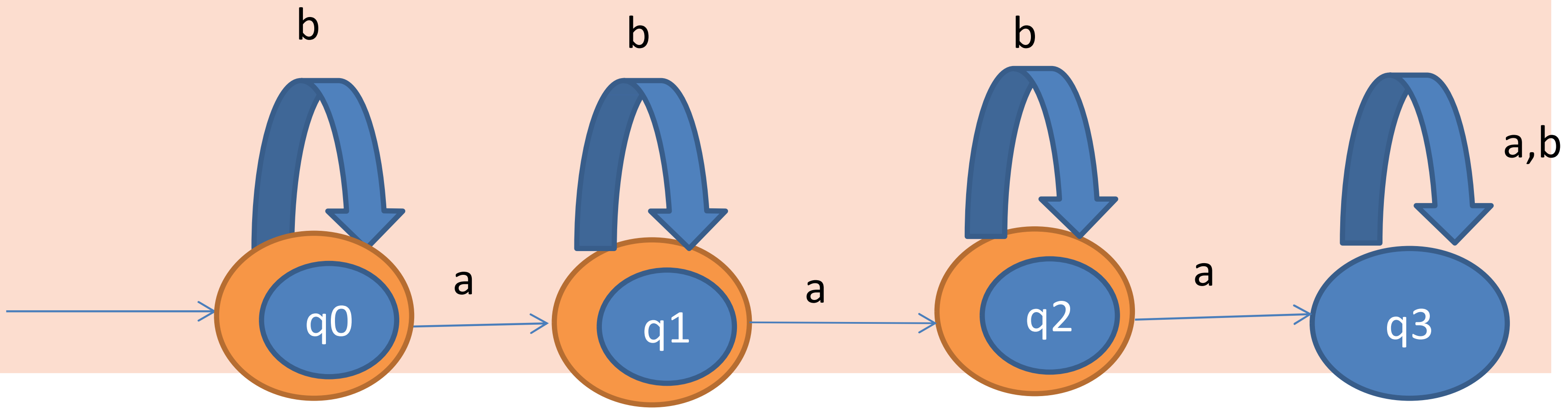
similar 3' 1's q



DESIGNING FINITE AUTOMATA

Example 11

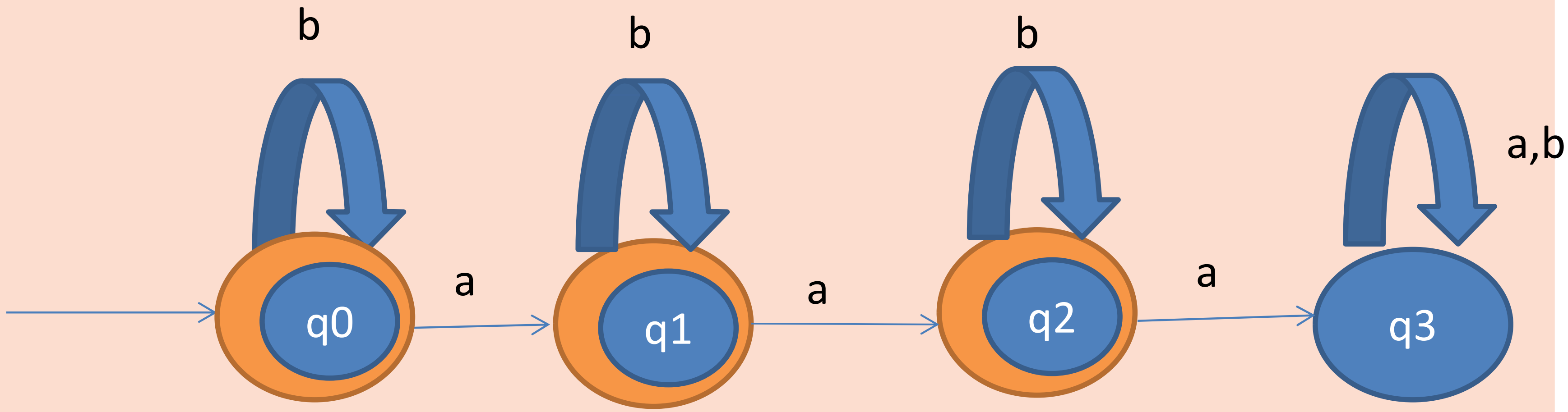
Design a DFA which accepts all the strings not having more than 2 a's over ∑={a,b}



DESIGNING FINITE AUTOMATA

Example 11

String aaba-Rejected
String bba-Accepted
String baba –Accepted
String aabbab-Rejected



DESIGNING FINITE AUTOMATA

Example 12 Try

Recognize the language given by following DFA

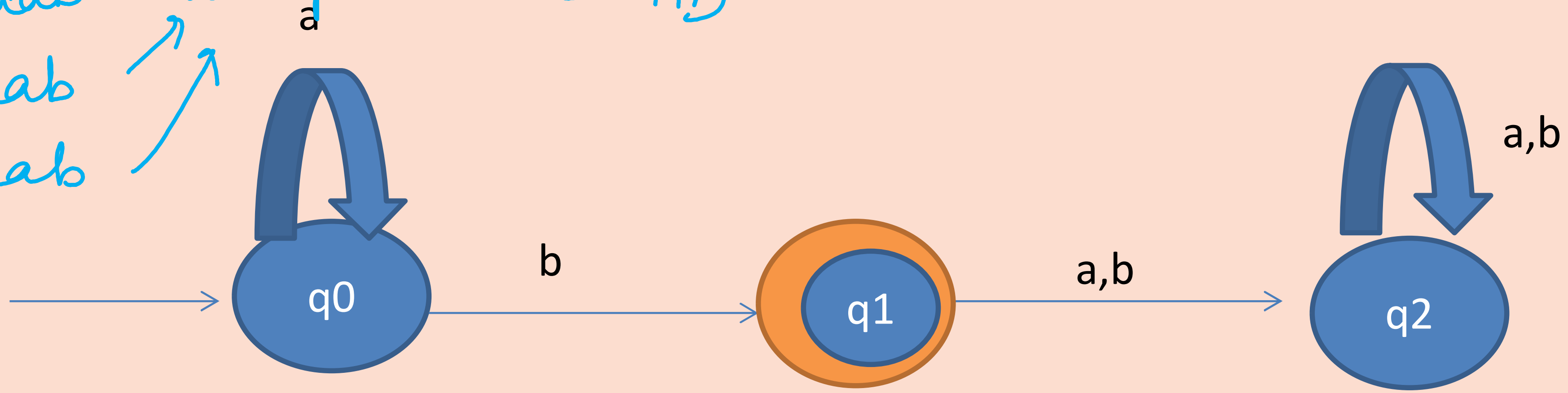
b → 0 a's → 1b

ab → accepted → 1 a's → 1b

aab → accepted → 2's → 1b

aaab

aaaab



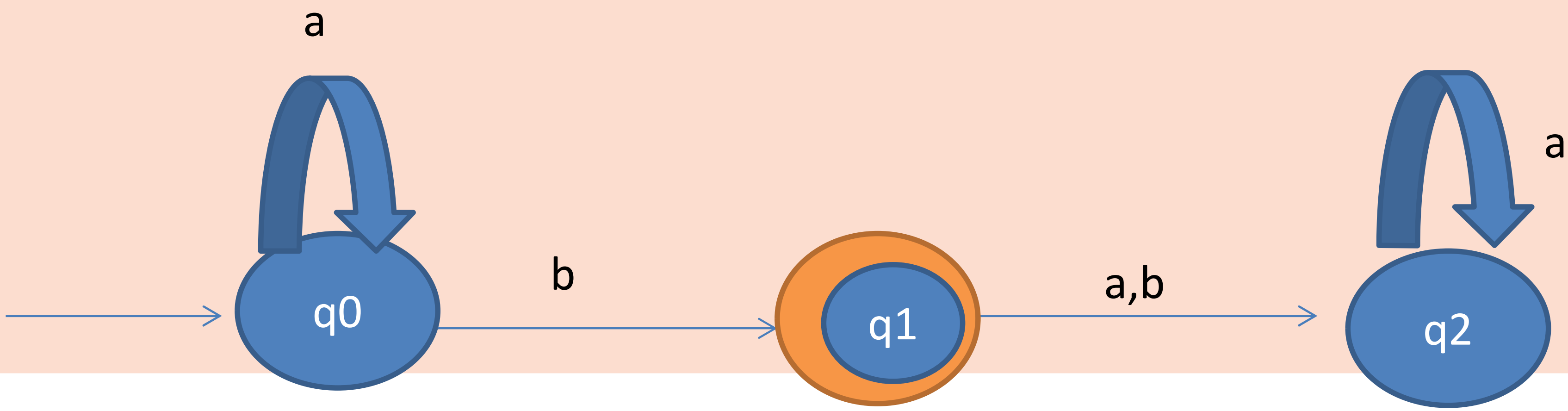
Recognizing lang from DFA

~~DESIGNING FINITE AUTOMATA~~

Example 12-Recognize the language given by following DFA

$\Rightarrow L = \{a^n b \mid n \geq 0\}$

any occurences of a, 1 occurence of b



DESIGNING FINITE AUTOMATA

Example 13

Design a DFA over ∑={a,b} for $(ab)^n$ for n>=0

DESIGNING FINITE AUTOMATA

Example 14

Design a DFA over $\sum=\{a,b\}$ for $(ab)^n$ for $n>=1$

DESIGNING FINITE AUTOMATA

Example 15

Design a FA with ∑ = {0, 1} accepts the only input 101.

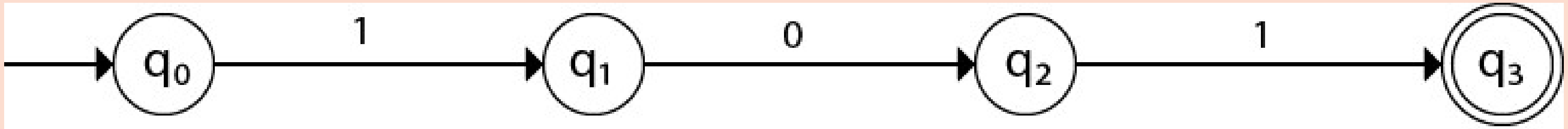


Fig: FA