**Regular Languages**: is a language that can be expressed with a regular expression or a deterministic or non-deterministic finite automata or state machine. We call a language regular if it can be decided if a word is in the language with an algorithm/a machine with constant (finite) memory by examining all symbols in the word one after another.

**Finite Automata**: models for computers with an extremely limited amount of memory

**Example1(Door)**:

*The automatic door*: We want it to be closed/open when a person is standing on the pad.

**States of the Door**: These figures show you the states of the door when someone could be on the rear or front pad and how the door is suppose to be controlled.
Example 2 (Toll Gate):
- Costs 25 cents
- Open/close
- State transition diagram - we can assume this takes nickels, dimes and quarters
- 3 bits of memory needed to know what state we’re in

 Deterministic finite automata: Deterministic means that it can only be in, and transition to, one state at a time (i.e. for some given input). A finite-state machine that accepts and rejects strings of symbols and only produces a unique computation (or run) of the automaton for each input string

Example 3:

- Start state (q₁)
- Accept state-double circle (q₂)
- Transitions-arrows
- Output is either accept or reject. We read symbols from input string (one by one, left to right)

Since the state transition diagram above has q₂ as the accept state, it accepts any string of at least one “1” and ends with a “1” or an even number of “0’s”.

A finite automation is a 5-tuple where (Q, Σ, S, q₀, F)

1. Q is a finite set of states
   Q = {q₁, q₂, q₃}
2. Σ is a finite set called the alphabet
   Σ = {0, 1}
3. S is a state transition. $Q \times \Sigma \rightarrow Q$ is the transition function

$S(q_1, 1) = q_2$

“I’m at $q_1$, and I get a 1, so I go to $q_2$. Defined in table

<table>
<thead>
<tr>
<th>$q$</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_1$</td>
<td>$q_1$</td>
<td>$q_2$</td>
</tr>
<tr>
<td>$q_2$</td>
<td>$q_3$</td>
<td>$q_2$</td>
</tr>
<tr>
<td>$q_3$</td>
<td>$q_2$</td>
<td>$q_2$,</td>
</tr>
</tbody>
</table>

4. $q_0 \in Q$ is the start state. In every state diagram you have to specify where to start

5. $F \subseteq Q$ is the set of accept states

$F = \{ q_2 \}$

If $A$ is the set of all strings that Machine $M$ accepts, $A$ is the language of machine $M$:

$L(M) = A$

$L(m) = \{w \mid w \text{ is a string over } \Sigma \text{ and } M \text{ accepts } -w\}$

$L(M) = \{w \mid \Sigma = \{0,1\} \text{ and } w \text{ contains at least one } 1 \text{ and ends in a } 1 \text{ or an even number of } 0\text{'s}\}$

**Example (M2):**

![Example M2 Diagram]

$L(M_2) = \{w \mid \text{w ends in 1}\}$

**Example (M3):**

![Example M3 Diagram]

$L(M_3) = \{w \mid \text{w is the empty string or it ends in 0}\}$

There will always be an accept state, since if there wasn’t, the language would be empty.
Example ($M_4$):

![ FSM diagram ]

$L(M_4) = \{ w \mid w \text{ starts and ends with the same letter from } \Sigma \}$

Definition: A language $A$ is called regular, if $\exists$ a finite automation $M$ S.T $L(M) = A$

Prove this is a regular $A = \{ w \mid w \text{ is a binary string containing an odd # of 1's} \}$

$M = \{ Q, \Sigma, \delta, q_1, F \}$
$Q = \{ q_1, q_2 \}$
$\Sigma = \{ 0, 1 \}$
$\delta = \text{ reference diagram}$
$F = \{ q_2 \}$