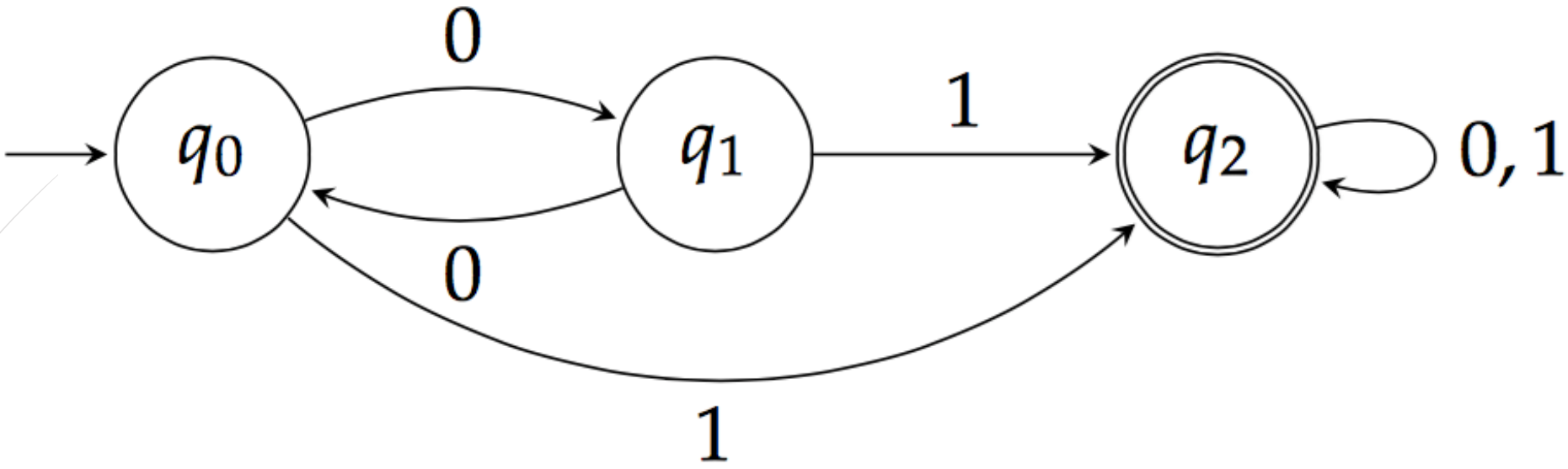
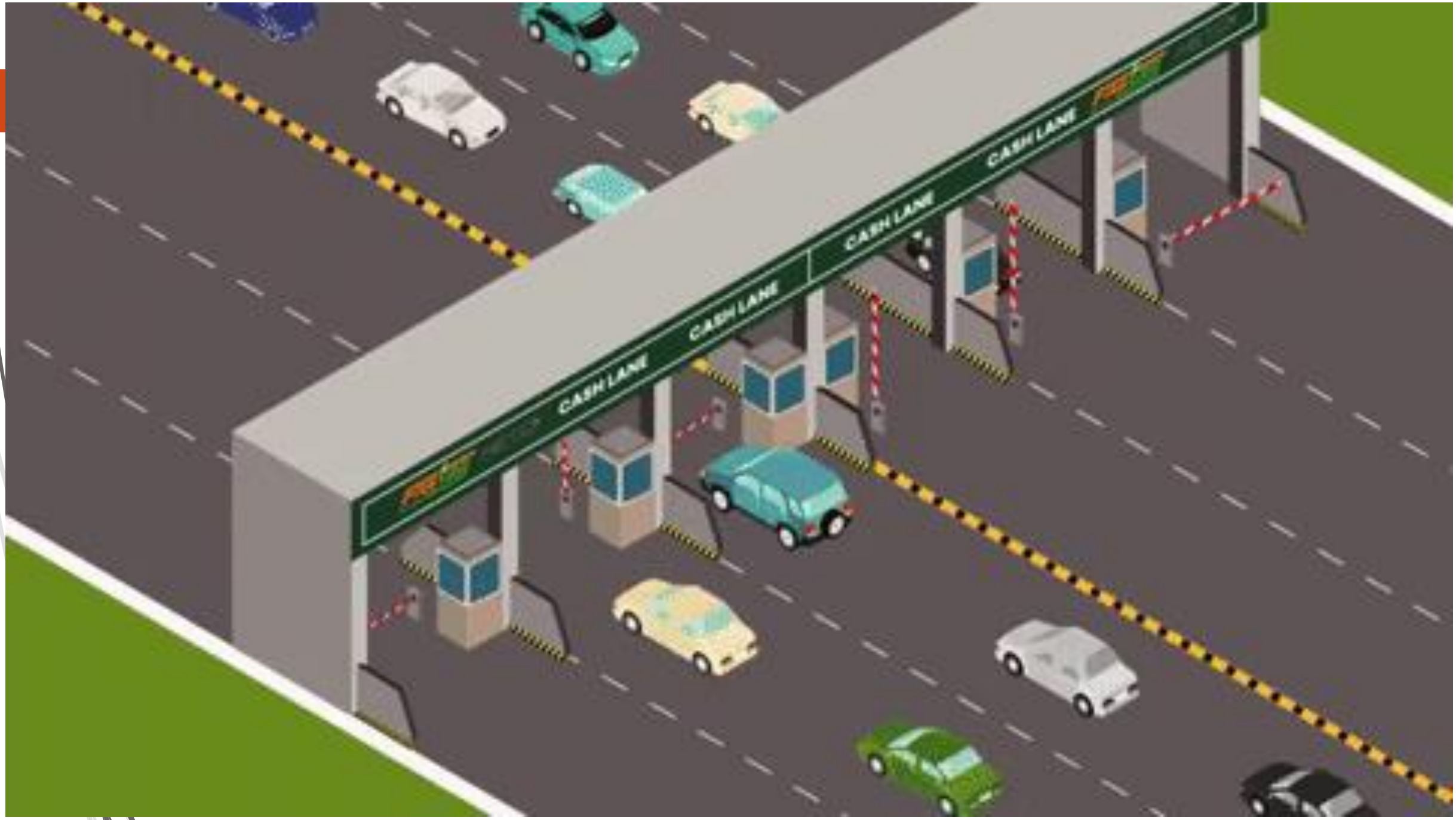


Theory of Computation

Lesson 1

Deterministic Finite State Automata

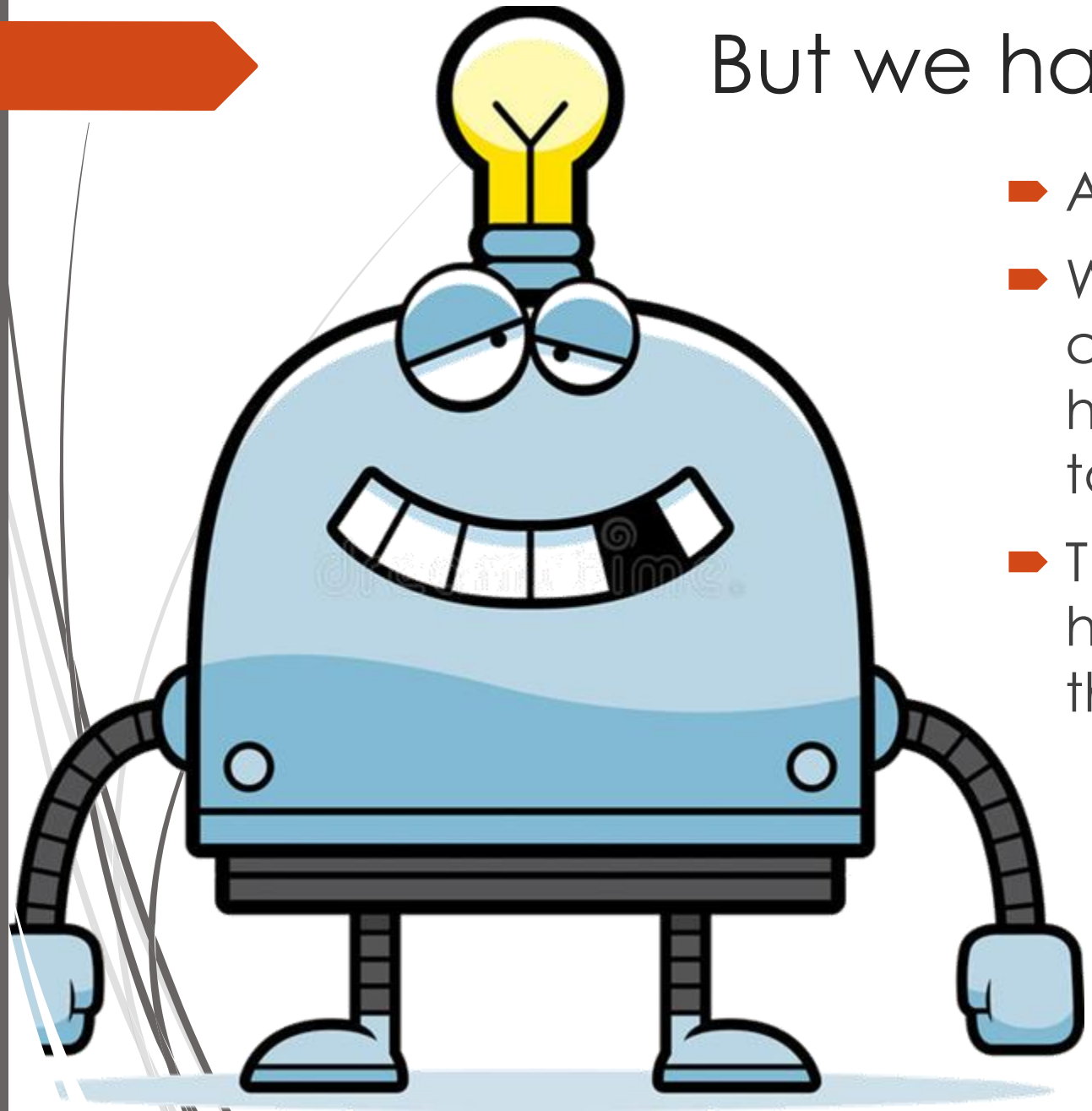

CASH LANE
CASH LANE
CASH LANE
CASH LANE

Imagine opening the gate with 25 cents.

Each coin will now represent the letters that make up a word for us.

- 5 + 5 + 5 + 5 + 5 ----> AAAAA
- 5 + 10 + 10 ----> ABB
- 5 + 5 + 5 + 10 ----> AAAB
- …
- 10 + 10 + 5 ----> BBA
- 25 ----> C

Now we must figure out which words should be accepted.

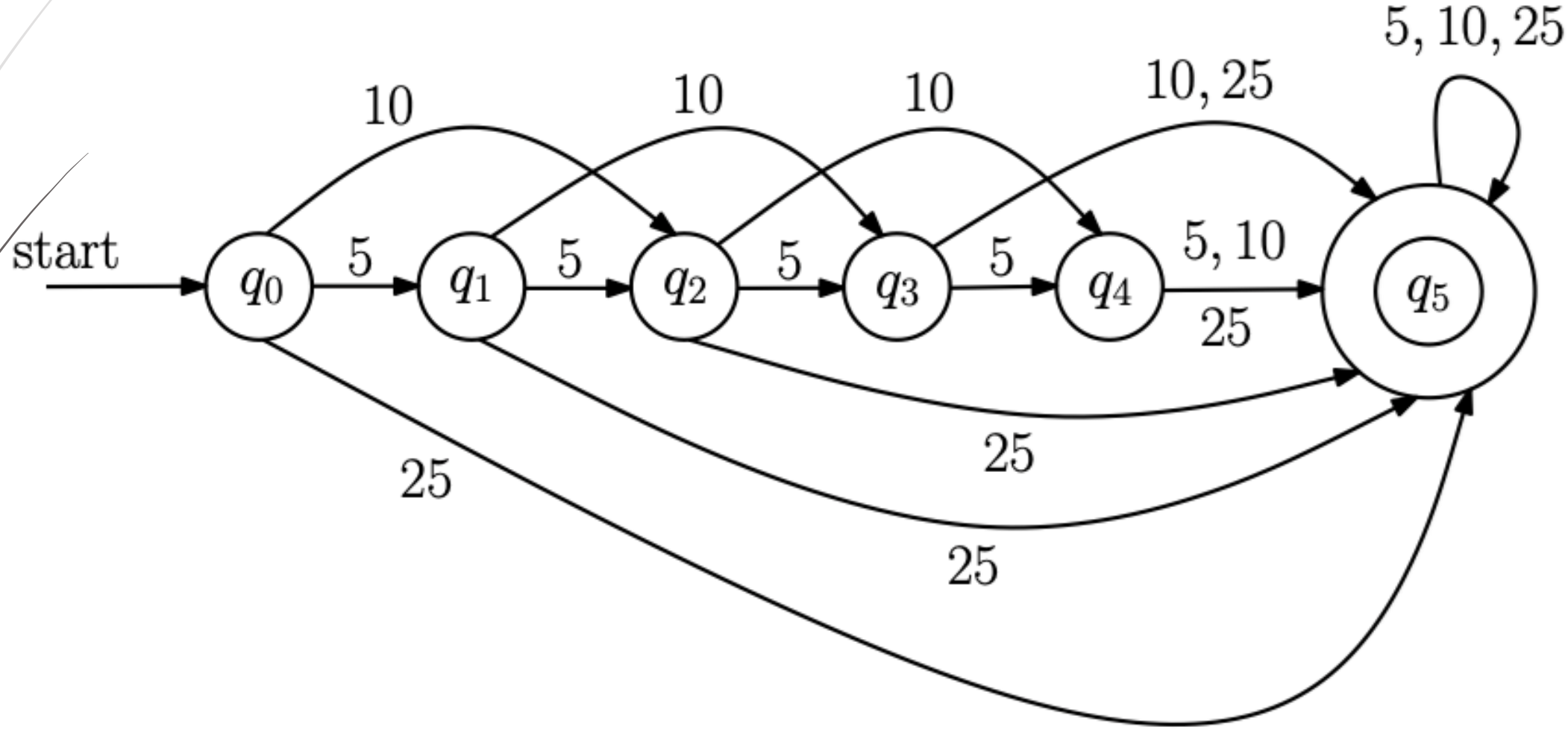


But we have a problem

- Automaton systems have no memory!
- When the user puts to machine a new coin, the system cannot remember how much coin has been thrown in total until that moment.
- Therefore, the system cannot calculate how much more coins needs to be thrown to open the gate.

Instead of Variables

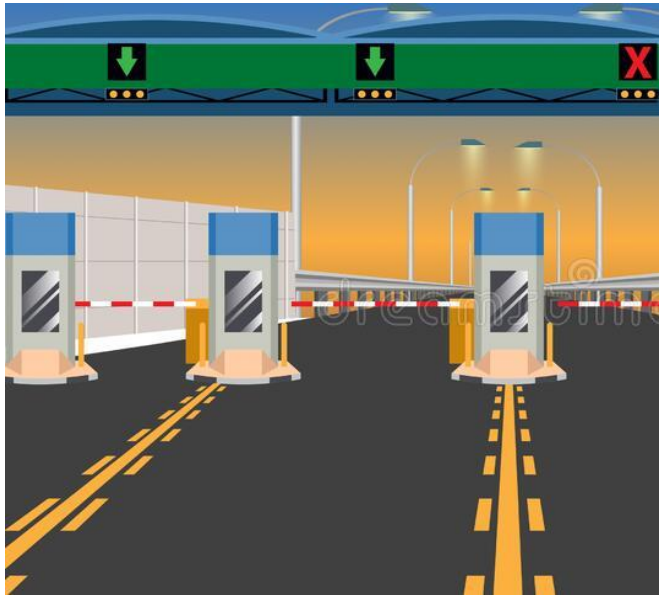
Finite State Automata use **states** to remember situations.



To save the world
We need Transformers

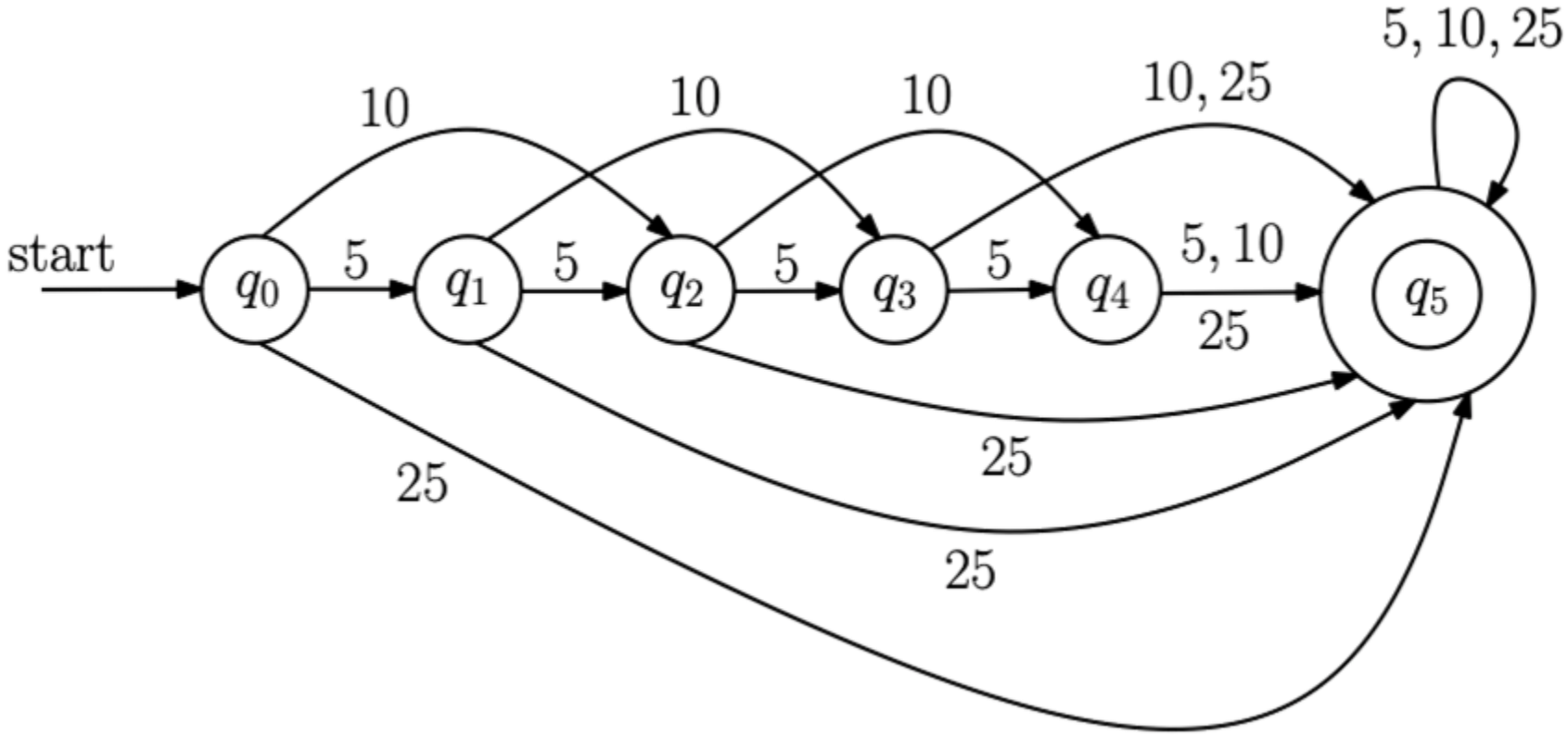
If we want to solve the toll-gate problem with finite state automata, we should see the coins thrown to open the gate as letters on the keyboard.

We need a Transformation

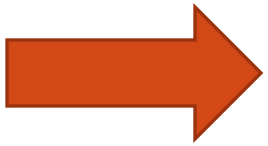


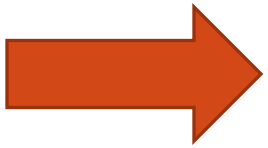
States

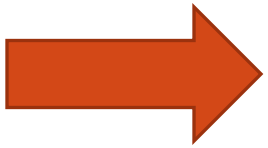
q_0 : there is no money at the automaton yet
q_1 : we have 5 cents only
q_2 : 10 cents
q_3 : 15 cents
q_4 : 20 cents
q_5 : 25 cents

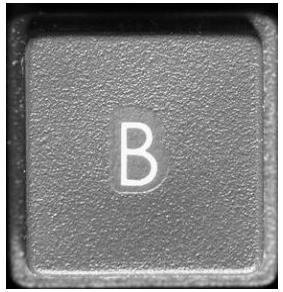


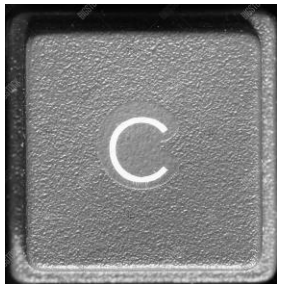
Alphabet







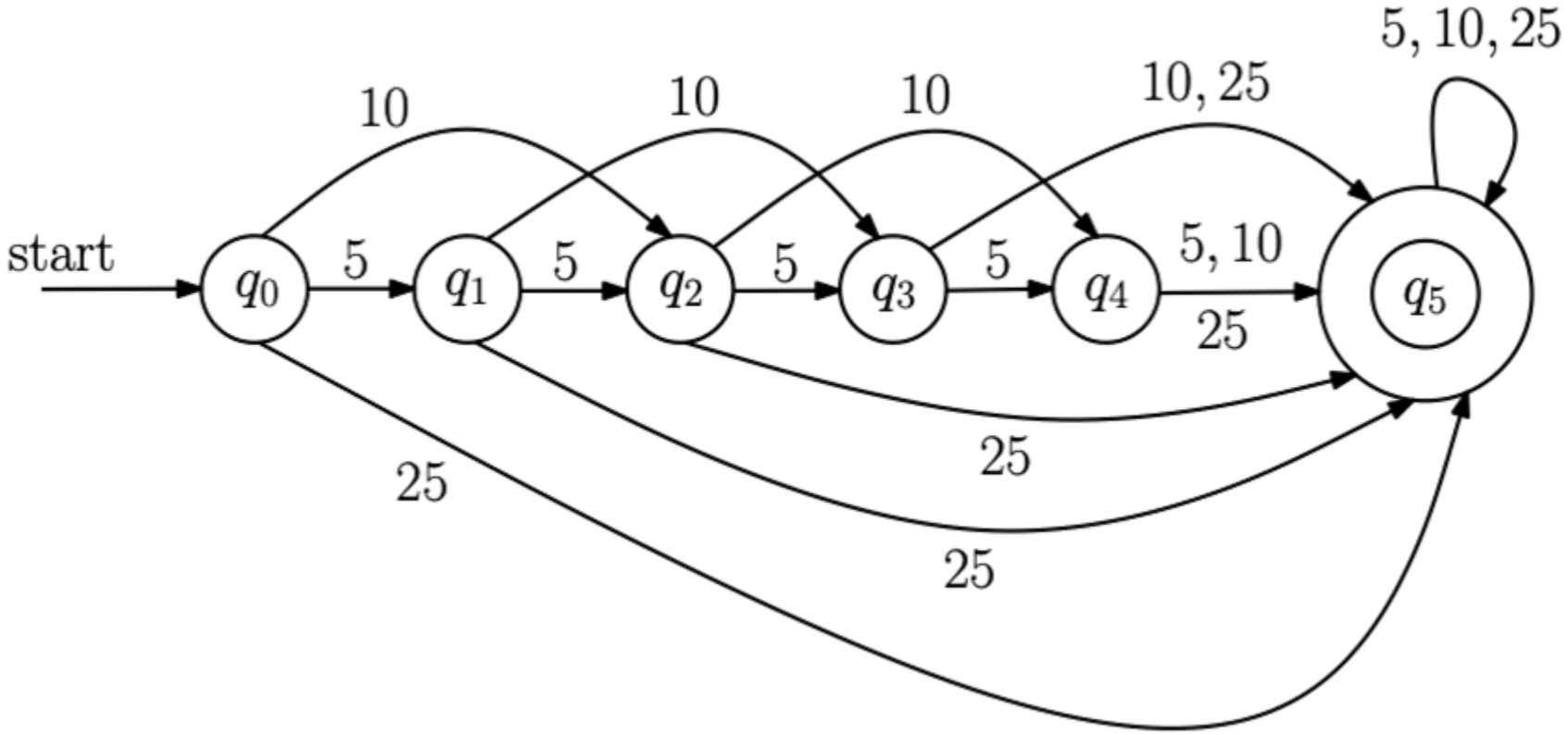




Since each coin the user throws will correspond to a letter, we must first define an alphabet and map which symbol each coin corresponds to.

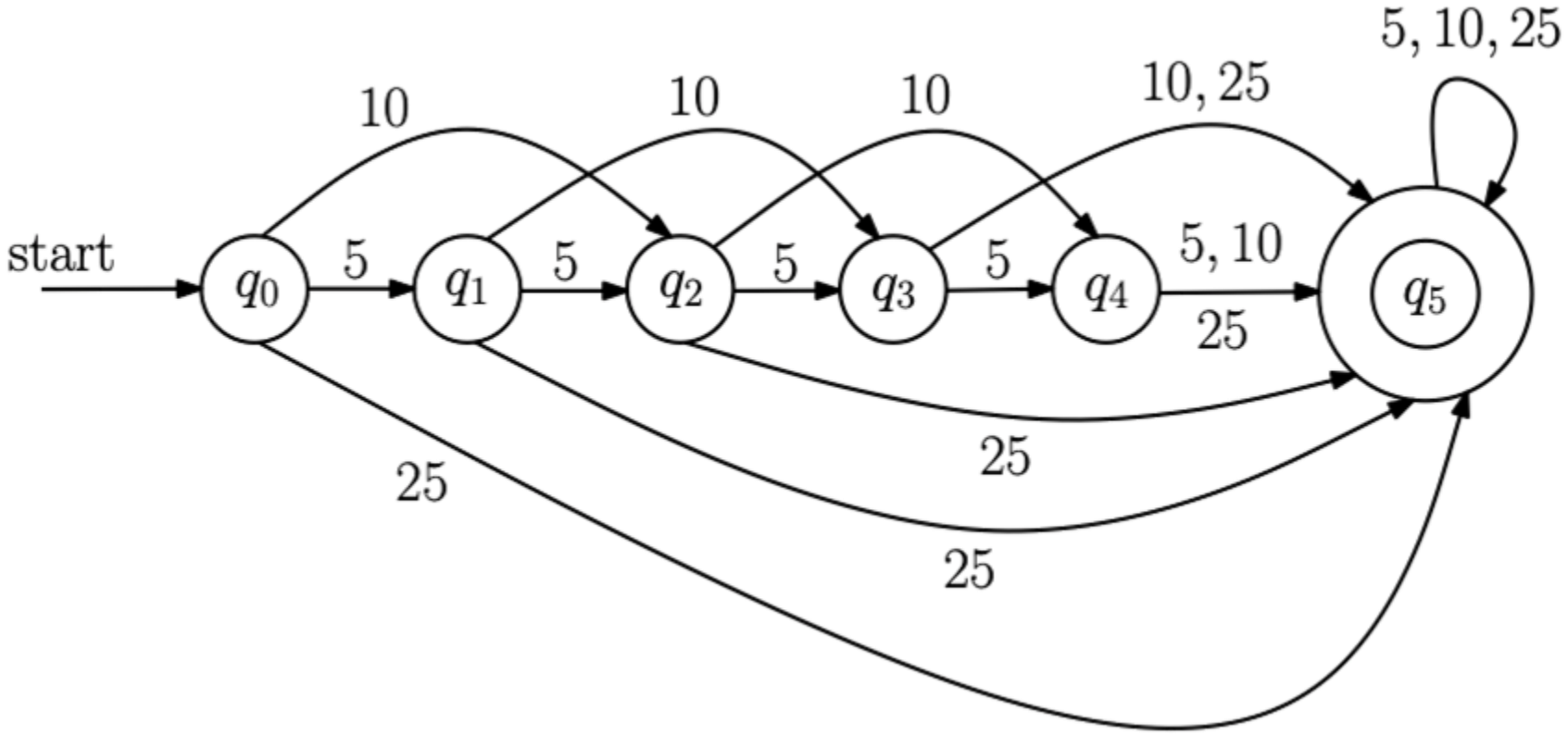
Transitions

We have to represent each symbol in our alphabet as an arrow going out of each state. Like the others, there should be three edges that come out of q_0: 5, 10, and 25 cents (because of our alphabet)



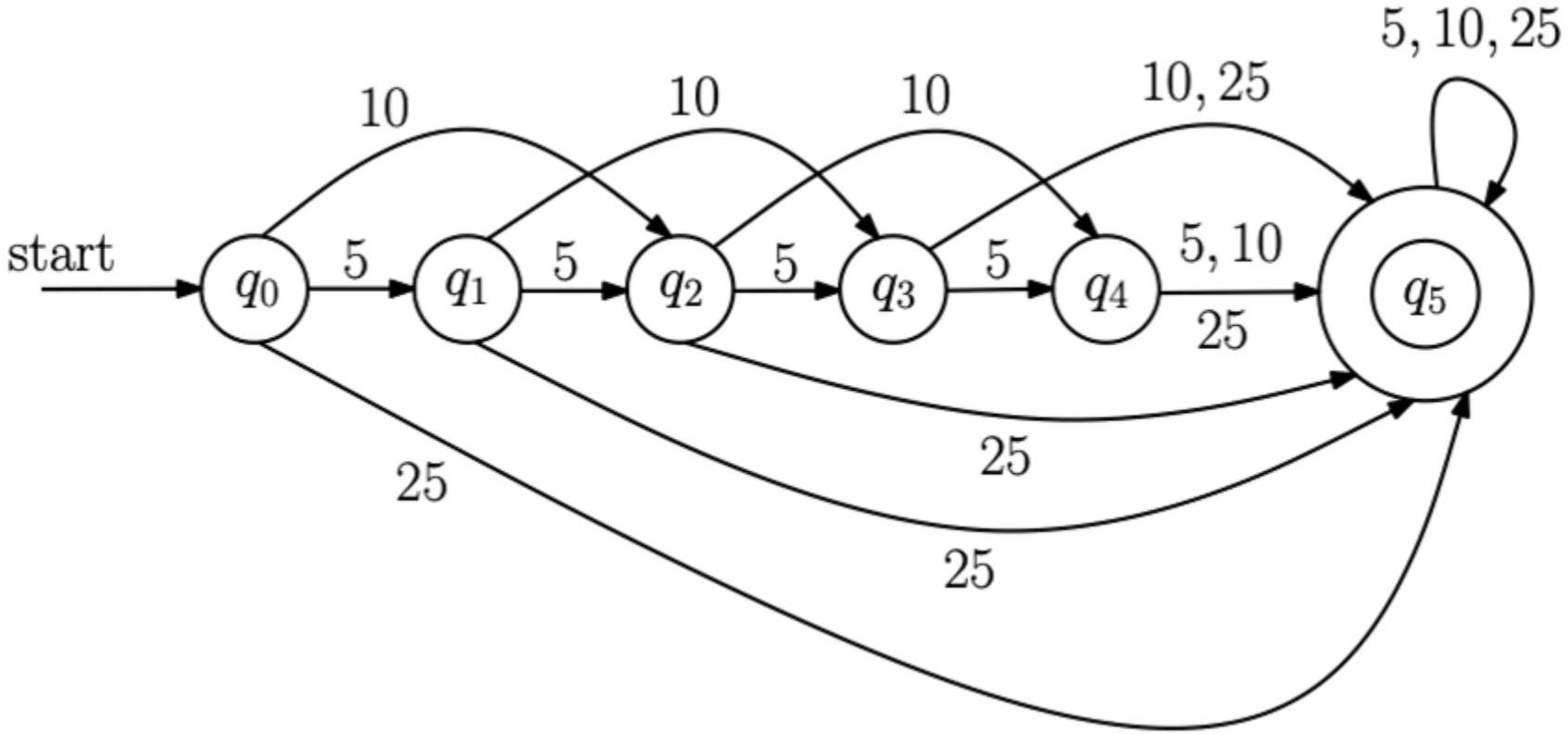
Start State

- One of the states defined to solve the problem must be identified as **Start** (or initial) state.



Accept State

- The states which should open the gate must be described as **Accept** (or final) states.



Formal Definition

We now come to the formal definition of a finite automaton.

Definition: A finite automaton is a 5-tuple $M = (Q, \Sigma, \delta, q, F)$ where

1. Q is a finite set, whose elements are called states,
2. Σ is a finite set, called the alphabet; the elements of Σ are called symbols,
3. $\delta : Q \times \Sigma \rightarrow Q$, called the transition function,
4. q is an element of Q; it is called the start state,
5. F is a subset of Q; the elements of F are called accept states.

It is your turn

- Let there be a system that works with 1€ and 2€ coins. Let's represent these two coins with a (1€) and b (2€). Draw the transition diagram of the finite state automaton that does not open the door without giving at least two €2.

A first example

A = {w : w is a binary string containing an **odd** number of 1s}.

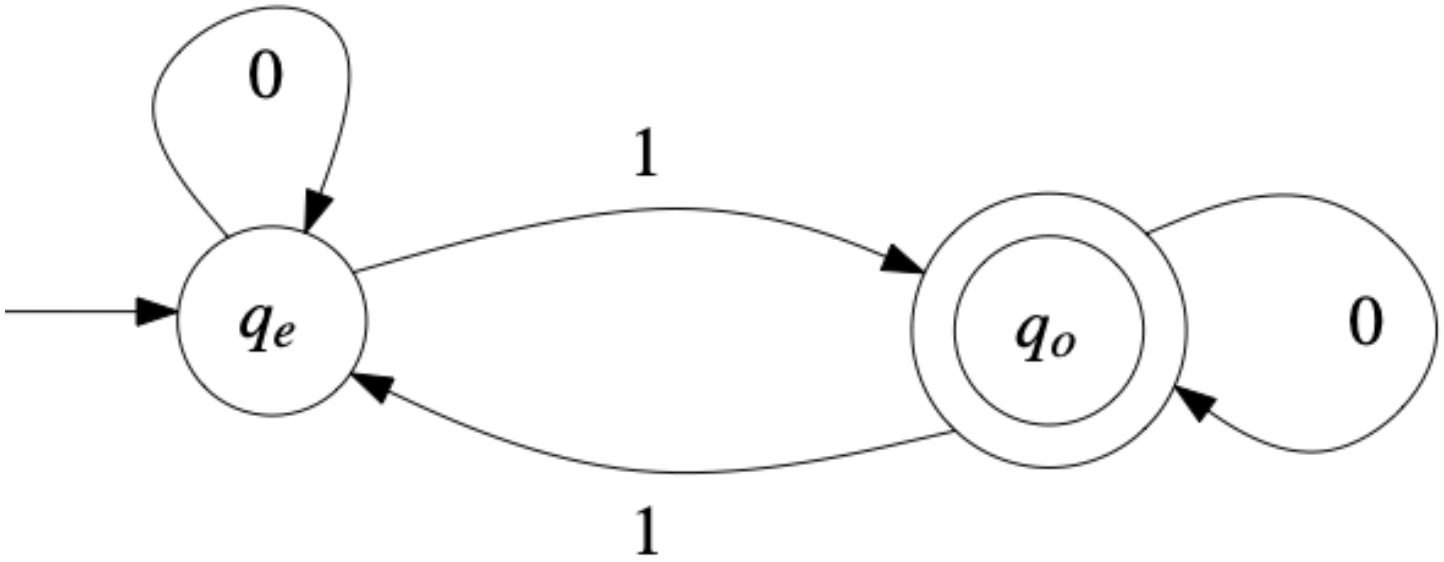
$\Sigma = \{0, 1\}$

0000 **1** 0000000 1 000000 **1** 0000000 1 000000 **1** 00000

odd even **odd** even **odd**

A first example

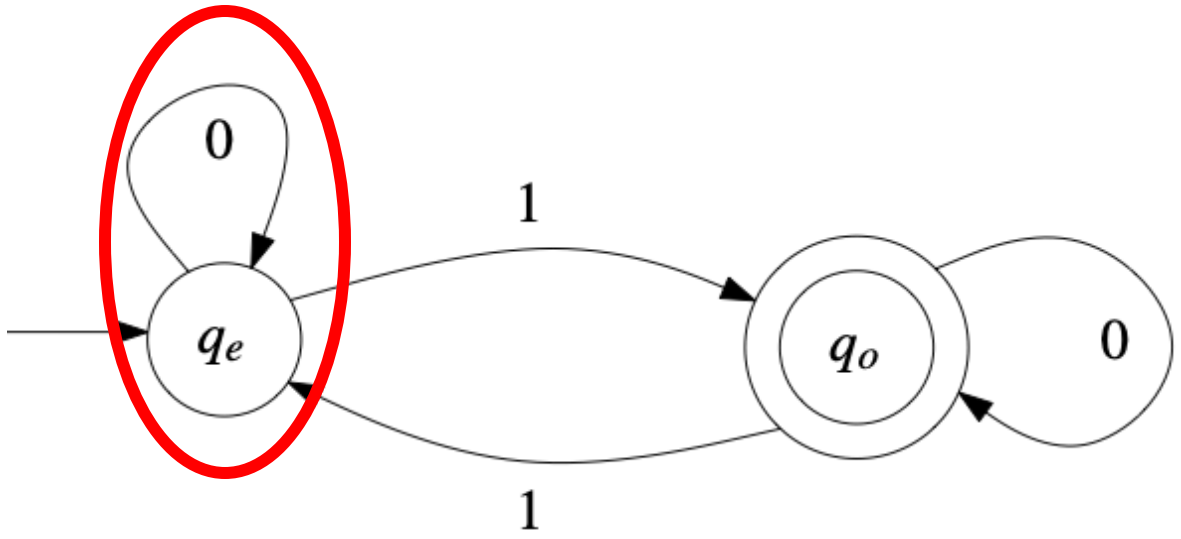
- q_e : even number of 1s
- q_o : odd number of 1s



A first example

Table Method

	0	1
q_e	q_e	q_o
q_o	q_o	q_e



A second example

A = {w : w is a binary string containing 101 as a substring}.

Here, when we reach a “101” sequence, we accept it. This substring can be at beginning, end or middle of the string.

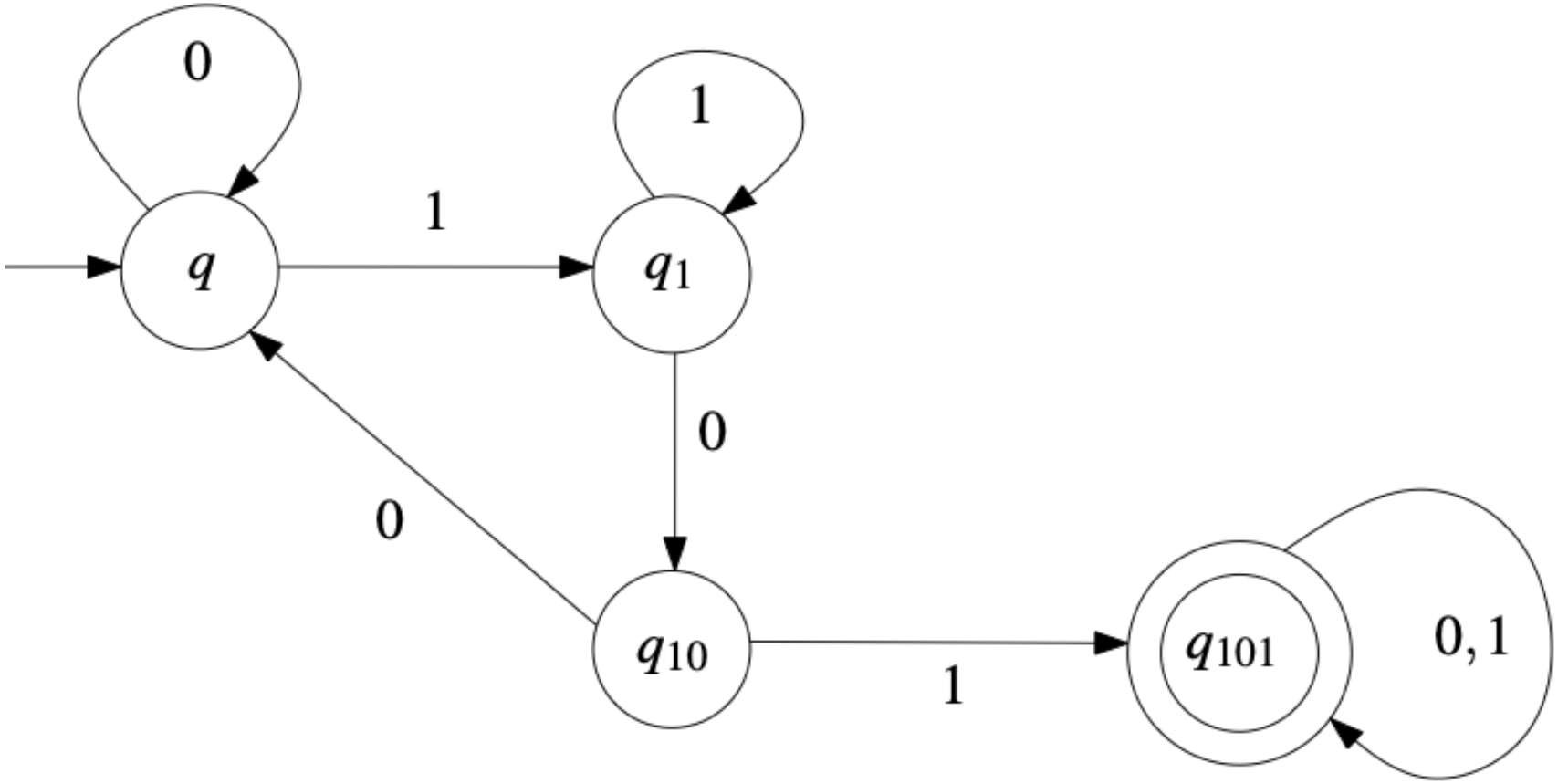
q : Start state, we have nothing

q_1 : The first symbol (“1”) made an alarm

q_{10} : The second symbol (“0”) increase the alarm level

q_{101} : The third symbol (“1”) accepts the string

A second example



A third example

A = {w ∈ {0,1}* : w has a 1 in the third position from the **LEFT**}

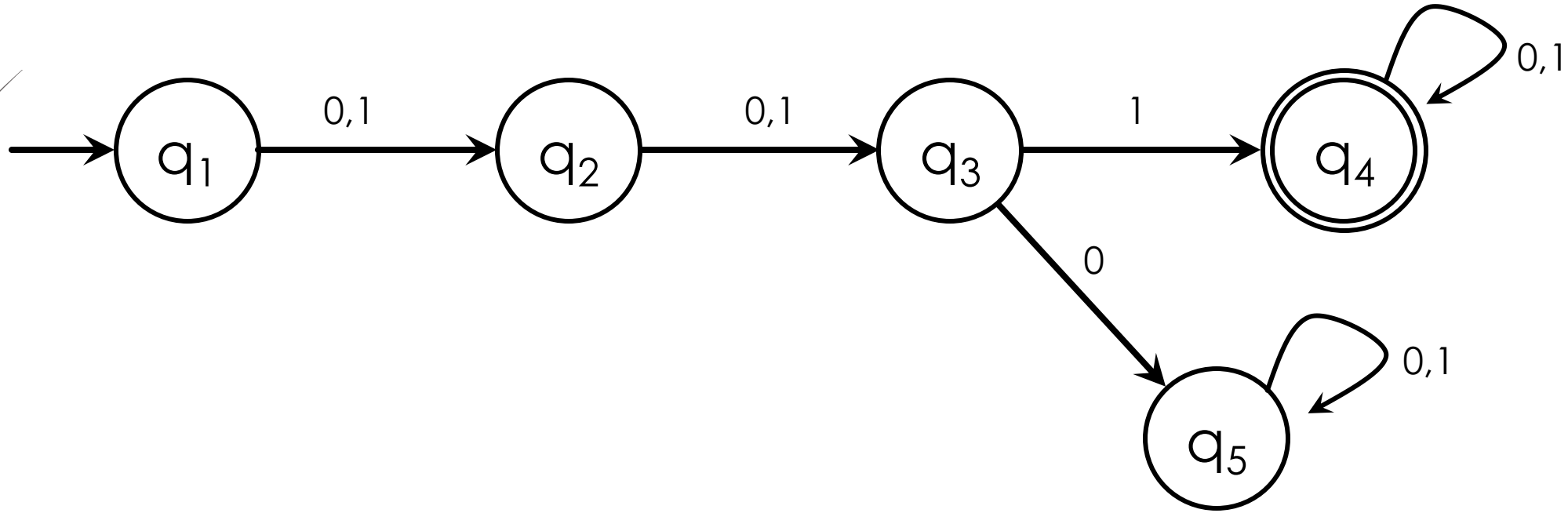
- This problem is easier than the previous one. Because the first and the second letters are not important, we can think at first the third letter from the start.

{0,1} {0,1} **1** {0,1}*

- If the third one is 1, then go an accept stay and stay there, else, go a reject state and stay there.

A third example

Here, we should think the solution directly. Thus, we draw the first four states. Then we add a last state for rejecting. So, using five states is enough.



It is your turn

$A = \{w \in \{0,1\}^* : w \text{ contains an even number of 0s and 1s}\}$

Here, we can assume that there are even numbers of 0s and 1s in the start state. Now we need four states as:

- q_{ee}: even number of 0s and even numbers of 1s
- q_{oe}: odd number of 0s and even numbers of 1s
- q_{eo}: even number of 0s and odd numbers of 1s
- q_{oo}: odd number of 0s and odd numbers of 1s

A fourth example

A = {w ∈ {0,1}* : w has a 1 in the third position from the right}

Here, the last three letters are important, and we need remember them all. Therefore, we need 8 different states:

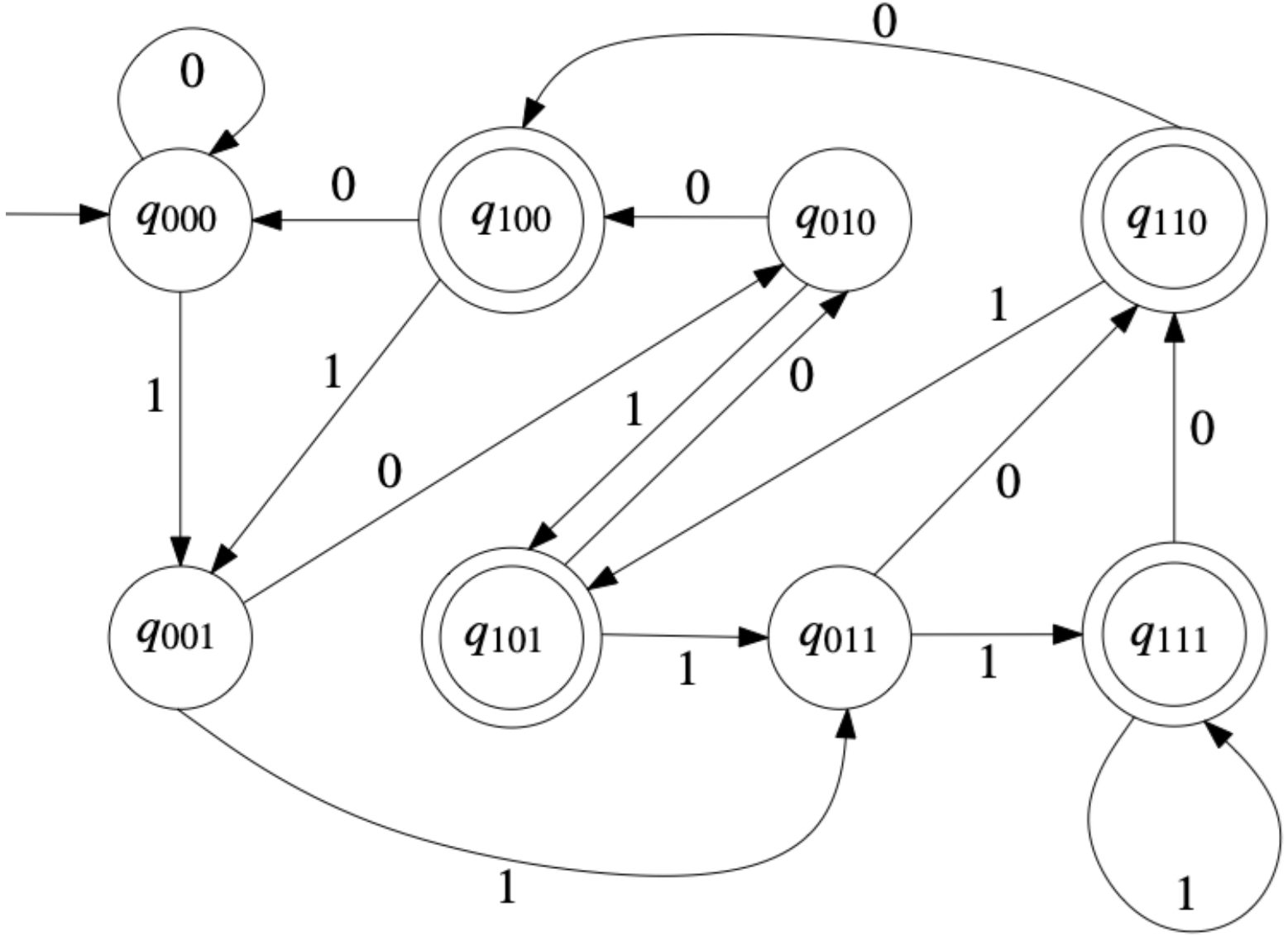
q_{000}	q_{100}
q_{001}	q_{101}
q_{010}	q_{110}
q_{011}	q_{111}

- Since each new symbol from the user is added to the right of the string, the state in the design may have to change

A fourth example

After determination of the states, designing the transitions will be easier.

☺

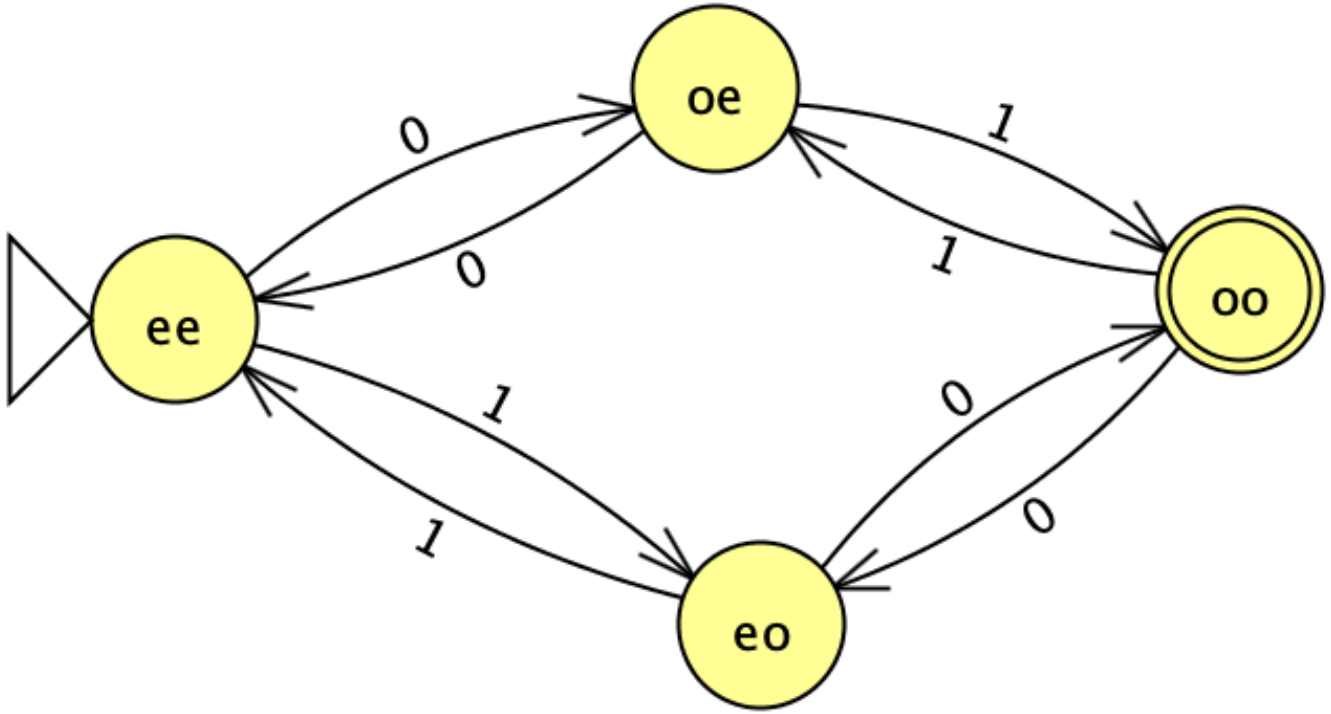


It is your turn

- Draw the transition diagram of the finite state automaton of the regular language where both 0's and 1's are odd.

A={01, 10, 0001, 0111, 1110, 1000, 111000,…}

Solution



Input	Result
00001111	Reject
010101	Accept
101010	Accept
11001100	Reject
0110011001	Accept

- That’s all.
- Thanks for listening.