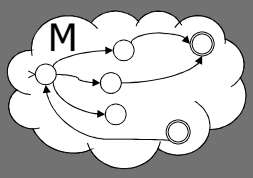
Date:12-12-2021

18CS54
Automata Theory & Computability

FSM used in Programming Language



Dr. Gopalakrishna M T,
Professor, CSE
✉ gopalakrishnamt@sjbit.edu.in

ll Jai Sri Gurudev ll

S J B INSTITUTE OF TECHNOLOGY

Implication of the View

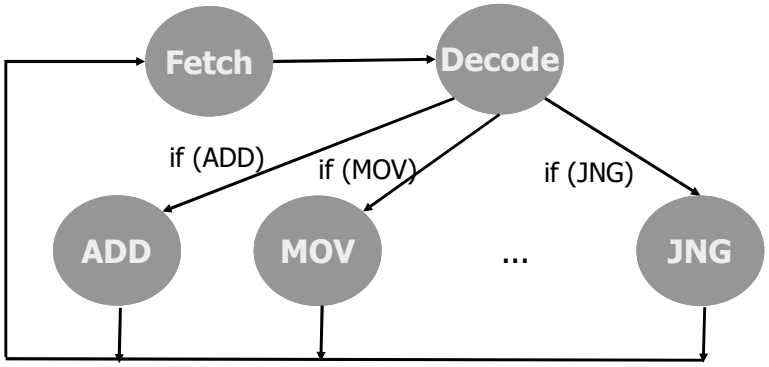
- We can do anything as long as we do not disturb the "state"

```
z = x * 3.1412;
if (z > 10)
  j = j + k;
```

- Since FP multiplication takes a long time, can we execute the addition at the same time?
  → out-of-order, speculative

FSM of the Computer

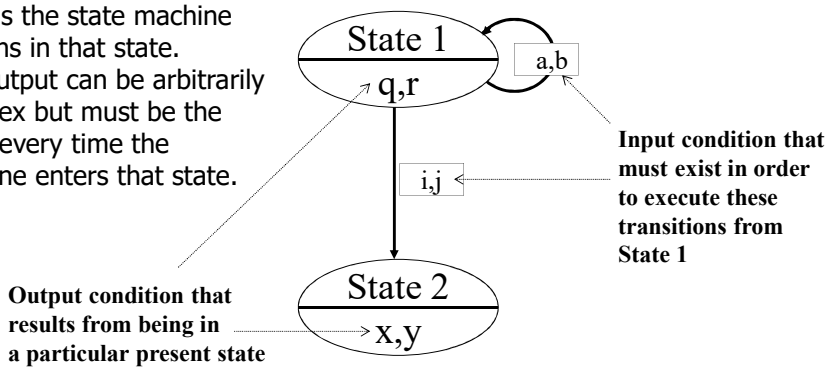
- For this highly simplified computer, the controller can be described by a FSM



Each state will generate certain control signals to control the datapath

Moore Machine

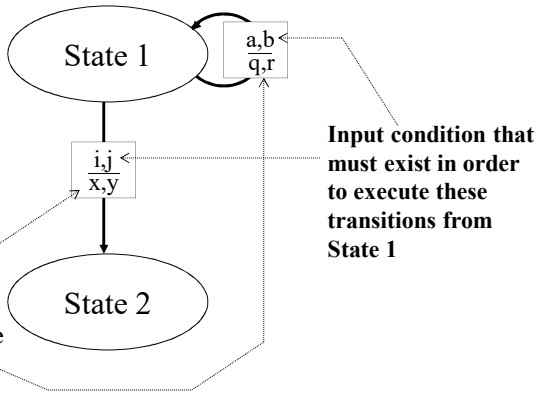
The Moore State Machine output remains the same as long as the state machine remains in that state.
The output can be arbitrarily complex but must be the same every time the machine enters that state.



Mealy Machine

The Mealy State Machine generates outputs based on:
- The Present State, and
- The Inputs to the M/c.

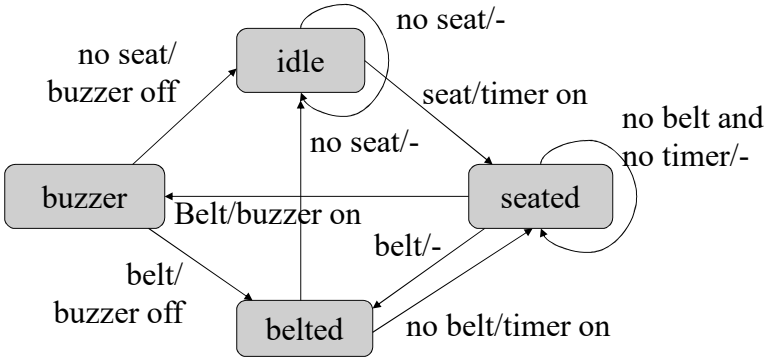
So, same state can generate many different patterns of output signals, depending on the inputs.
Outputs are shown on transitions since they are determined in the same way as is the next state.



Safety Belt Control

- We want to design a controller for safety belt
  - If the seat is seated and the belt is not buckled within a set time, a buzzer will sound until the belt is buckled → event driven
  - Inputs: seat sensor, timer, belt sensor
  - Output: buzzer, timer
  - System: specialized computer for reacting according to events sensed by the sensors

FSM for Event-driven Systems



C Code Structure

- Current state is kept in a variable
- State table is implemented as a switch
  - Cases define states
  - States can test inputs and produce outputs

```
while (TRUE) {
   switch (state) {
        case state1: …
   }
}
```

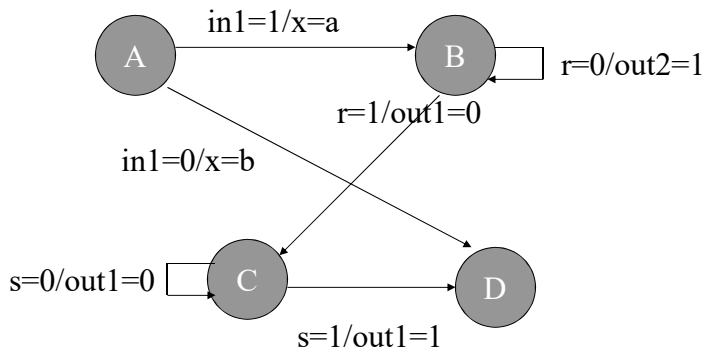
- Switch is repeatedly evaluated by while-loop

C Implementation

```
#define IDLE 0
#define SEATED 1
#define BELTED 2
#define BUZZER 3
switch (state) {
  case IDLE: if (seat)
     { state = SEATED; timer_on = TRUE; }
     break;
  case SEATED: if (belt) state = BELTED;
          else if (timer) state = BUZZER;
     break;
  …
}
```

8

Another Example



9

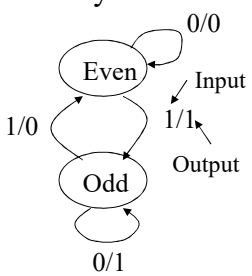
C State Table

```
switch (state) {
case A: if (in1==1) { x = a; state = B; }
  else { x = b; state = D; }
  break;
case B: if (r==0) { out2 = 1; state = B;
  }
  else { out1 = 0; state = C; }
  break;
case C: if (s==0) { out1 = 0; state = C;
  }
  else { out1 = 1; state = D; }
  break;
```

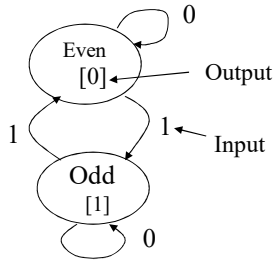

FSM as Recognizer/Translator

- Outputs a '0' if an even # of 1's is received and outputs a '1' otherwise
- What is state?
  - Two states: whether an even # of 1s have been received, or an odd # of 1s have been received

Mealy Machine

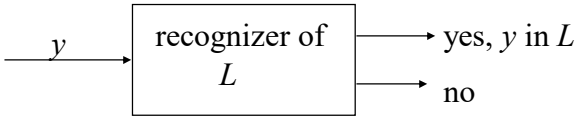


Moore Machine

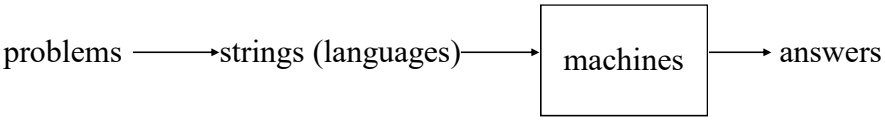


Finite State Machines

- Language recognizer (acceptor)



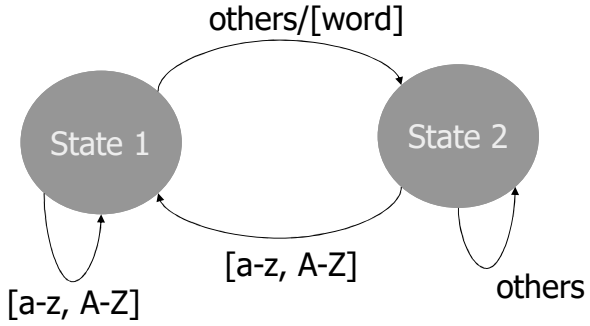
- Problem solver



12

FSM as Recognizer/Translator

- How to recognize words in a document?
  - Assume characters in the document are input sequentially
  - What is state?



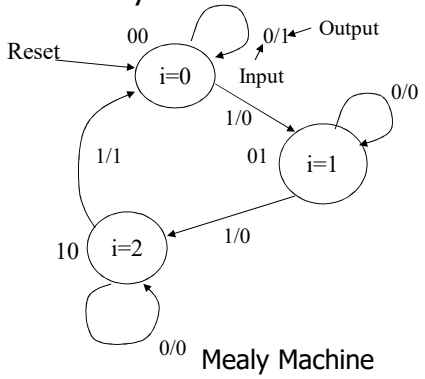
13

Example

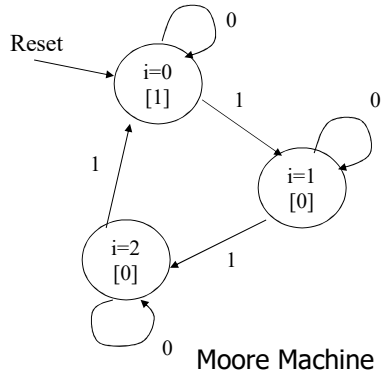
- A FSM that outputs a '1' whenever it receives a multiple of 3 # of 1's on a serial input
- Relevant information to solve the problem:
  (A) A multiple of 3 # is received
  (B) A non-multiple of 3 # is received
- Questions to consider:
  (1) How do we go from (A)→(B)
  Ans.: If a '0' is received
  (2) How do we go from (B)→(A)
  Ans.: Not clear; need to consider
  (B1): 3y+1 # of 1's received.
  (B2): 3y+2 # of 1's received.
  } y is an integer ≥ 0

Example

- Transitions between 3 classes of information:
  (A) $\xrightarrow[\text{1 received}]{}$ (B1) $\xrightarrow[\text{1 received}]{}$ (B2) $\xrightarrow[\text{1 received}]{}$ (A)
  - They can be considered states of the FSM:



Mealy Machine



Moore Machine