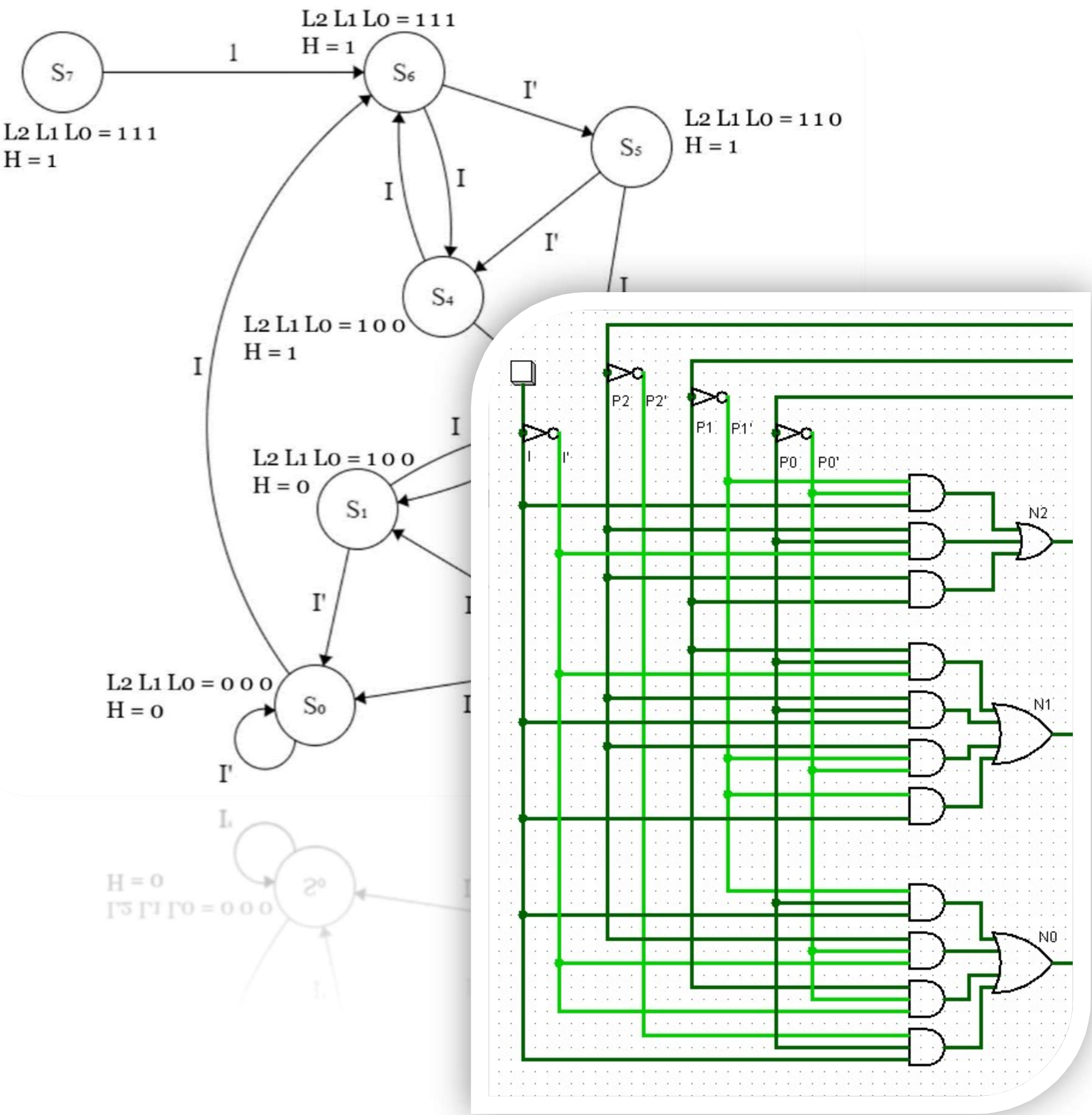
CT101 - Assignment 4
Finite State Machines

Maxwell Maia

Student ID: 21236277



February 2022

Definition of a finite state machine

A finite state machine has multiple states of which it can only exist in one at a time. It has a starting states where it begins its operation. Certain inputs will lead to certain changes in state. These state changes are called “transitions,” they occur when the finite state machine goes from one state to another. State changes are mapped by a transition function which maps inputs and current states to a “next state.” Certain outputs may be generated depending on the state transitions and the inputs or just the state transitions. (Black, 2021).

History

Edward Forrest Moore was an American professor of mathematics and computer science. Born: 1925 and died in 2003. He invented the first finite state machine which was called the “Moore machine.” (Wikipedia Contributors, 2019).

Illustrated example

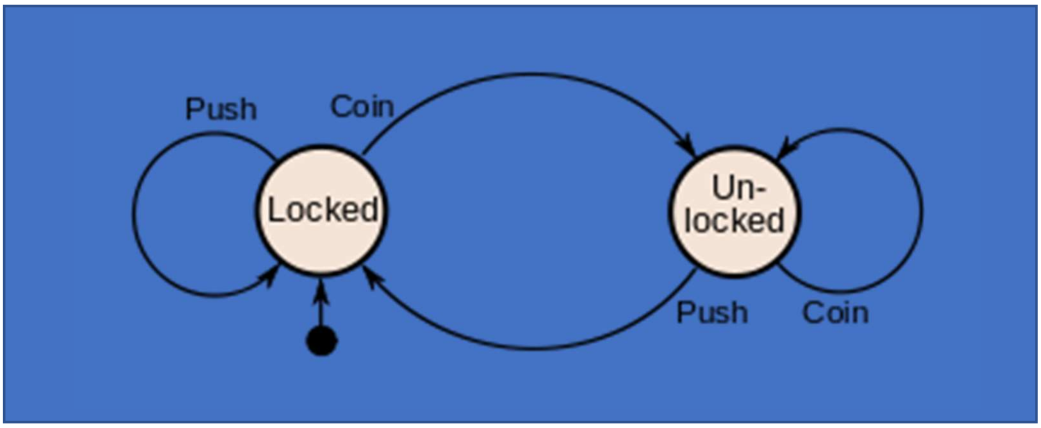


Figure 1: Turnstile finite state machine example. (Source: Brilliant:online)

The above figure shows a basic diagram of a finite state machine. This example describes the operation of a coin operated turnstile.

Figure 2 shows a turnstile. The following description is how we expect a turnstile to function. Initially the turnstile is locked. When a coin is inserted, the turnstile is unlocked. Once the turnstile is pushed, to let someone through, it will lock. The turnstile remains locked if it is pushed while locked. The turnstile remains unlocked if a coin is inserted while it is unlocked. (Brilliant.org, n.d.).

Figure 2: Picture of a coin-operated turnstile. (Source: eds:online)

There are 2 states in this finite state machine: “locked” and “unlocked”. States are depicted with an oval. However, in this basic diagram, states are depicted with circles.

There are 2 inputs: “the turnstile is pushed” and “a coin is inserted.” Inputs are labelled on transitions. Transitions are depicted by arrows leading from state to state.

A certain transition between states occurs depending on a combination of the current state information and the current input information.

The turnstile begins operation in the “locked” state. If it is pushed, it will remain in the locked state. If a coin is inserted. It will transition to the “unlocked” state. In the “unlocked” state, if a coin is inserted, it will remain in the “unlocked” state. If the turnstile is pushed to let a person through, it will transition to the “locked state.”

The diagram completely describes the intended operation of the turnstile. It should include all of the possible states the system can be in.

Example uses for a finite state machine

Pen

A pen has 2 states. The point is retracted (S0) / the point is extended (S1). It has 1 input - the button.

Initially the pen is in state S0. When the button is pressed the state changes to S1. In S1, if the button is pressed, the state changes to S0 and the point retracts.

The output is the point being retracted or extended.

Safe (digital combination lock)

This machine consists of a starting state, where the safe is locked. Then more "locked" states that represent each correct digit input. When a correct digit is input, the machine transitions to the next "locked" state. If an incorrect digit is input, the machine transitions to the starting state. Eventually, through the consecutive correct input of digits, the machine will transition from the last "locked" state to the "unlocked" state and the safe will open.

(Software Engineering Stack Exchange, 2011).

Pacman ghost AI

The ghost can be in one of the 3 states: chasing Pacman, dead or frightened. A finite state machine can be used to calculate the state that the ghost should be in.

When the game starts, the ghost is chasing Pacman:
If Pacman eats an energizer, transition to frightened.

In rightened state:
If frightened timer ends, transition to chase Pacman.
If Pacman hits the ghost, transition to dead.

In Dead state:
If ghost enters home, transition to chase Pacman.

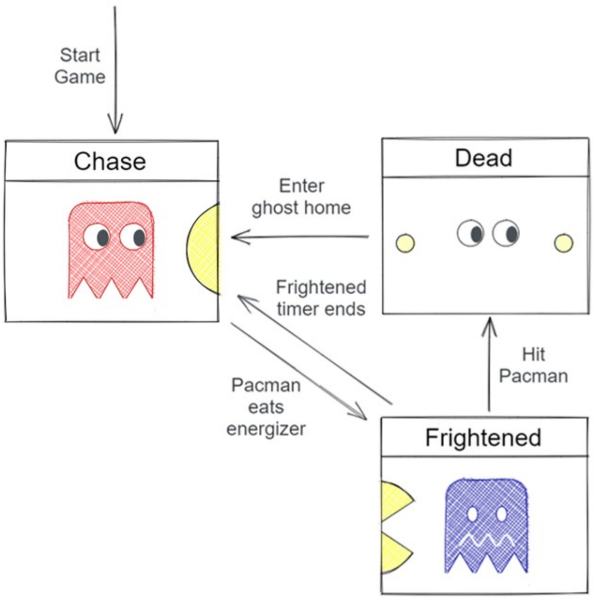


Figure 3: State diagram of a Pacman ghost.
(Scott Logic, 2020)

Mealy and Moore finite state machines

The only difference lies in how the output is generated.

In a Mealy state machine, the output of that machine is a function of the input and the present state. (Neso Academy, 2015b).

Output is determined by the present state and the input.

In a Moore state machine, the output of that machine is a function of the present state only. (Neso Academy, 2015a).

Output is determined by the state.

Mealy	**Moore**
Output dependant on the current state and input.	Output dependant on the current state only.
Generally uses less states to represent a function. So needs less hardware.	Uses more states to represent the same function. So needs more hardware.
More complex design.	Simpler design.
The output can change at any point in a clock cycle (faster), as the input changes. Reason: output logic is not dependant on the clock. Output logic is dependent on state and input. Therefore, if the input changes, the output may change (if the designer intended it to).	The output will only change on an active clock edge (slower). Reason: a particular state is linked to a particular output. The state will only change on an active clock edge. Therefore, the output will only change on an active clock edge.

Output is calculated in the same logic block as the next state logic.	Output logic is calculated separately from next state logic. (After the next state logic and register).
Asynchronous output. The state will change synchronously to the clock.	The output and next state change synchronously to the clock's active edge.
Next state logic is the same.	Next state logic is the same.

(Sidhartha, 2016), (Singhal, 2022).

Full FSM Example

Specification – Killer shack game

My FSM is a game where a survivor is running away from a killer. The killer is faster than the survivor and will catch up to the survivor over time. Luckily, the survivor is running past many large obstacles that they could throw down behind them at any time to keep the killer at a distance. If they drop an obstacle too early though, the killer avoids the obstacle and gets much closer to the survivor.

A clock reaches its rising edge every 6 seconds. At this point the state will change depending on the player's (the survivor) input.

Inputs

The only input is whether the survivor drops an obstacle or not. Represented by 1 (obstacle) or 0 (no obstacle). This input is labelled as I.

States

The survivor can be healthy, injured or dead. There is a distance between the killer and survivor: 3 metres, 2 metres or 1 meter. This information is represented by the states of the finite state machine.

Outputs

4 bits in total.

The first 3 bits (L_2 L_1 L_0) represent the distance between the killer and the survivor. Each bit individually controls an LED. This will give the player a visual display of the distance between the killer and survivor. 2 bits could be used for this but 3 bits were intentionally used to simplify the output logic.

1 light = 1 meter

2 lights = 2 metres

3 lights = 3 metres.

The last bit (H) controls a light that represents whether the survivor is "healthy" or not.

Summary of states

States in binary	State name	Description
000	S0	Dead
001	S1	1m distance to killer. Survivor injured.
010	S2	2m distance to killer. Survivor injured.
011	S3	3m distance to killer. Survivor injured.
100	S4	1m distance to killer. Survivor healthy.
101	S5	2m distance to killer. Survivor healthy.
110	S6	3m distance to killer. Survivor healthy.
111	S7	Dummy state

Expected behaviour

The killer is faster than the survivor.

So

- After every clock cycle the distance between the killer and survivor decreases by 1.

- Every clock cycle, the survivor will either drop an obstacle behind themselves or not. This is the input (0/1).

- If the survivor drops and obstacle behind themselves when the killer is 1 meter behind them, (the killer is then stunned and) the distance between the killer and survivor is set to 3 metres.

- If the survivor drops the obstacle when the distance is 3 or 2 metres, the distance is reduced by 2 metres.

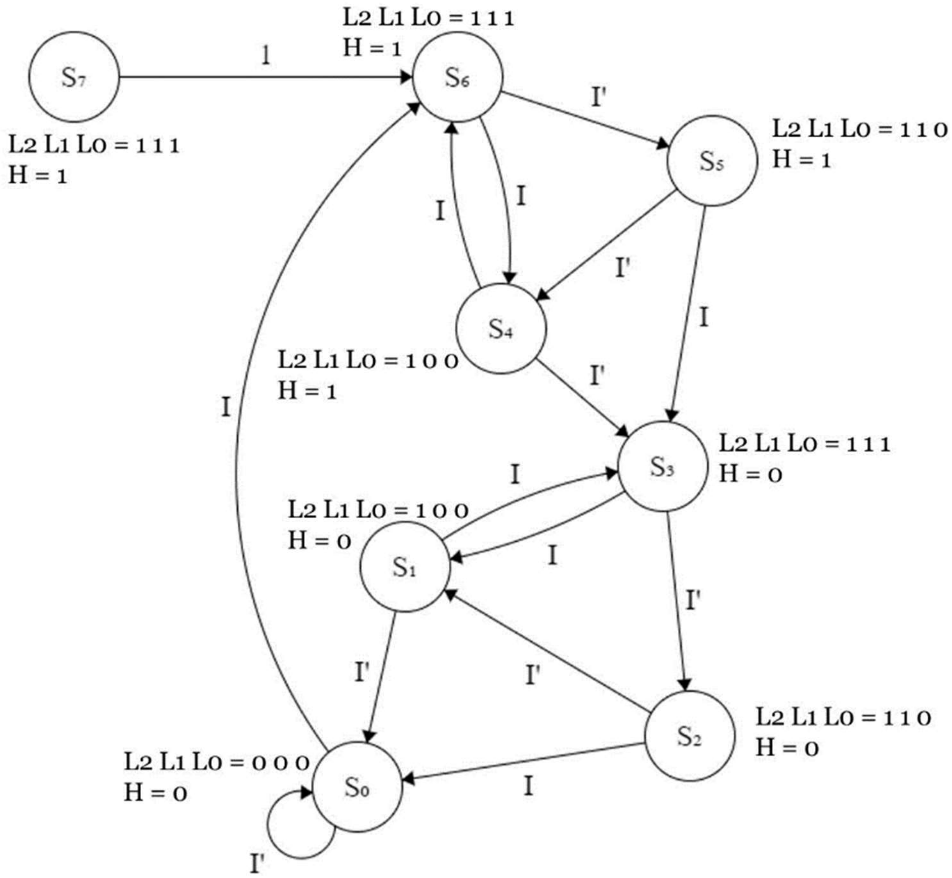
- If, at any point, the distance is less than 1 meter, the survivor (is hit and) goes from either healthy to injured or injured to dead.

- If the input is 1 when the survivor is dead, the game will restart.

- One dummy state will be used since there are 7 states being used but 8 possible combinations in a 3-bit number. The dummy state transitions to S6 on the next rising edge of the clock no matter what the input is. The dummy state is S7 = 111.

State table and diagram

Present state	Input		Next state	$L_2 L_1 L_0$	H
S7 (Dummy state)	X		S6	111	1
S6	0		S5	110	1
S6	1		S4	100	1
S5	0		S4	100	1
S5	1		S3	111	0
S4	0		S3	111	0
S4	1		S6	111	1
S3	0		S2	110	0
S3	1		S1	100	0
S2	0		S1	100	0
S2	1		S0	000	0
S1	0		S0	000	0
S1	1		S3	111	0
S0	0		S0	000	0
S0	1		S6	111	1



Next state logic

Present state P_2 P_1 P_0	Input I		Next state N_2 N_1 N_0
111	0		110
111	1		110
110	0		101
110	1		100
101	0		100
101	1		011
100	0		011
100	1		110
011	0		010
011	1		001
010	0		001
010	1		000
001	0		000
001	1		011
000	0		000
000	1		110

Karnaugh maps

<u>N2</u> = $P_1'P_0'I + P_2P_0I' + P_2P_1$

P2 P1 \ P0 I	00	01	11	01
00	0	1	0	0
01	0	0	0	0
11	1	1	1	1
10	0	1	0	1

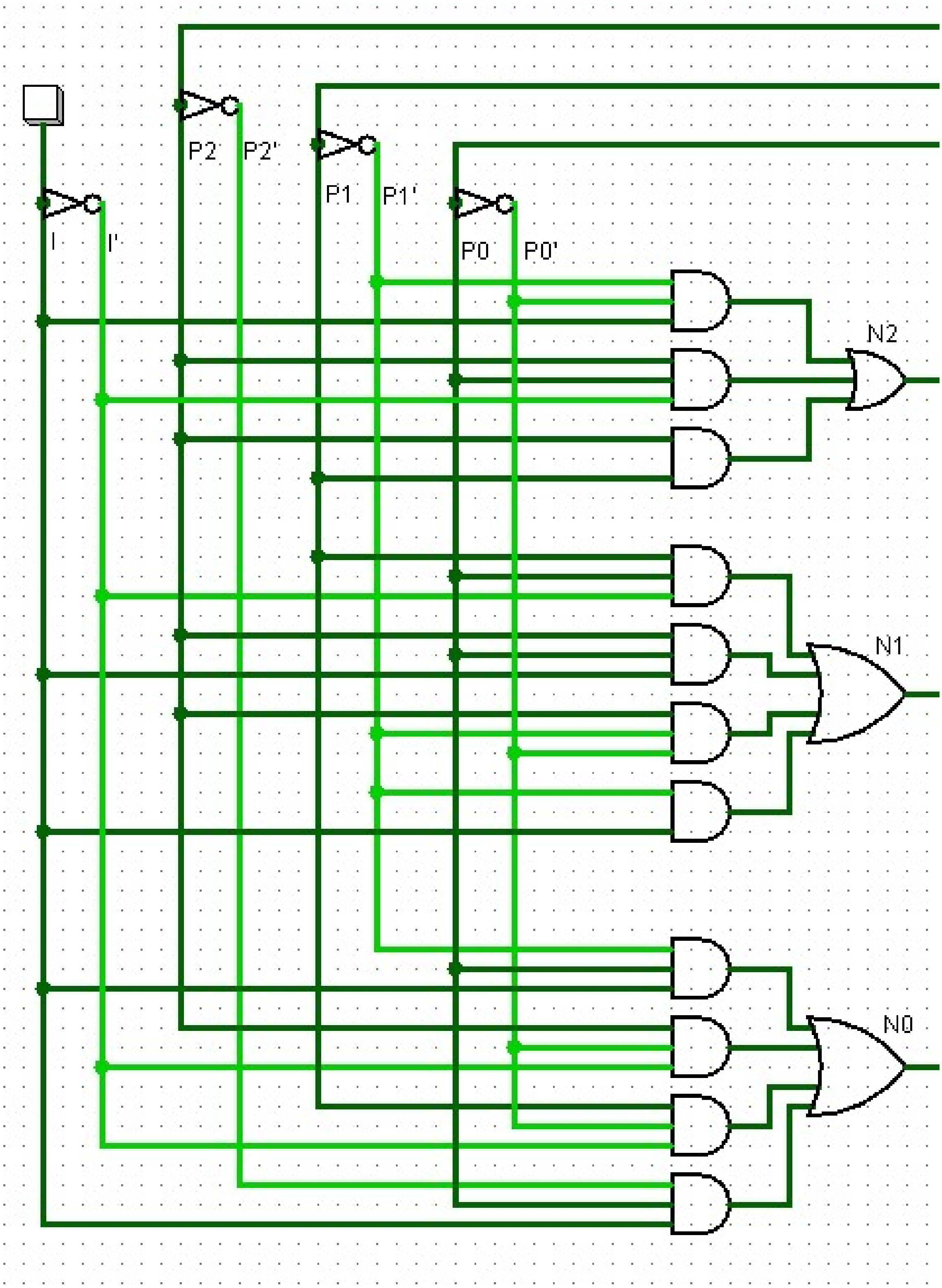
<u>N1</u> = $P_1P_0I' + P_2P_0I + P_2P_1'P_0' + P_1'I$

P2 P1 \ P0 I	00	01	11	01
00	0	1	1	0
01	0	0	0	1
11	0	0	1	1
10	1	1	1	0

<u>N0</u> = $P_1'P_0I + P_2P_0'I' + P_1P_0'I' + P_2'P_0I$

P2 P1 \ P0 I	00	01	11	01
00	0	0	1	0
01	1	0	1	0
11	1	0	0	0
10	1	0	1	0

Next state logic illustrated



Bibliography

Brilliant.org (n.d.). *Finite State Machines | Brilliant Math & Science Wiki*. [online] brilliant.org. Available at: https://brilliant.org/wiki/finite-state-machines/# [Accessed 27 Jan. 2022].

Neso Academy (2015a). Mealy and Moore State Machines (Part 1). [online] www.youtube.com. Available at: https://www.youtube.com/watch?v=0_OZKWdCixw&ab_channel=NesoAcademy [Accessed 2 Feb. 2022].

Neso Academy (2015b). Mealy and Moore State Machines (Part 2). [online] www.youtube.com. Available at: https://www.youtube.com/watch?v=vdZKehWMJSM&ab_channel=NesoAcademy [Accessed 2 Feb. 2022].

Paul E. Black, "finite state machine", in [*Dictionary of Algorithms and Data Structures*]() [online], Paul E. Black, ed. 4 October 2021. Available at: https://www.nist.gov/dads/HTML/finiteStateMachine.html
[Accessed 26/01/2022].

Sidhartha (2016). Mealy Vs. Moore Machine. [online] VLSIFacts. Available at: http://www.vlsifacts.com/mealy-vs-moore-machine/ [Accessed 13 Feb. 2022].

Singhal, U. (2022). Dare2Compete - Competitions, Quizzes, Hackathons, Scholarships and Internships for Students and Corporates. [online] dare2compete.com. Available at: https://dare2compete.com/blog/difference-between-mealy-and-moore-machine.
[Accessed 13/02/2022].

Software Engineering Stack Exchange. (2011). Examples of Finite State Machines. [online] Available at: https://softwareengineering.stackexchange.com/questions/47806/examples-of-finite-state-machines [Accessed 30 Jan. 2022].

Stobbs, M. (2020). Finite State Machines: The Developer’s Bug Spray. [online] Scott Logic. Available at: https://blog.scottlogic.com/2020/12/08/finite-state-machines.html [Accessed 13 Feb. 2022].

Wikipedia Contributors (2019). Finite-state machine. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Finite-state_machine [Accessed 2 Feb. 2022].

Images

Brilliant. *Turnstile finite state machine example.* [Online Image]
[Accessed on 27th January 2022]
https://brilliant.org/wiki/finite-state-machines/#

Scott Logic (2020). *Simple ghost fsm*. [Online Image] *Scott Logic*. Available at:
https://blog.scottlogic.com/2020/12/08/finite-state-machines.html [Accessed 13 Feb. 2022].

eds. *Picture of a coin operated turnstile*. [Online Image]
[Accessed on 27th January 2022]
https://www.edsuk.com/turnstiles/coin-operated-toilet-turnstiles/