AQA Computer Science AS Level

3.4.2 Finite state machines (FSMs)

Advanced Notes

Specification:

3.4.2.1 Finite state machines (FSMs) without output:

Be able to draw and interpret simple state transition diagrams and state transition tables for FSMs with no output.

Finite State Machines

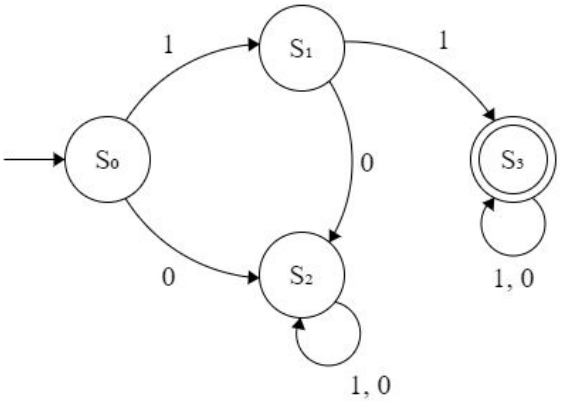
A finite state machine (or FSM for short) is a computational model for a machine that is always in a fixed state. Each finite state machine has a finite number of states and can only ever be in one state at a point in time.

The state of a finite state machine will change depending on the current state and the input data. If the input data is valid, the finite state machine will terminate in an accepting state.

A finite state machine's state can change and does so according to transition rules, rules that describe what a finite state machine should do given certain criteria.

State Transition Diagrams

State transition diagrams are used by computer scientists as a visual representation of a finite state machine. They consist of states (circles) joined by transitions (arrows) and always have a start state, indicated with a leading arrow. An accepting state is shown as a double circle.



For example, the state transition diagram above has four states: S_0, S_1, S_2 and S_3. The start state is S_0 and S_3 is an accepting state.

The transition functions are each represented by an arrow (the leading arrow merely signifies the start state and does not represent a transition function).

The finite state machine represented by the state transition diagram will only accept input data that starts with `11`. For example: `11`, `110`, `11101` and `11001100`.

State Transition Tables

The transition function between S_0 and S_1 in the previous diagram could be described in English as "*If the finite state machine is in state S_0 and the input is 1, move to state S_1.*"

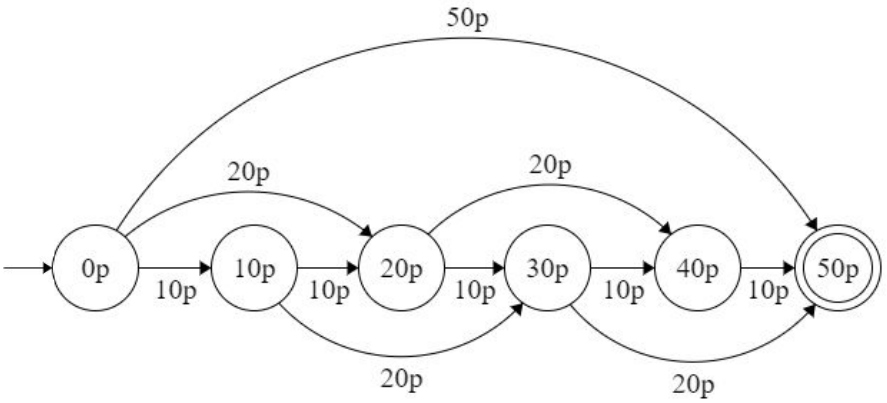
The transition functions in a finite state machine can be notated more formally using a state transition table, with columns for current state, input and next state, like the one below.

Current State	Input	Next State
S_0	1	S_1
S_0	0	S_2
S_1	1	S_3
S_1	0	S_2

Example - Parking Machine

The finite state machine shown by the state transition diagram below represents a parking machine which requires 50p to be payed. The machine is only designed to take coins worth 10p or more.

If the customer first pays 10p, the machine moves into the 10p state, from which the customer can input another 10p or 20p. The 50p state is the accepting state.



Example - Parity

This finite state machine represents odd parity. Only numbers with an odd total of 1s will be accepted.

The machine has two states, one start state and one accepting state.

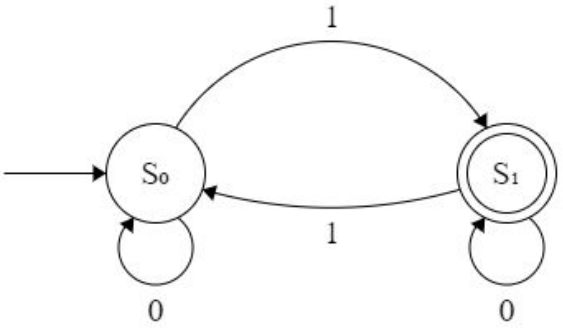
There are a total of four transition functions, which are shown in the state transition table on the left.

Synoptic Link

Even and odd **parity** can be used to **check for errors** in transmitted data.

Parity bits are explained in **Error checking and correction** under **Information coding systems.**

Current State	Input	Next State
S_0	1	S_1
S_0	0	S_0
S_1	1	S_0
S_1	0	S_1



Accepted input strings include `1`, `00000001`, `00100011`, `11111011` and `01010111`

Rejected input strings include `0`, `00000000`, `00011011`, `11111111` and `01011010`