WORKSHEET

A6.1 Finite state machines

Theory of computation · A level · AQA 7517 4.4.2.1 · about 20 min

Name _____ Class _____ Date _____

What this lesson is about

States, transitions and accepting states; state transition diagrams and tables; tracing an FSM and running one from a table.

- What an FSM is
- State transition diagrams
- State transition tables
- Tracing a run
- An FSM in Python
- Missing transitions and trap states
- Where FSMs are used

Questions

5 marks in all

1. In a state transition diagram for an FSM with no output, what does a double circle show? [1 mark]

- ☐ **A** The start state
- ☐ **B** An accepting state
- ☐ **C** A trap state
- ☐ **D** A state with a loop

2. The parity FSM starts in S0 (accepting). A 0 keeps the state and a 1 swaps S0 and S1. Which state is it in after reading 11010? [1 mark]

3. An FSM has 4 states and an input alphabet of 3 symbols. How many rows does its complete state transition table have? [1 mark]

4. An FSM has no transition shown for a 0 in state S0. What normally happens if it reads a 0 in S0? [1 mark]

- ☐ **A** It stays in S0
- ☐ **B** It moves to the start state
- ☐ **C** The input is rejected
- ☐ **D** The input is accepted

5. What does this program print? [1 mark]

```
table = {("S0", "a"): "S1", ("S0", "b"): "S0", ("S1", "a"): "S1", ("S1", "b"): "S0"}
state = "S0"
for symbol in "abba":
    state = table[(state, symbol)]
    print(state, end=" ")
print(state == "S1")
```

The task: ends in 01

Design a finite state machine that accepts binary strings that **end in `01`**, store it as a transition table, and run it. - `table` is a dictionary: each key is a pair `(state, symbol)`, where `symbol` is the string `"0"` or `"1"`, and each value is the next state. Include a transition for every state with both symbols. - The machine must decide by following transitions one symbol at a time. Do not look at the end of the string directly. - For each string in `tests`, in order, print one line: the string, a space, then `accept` or `reject`. For example `1101 accept`. Six lines in all. Work out what each state needs to remember before you write the table: how much of `01` has the machine just seen?

```
# the two lines every program starts with: the commands, then the robot
from bugbot import *
connect()

tests = ["01", "1101", "0110", "1", "00101", "0100"]

table = {}
```

Plan your program here, then type it in and press Run.

Do it on the robot

www.bugbotlab.com/learn/a6-1-finite-state-machines/

The simulator checks it and tells you when it passes. Nothing to install, no account.

Challenges

1. Draw the state transition diagram for your machine. Which state is accepting?
2. Change your table so it accepts strings that **contain** `00` anywhere. Does it need a trap state, or a state it can never leave?
3. What does the parity machine do with the empty string? Why is that the right answer?