WORKSHEET

A6.2 Mealy machines: FSMs with output

Theory of computation · A level · AQA 7517 4.4.2.1 · about 25 min

Name _____ Class _____ Date _____

What this lesson is about

Output on every transition, tracing a Mealy machine, and a search, approach and stop controller for the robot.

- Labelling transitions with output
- Tracing a Mealy machine
- Mealy and Moore
- A robot controller as a Mealy machine

Questions

5 marks in all

1. In a Mealy machine, what does the output depend on? [1 mark]

- ☐ **A** The current state only
- ☐ **B** The current state and the input symbol
- ☐ **C** The input symbol only
- ☐ **D** Whether the state is accepting

2. The edge detector outputs 1 only on the transition LOW to HIGH (input 1 in state LOW), and 0 otherwise. It starts in LOW. What does it output for the input 1101? [1 mark]

3. How long is a Mealy machine's output string compared with its input string? [1 mark]

- ☐ **A** Always one symbol
- ☐ **B** The same length
- ☐ **C** One symbol longer
- ☐ **D** It depends on the accepting states

4. Which of these are true of a Mealy machine? [1 mark]

Tick every answer that is true.

- ☐ **A** Each transition is labelled input/output
- ☐ **B** It has accepting states that decide yes or no
- ☐ **C** Its state transition table has an output column
- ☐ **D** It has a finite number of states

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

```
table = {("A", "0"): ("A", "x"), ("A", "1"): ("B", "y"), ("B", "0"): ("A", "z"), ("B", "1"):
("B", "x")}
state = "A"
output = ""
for symbol in "01101":
    state, o = table[(state, symbol)]
    output = output + o
print(output, state)
```

The task: search, approach, stop

Marker 4 is somewhere on the mat, out of view. Run the controller above as a table-driven Mealy machine so the robot finds it, drives up and stops. - `sense()` is written for you. It returns a pair: the input symbol (`"none"`, `"far"` or `"near"`) and a steering value in degrees that `act` uses. - `act(output, steer)` is written for you. It carries out one output symbol (`"spin"`, `"drive"` or `"halt"`). - Write `table` as a dictionary: each key is `(state, input)` and each value is `(next_state, output)`, with all six rows from the table above. - Start in `"SEARCH"`. Each tick: sense, look up the transition, act on the output, move to the next state, then `wait(0.1)`. Stop looping when the state is `"STOP"`. - Whenever the state changes, print the old and new state in the form `SEARCH -> APPROACH`. The robot must stop within 20 cm of the marker.

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

def bearing_of(cx):
    return (cx - 160) * 120 / 320      # pixels to degrees

def sense():
    tags = [t for t in apriltags() if t[0] == 4]
    if not tags:
        return "none", 0
    if tags[0][3] <= 15:
        return "near", 0
    return "far", bearing_of(tags[0][1])

def act(output, steer):
    if output == "spin":
        turn_right(40)
    elif output == "drive":
        drive(60, 0, steer * 3)
    elif output == "halt":
        stop()

set_cv("apriltag")
table = {}
state = "SEARCH"
```

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

Do it on the robot

www.bugbotlab.com/learn/a6-2-mealy-machines/

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

Challenges

1. Add an input symbol `bumped` and a state `BACK_OFF` that reverses for a moment. Write the new rows of the table first.
2. Draw a Mealy machine that outputs 1 on a **falling** edge, when the bumper is released.
3. Rewrite the edge detector as a Moore machine. How many states does it need?