State Machines for Robotics

CSCI 420-02 Robotics

WILLIAM & MARY

CHARTERED 1693

State Machines

- Represent the physical or logical *state* of the world or robot
- Events or actions trigger transitions between states

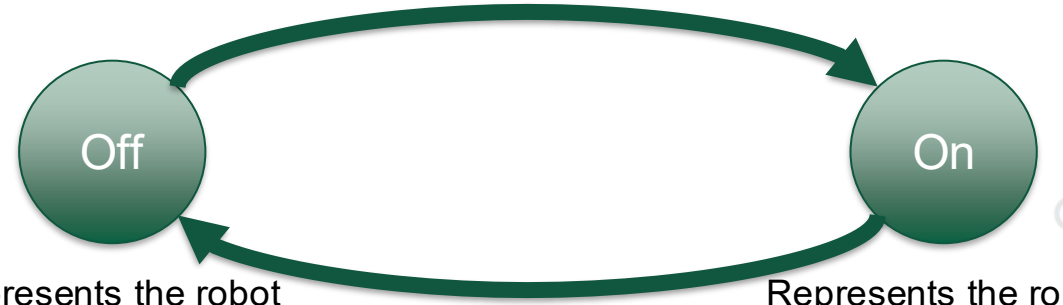
Extra reading on Moore Machines and Mealy Machines.

Represented with Graphs

Represents the robot being unpowered

Represents the robot being powered

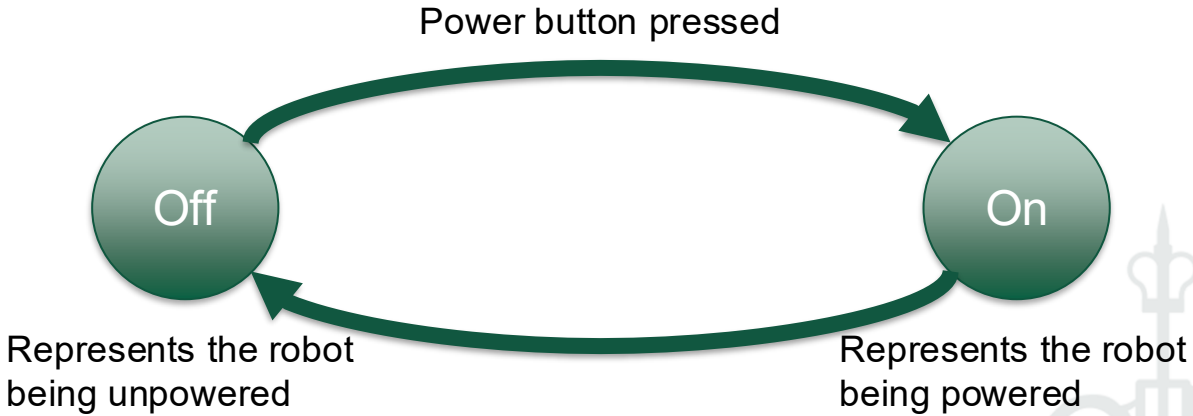
Represented with Graphs



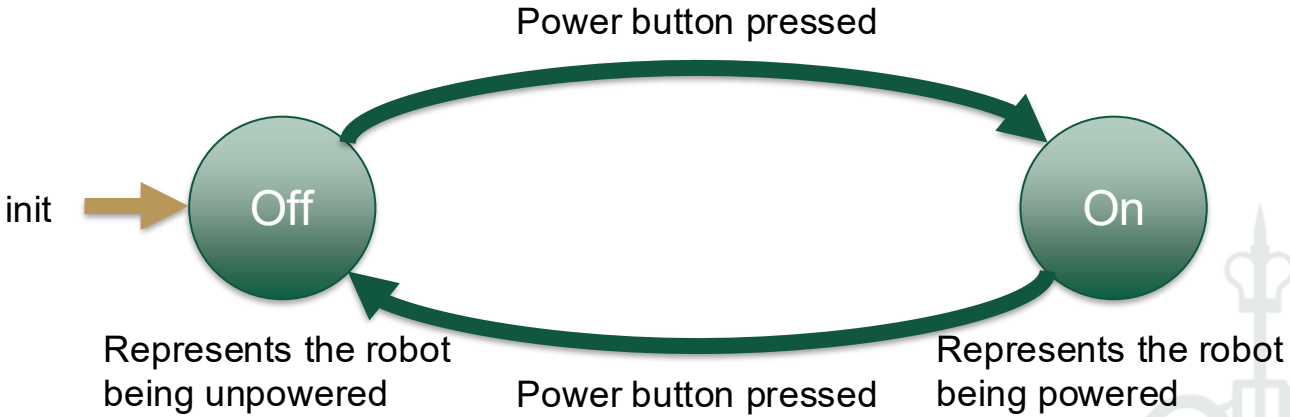
Represents the robot being unpowered

Represents the robot being powered

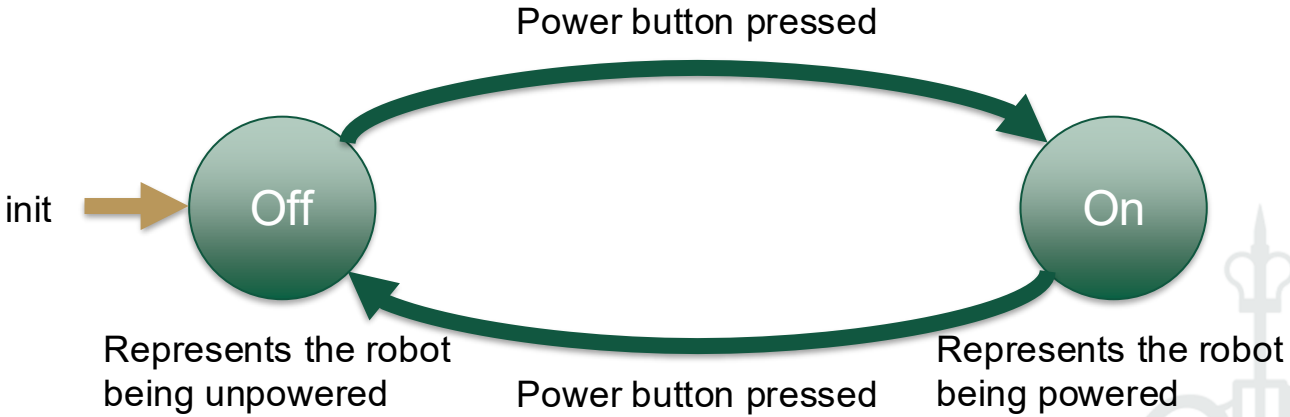
Represented with Graphs



Represented with Graphs

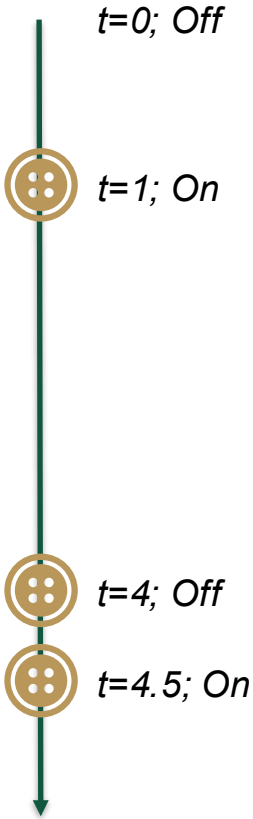


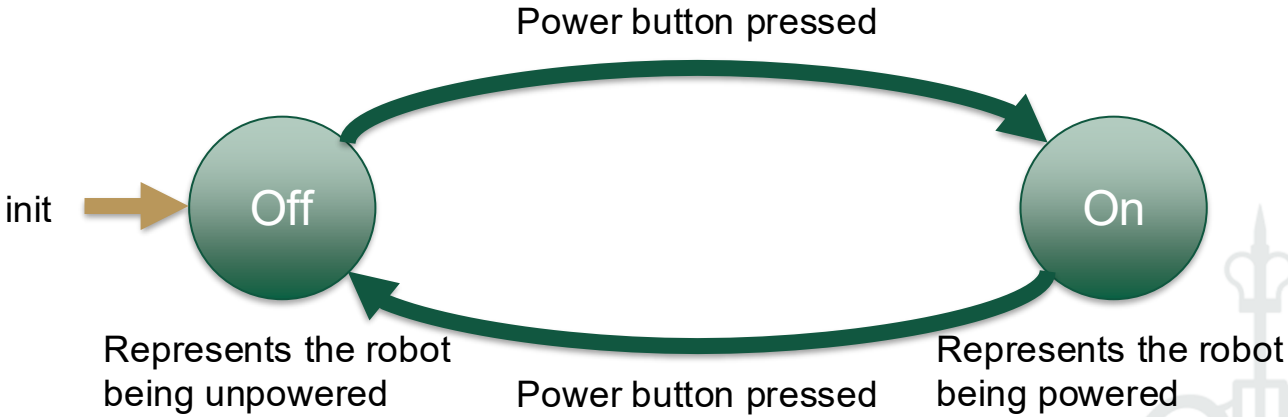
Represented with Graphs



This state machine is ***event based***.
If nothing happens, no state changes.

Represented with Graphs





This state machine is ***event based***.
If nothing happens, no state changes.

Timed State Machines

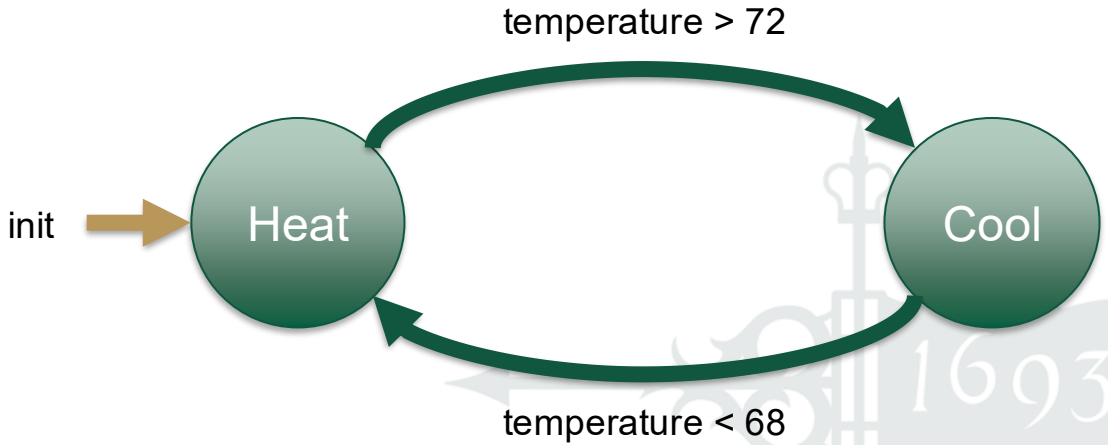
- Don’t wait for events – check at interval

Timed State Machines

- Don’t wait for events – check at interval

Timed State Machines

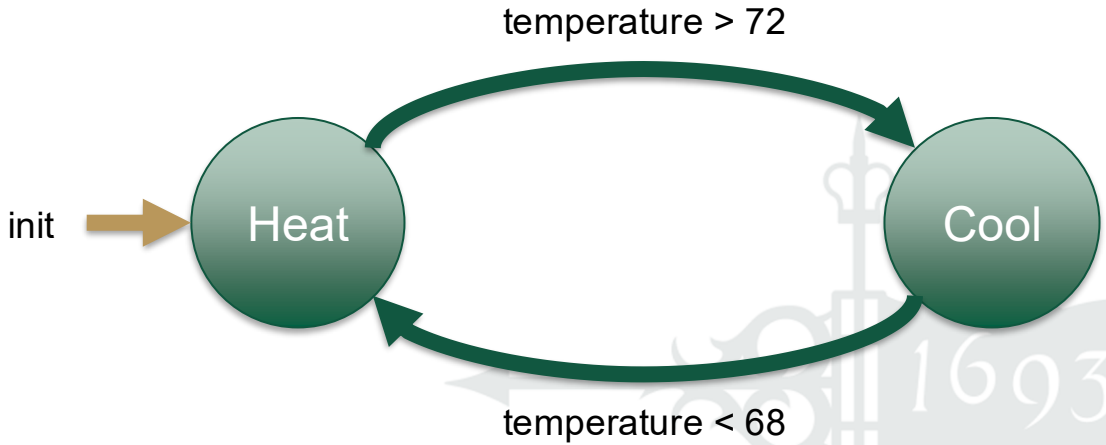
- Don’t wait for events – check at interval



Timed State Machines

- Don’t wait for events – check at interval

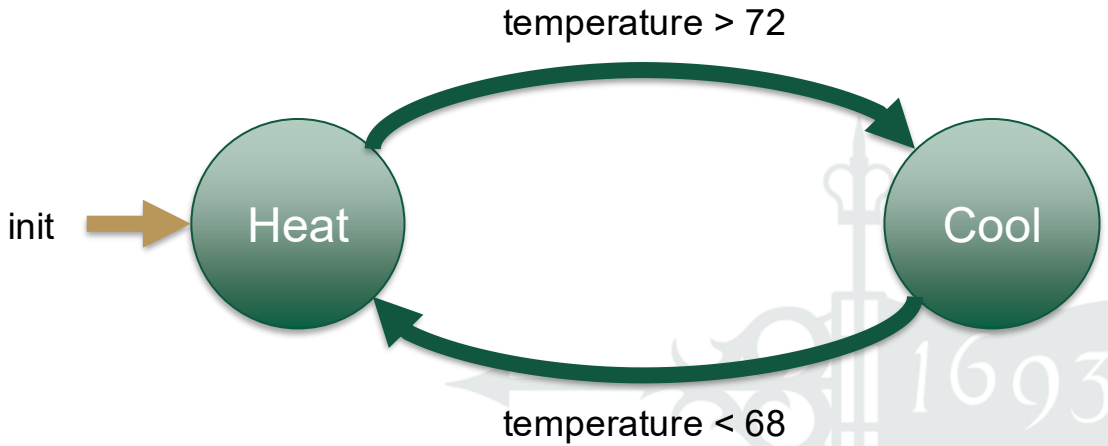
Time	State	Temp
0	Heat	70



Timed State Machines

- Don’t wait for events – check at interval

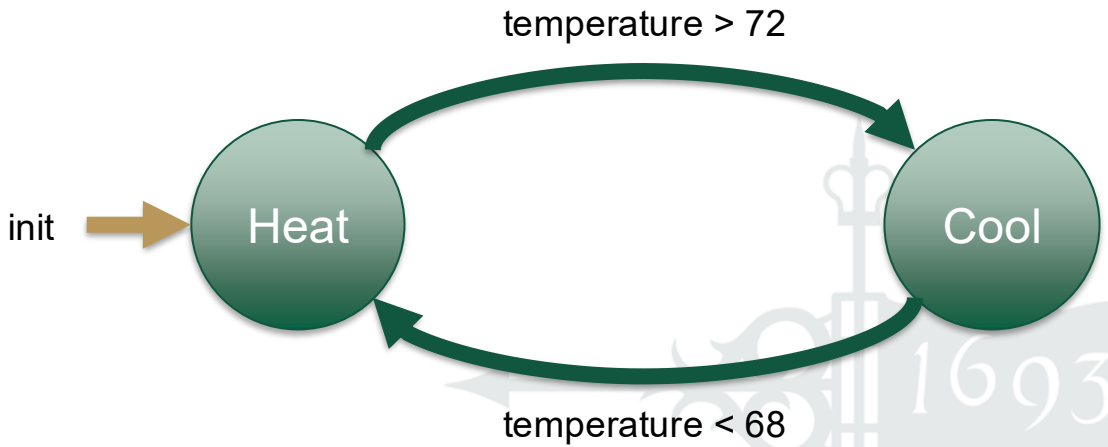
Time	State	Temp
0	Heat	70
1	Heat	73
2	Cool	72
3	Cool	68
4	Cool	65
5	Heat	68
6	Heat	73



How do we use states?

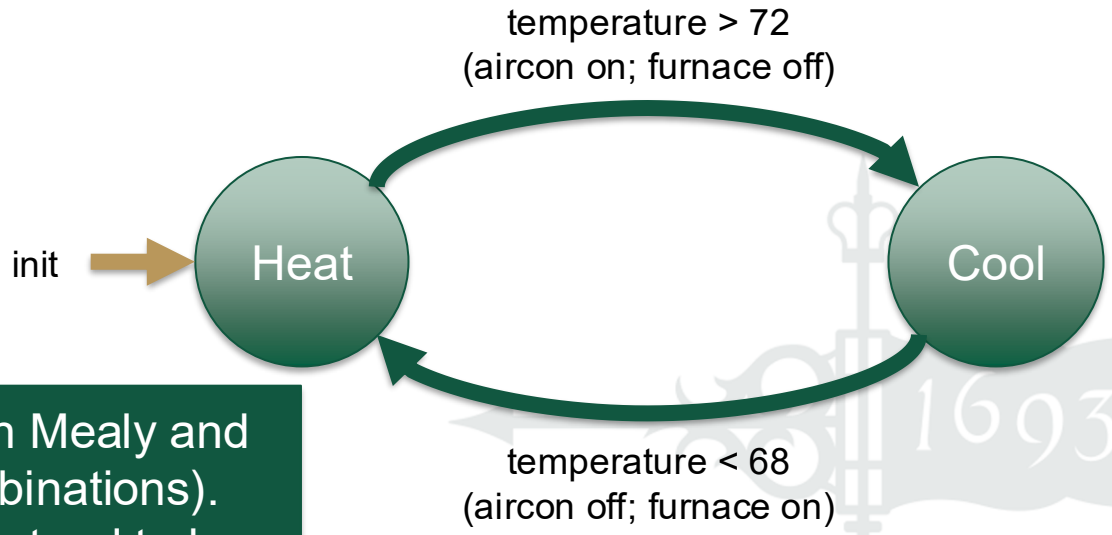
- Change behavior based on state

```
if state == HEAT:
  # turn on heat
  aircon = false
  furnace = true

elif state == COOL:
  # turn on cool
  aircon = true
  furnace = false

else:
  # invalid state
```



Mealy Machine

- Change behavior *during transition*



You can write code for both Mealy and Moore machines (or combinations). However, Moore machines tend to be easier to maintain and debug.

State Machines are Complete

- Behavior is always defined