Lab 2B

State Machines

This lecture is part of the RACECAR-MN introductory robotics course.
You can visit the course webpage at mitll-racecar-mn.readthedocs.io.

Objectives

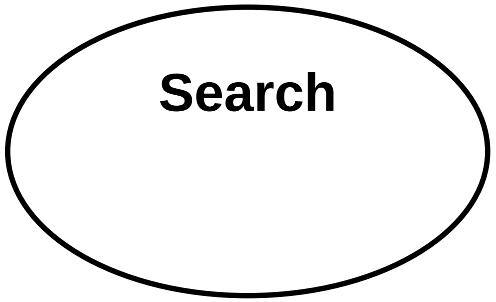
Main Objective: Write a fully autonomous racecar program to find and park near a cone

Learning Objectives

- Design and implement a state machine
- Use contour area to estimate object distance

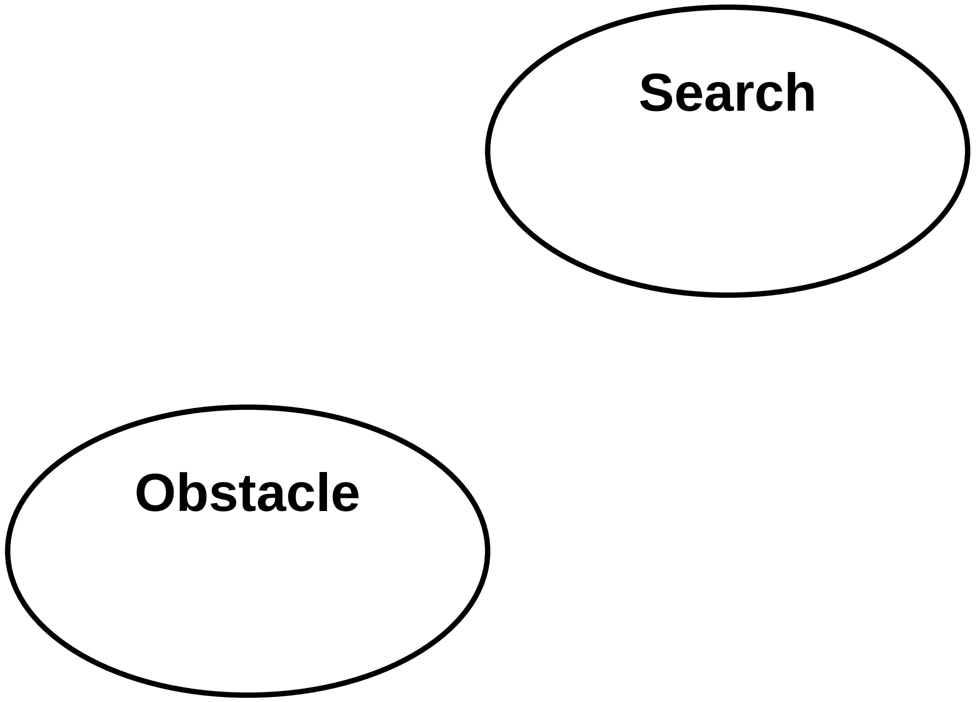
State Machines

Objective: explore an environment to find a cone



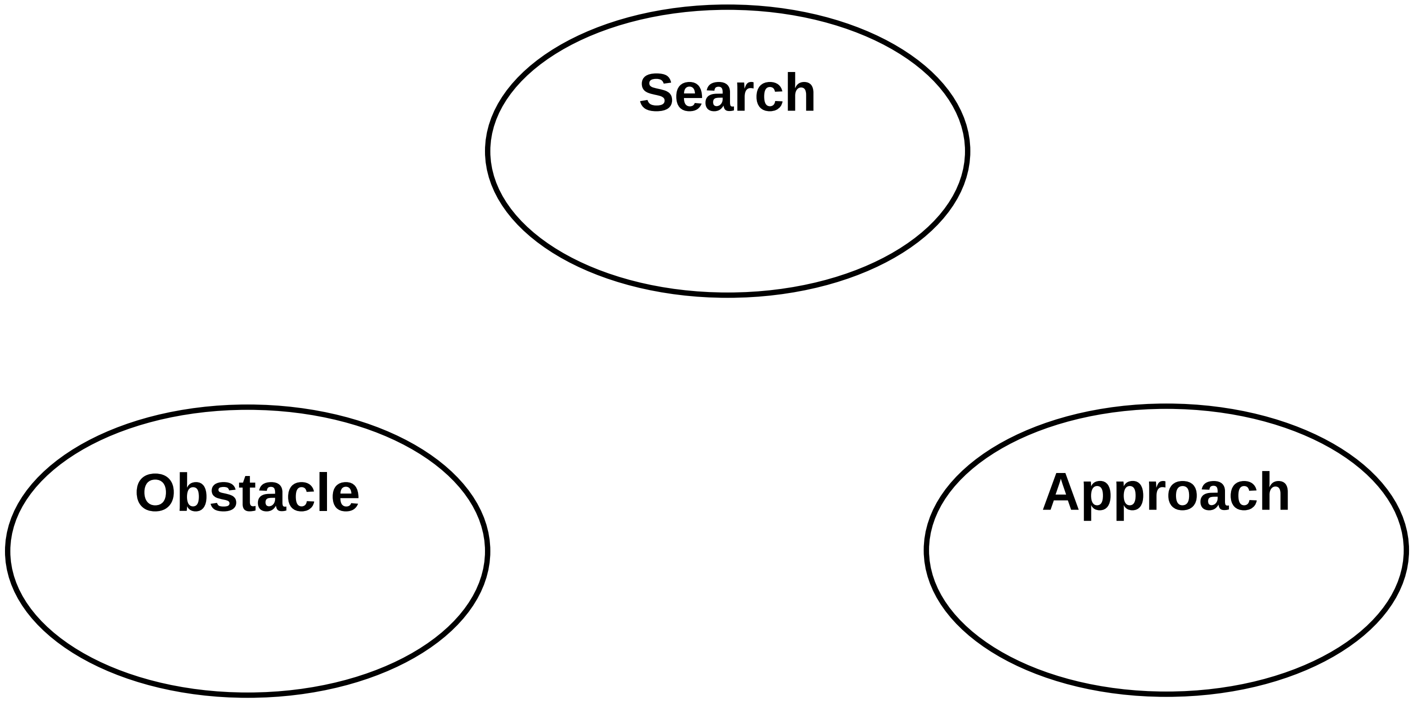
State Machines

Objective: explore an environment to find a cone



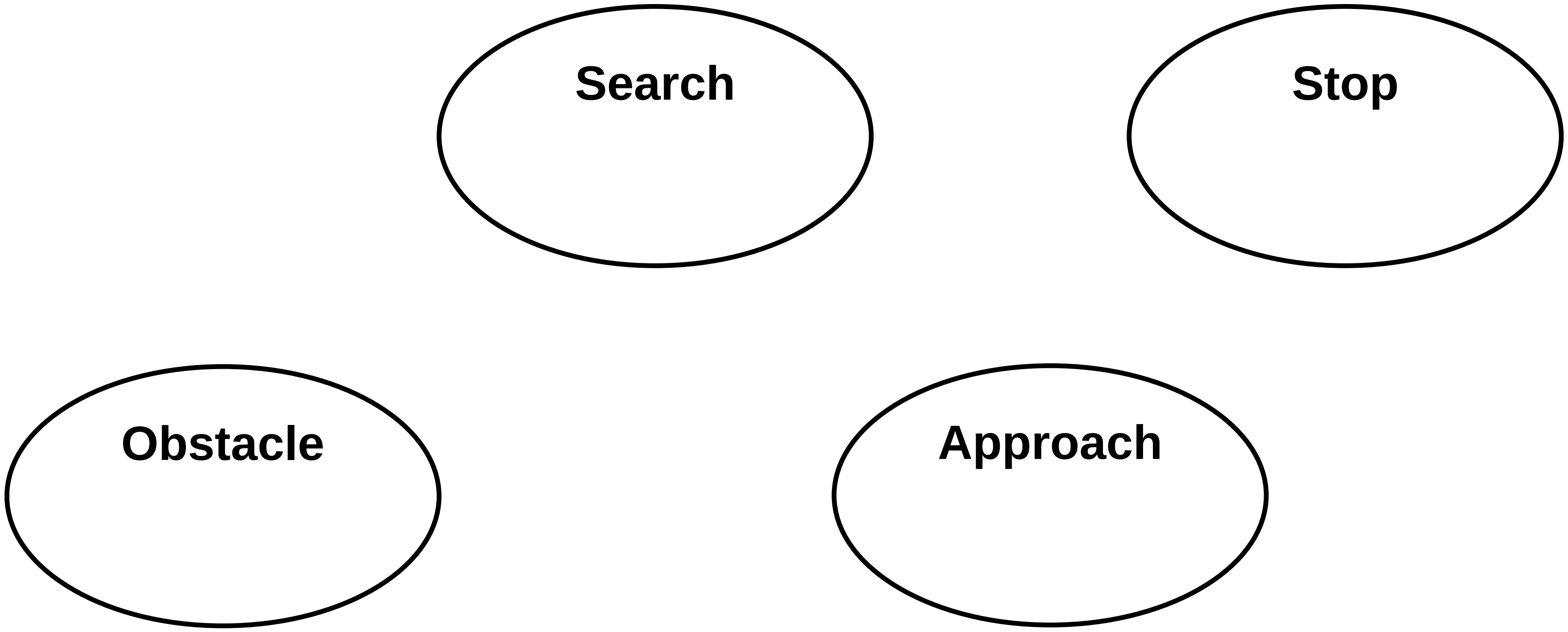
State Machines

Objective: explore an environment to find a cone



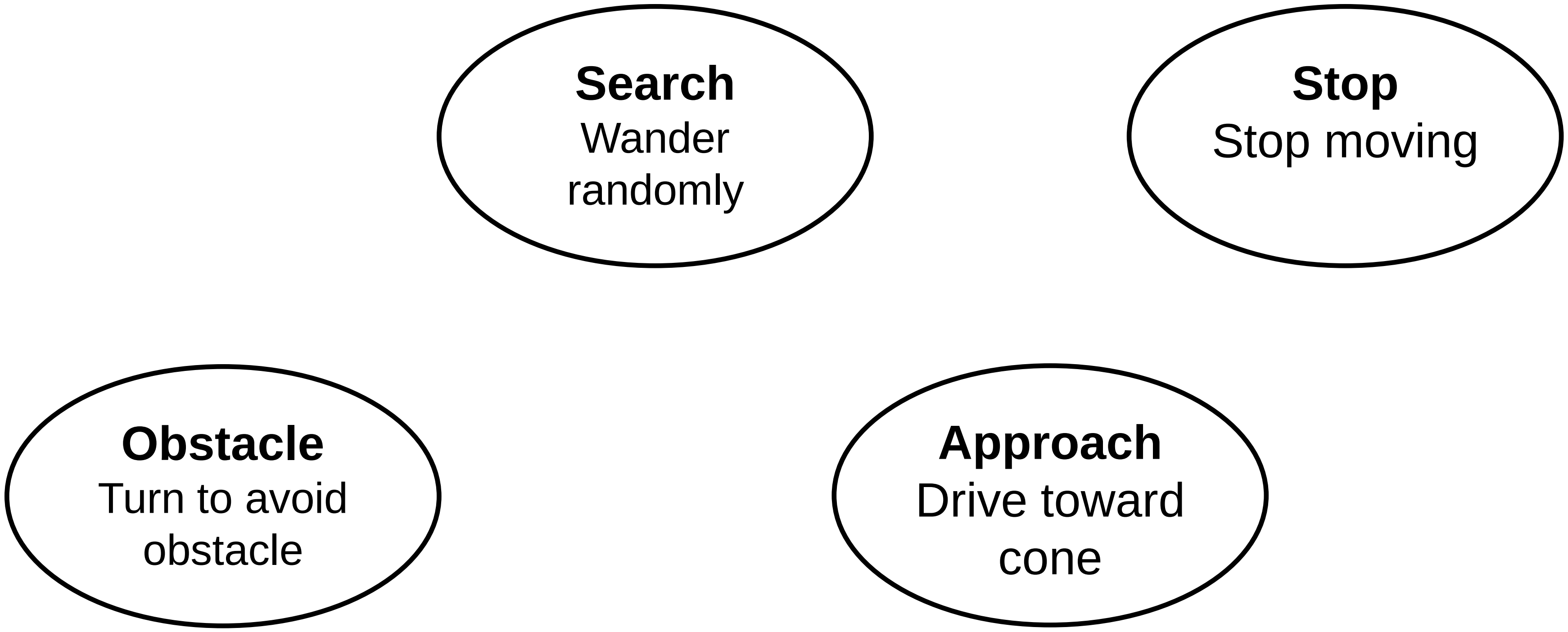
State Machines

Objective: explore an environment to find a cone



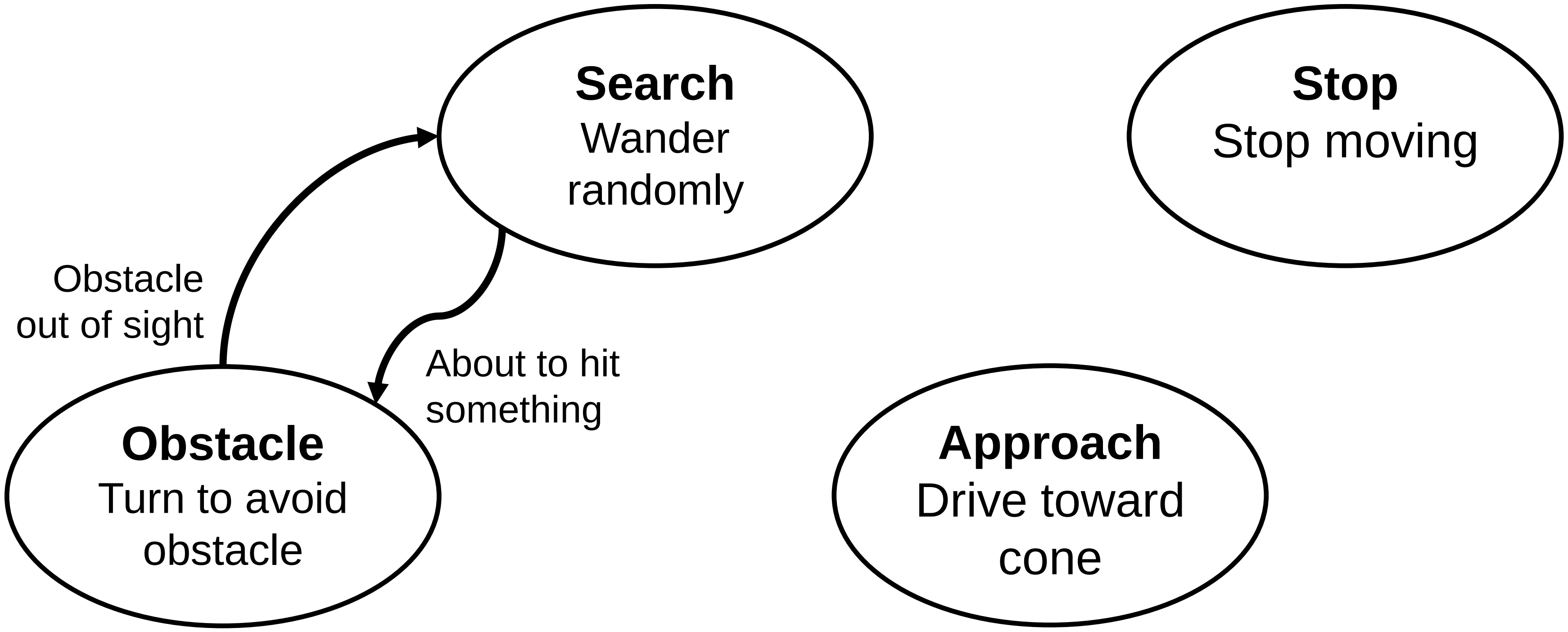
State Machines

Objective: explore an environment to find a cone



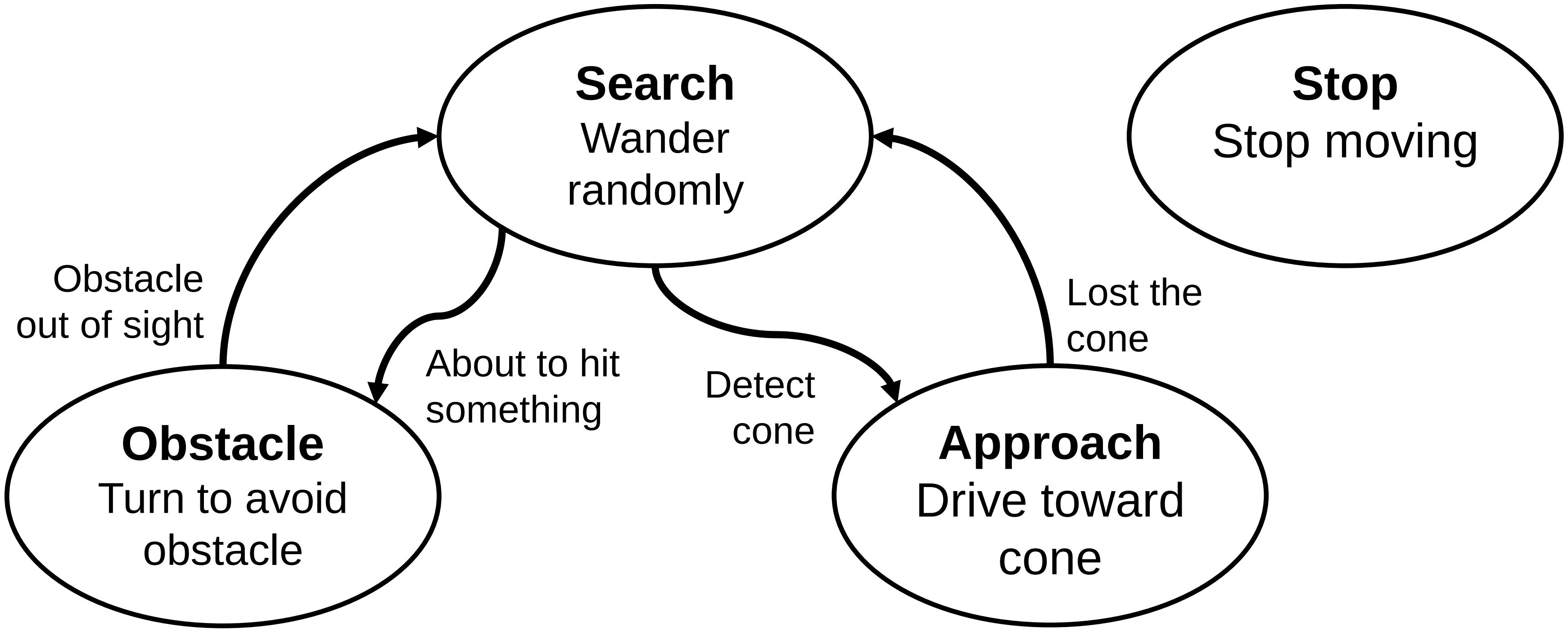
State Machines

Objective: explore an environment to find a cone



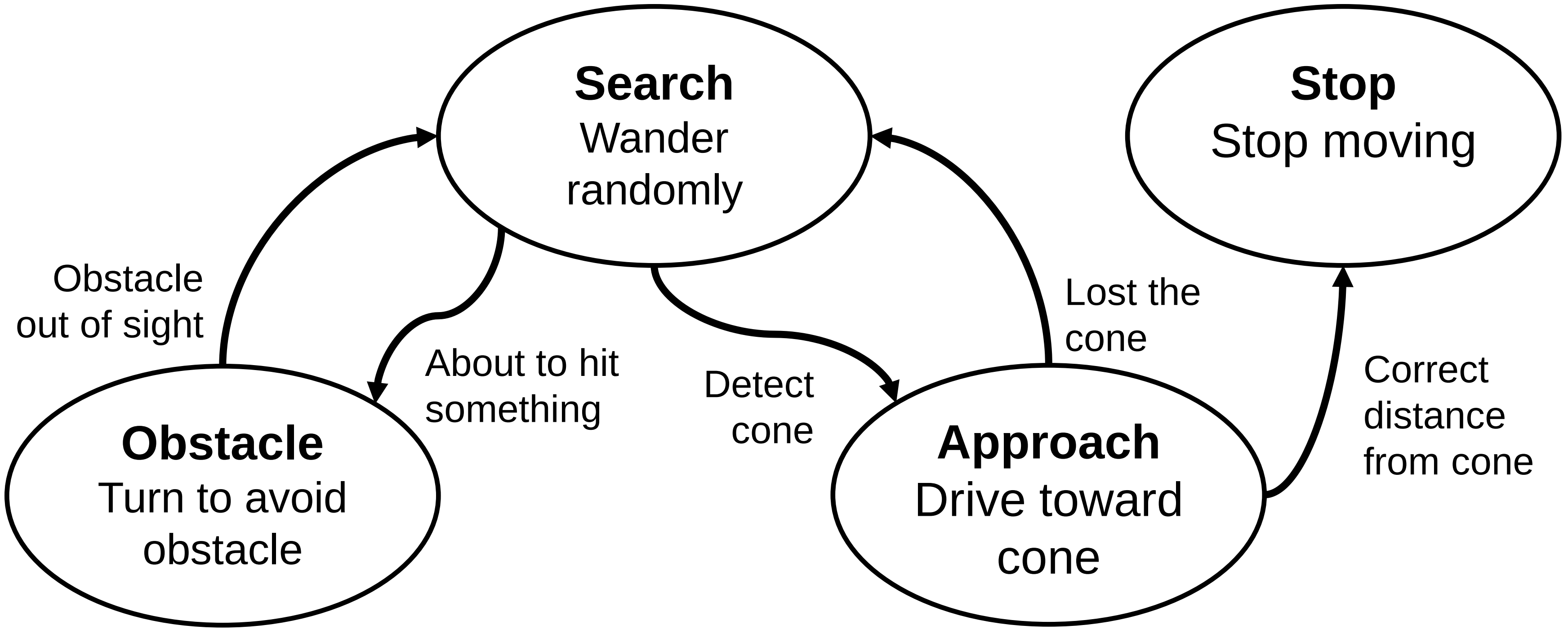
State Machines

Objective: explore an environment to find a cone



State Machines

Objective: explore an environment to find a cone



Implementation with Enums

```
from enum import IntEnum

class State(IntEnum):
    search = 0
    obstacle = 1
    approach = 2
    stop = 3

cur_state: State = State.search
```

Implementation with Enums

```python
from enum import IntEnum

class State(IntEnum):
    search = 0
    obstacle = 1
    approach = 2
    stop = 3

cur_state: State = State.search

def start():
    """
    This function is run once every time the start button is pressed
    """
    global cur_state
    cur_state = State.search
```

Implementation with Enums

```
def update():
    """
    After start() is run, this function is run every
    frame until the back button is pressed
    """
    global cur_state

    speed: float = 0
    angle: float = 0
    if cur_state == State.search:
        # Set speed and angle to "wander"

        if cone_identified:
            cur_state = State.approach

        if about_to_hit_something:
            cur_state = State.obstacle
```

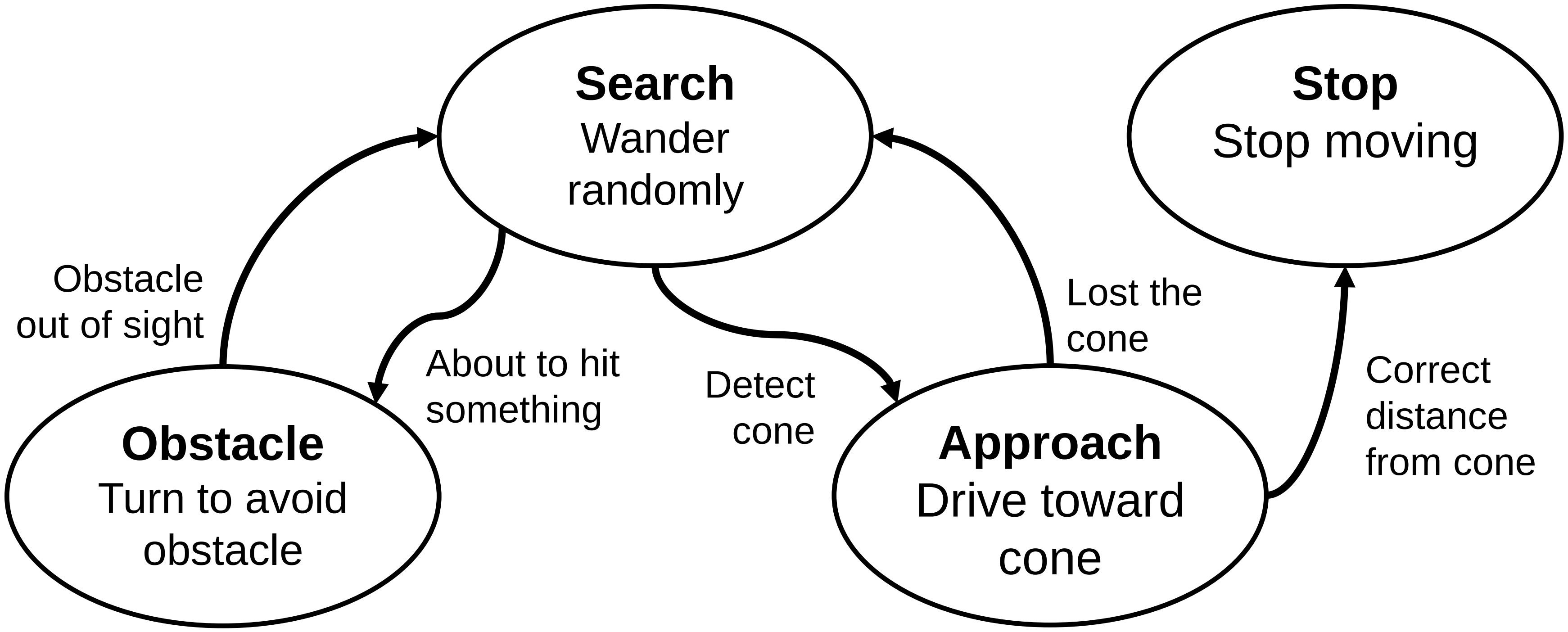
State behavior

State transitions
(arrows out of state)

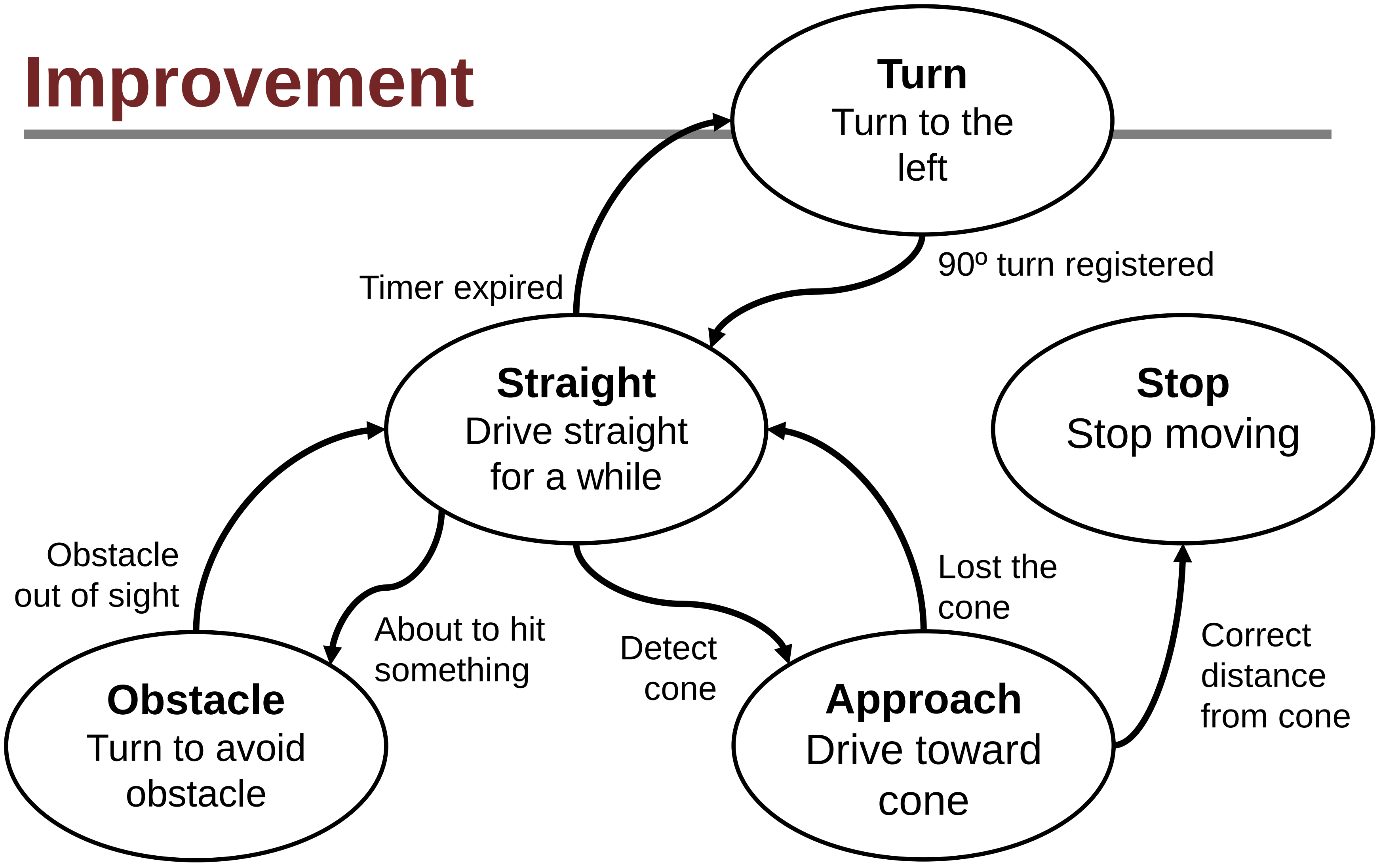
Implementation with Enums

```
elif cur_state == State.obstacle:
    # Set speed and angle to avoid obstacle

    if obstacle_avoided:
        cur_state = State.search

elif cur_state == State.approach:
    # Set angle to face cone and approach

    if next_to_cone:
        cur_state = State.stop

    if not cone_identified:
        cur_state = State.search

elif cur_state == State.stop:
    speed = 0
    angle = 0

rc.drive.set_speed_angle(speed, angle)
```

Improvement



Improvement



Design Strategy

1. Brainstorm the necessary states
2. Determine the action in each state
3. Determine when to transition between states
4. **Iterate**: add states/tune relationships as needed

Implementation Strategy

1. Create `State` enum and global variable `cur_state`
2. Initialize enum to starting state in `start`
3. In `update`, do the following for each state:
   – Create an `if` block
   – Set car outputs (speed, angle, etc.)
   – Check for state transitions out of that state

Exercise

Group activity

- Design a state machine for a "rideshare" program:
  - Wanders until receiving a ride request
  - Drives to requester, then to their location
- Bonus objectives
  - How to handle stop signs and stop lights
  - How to pass another car
- If you have extra time, start converting your state machine into Python (you can use lots of pseudo-code)

Lab 2 – Cone Parking

- Objective: Identify an orange cone, align the car with the cone, and park 30 cm away
  - Consider what to do if the cone is far from the center and very close to the car
  - Stretch goal: how to handle if the cone is out of sight to start?

lab2b.py