Behavior Based Control

Week #7

Prof. Ryan Kastner

Oh Behave!

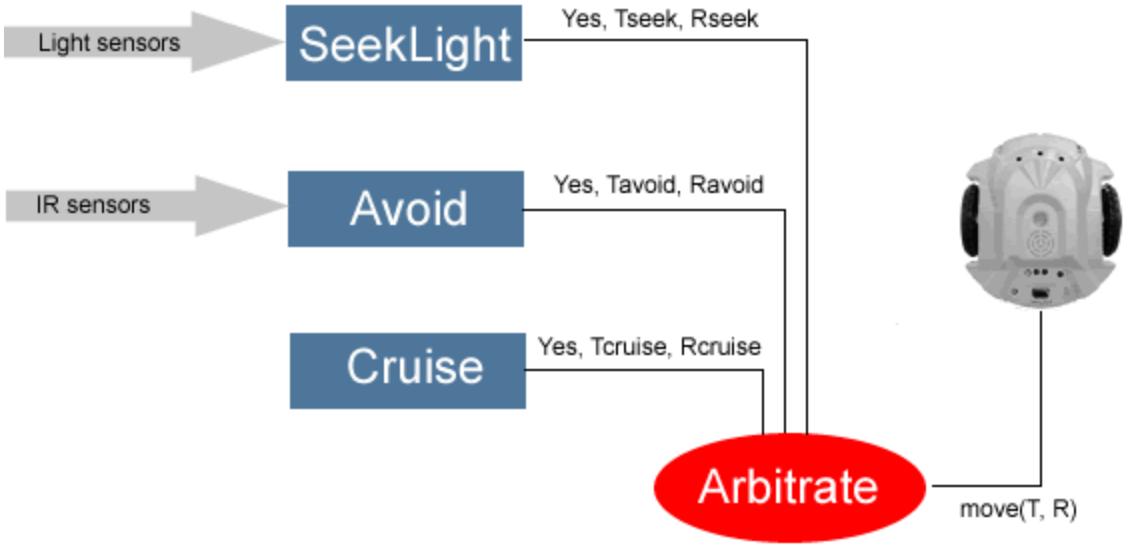
- A program is all about exercising control
- Python programs control the computer which communicates with the Myro
- When writing robot control programs, the structure you use to organize the program itself is a control strategy
- Programming a robot is specifying automated control
- Sensing and control together form Reactive Control

Structuring Robot Programs

- Structuring robot programs makes designing behaviors easy
- Sensor Fusion: Just another buzz name for Reactive Control or Direct Control
- In *behavior-based control* you get away from sensors and focus the design of your robot programs based on the number and kinds of behaviors your robot has to carry out

Behavior in a Maze

- The robot in a Maze has three behaviors:
  - Cruise (If there is no obstacle)
  - Avoid Obstacles (If present)
  - Seek Light (If present)
- Define each behavior as an individual decision unit



Design Each Behavior

```
cruiseSpeed = 0.8
turnSpeed = 0.8
lightThresh = 80

def cruise():
# is always ON, just move forward
return [True, cruiseSpeed, 0]

def avoid():
    # see if there are any obstacles
    L, R = getIR()
    L = 1 - L
    R = 1 - R
    if L:
        return [True, 0, -turnSpeed]
    elif R:
        return [True, 0, turnSpeed]
    else:
        return [False, 0, 0]
```

```
def seekLight():
    L, C, R = getLight()
    if L < lightThresh:
        return [True, cruiseSpeed/2.0, turnSpeed]
    elif R < lightThresh:
        return [True, cruiseSpeed/2.0, -turnSpeed]
    else:
        return [False, 0, 0]
```

Arbitration Schemes

- To control the Robot, one has to decide which recommendation to chose
  - Priority Assignment: Also called subsumption architecture
    - Higher the module in the figure, higher the priority
- By arranging control:
  - Design of each behavior is easy
  - Testing becomes easy
  - More behaviors can be added

Names and Return Values

- In Python a name can represent anything as its value: a number, a picture, a function, etc.
  - E.g. : *behaviors = [seekLight, avoid, cruise]*
  - List named behaviors is a list of function names each of which denote the actual function as its value

```
for behavior in behaviors:
    output, T, R = behavior()
```

- In each iteration of the loop, the variable behavior takes on successive values from this list: *seekLight*, *avoid*, and *cruise*

Design Main and Arbitrate

```
# list of behaviors, ordered by priority (left is highest)
behaviors = [seekLight, avoid, cruise]

def main():
    while True:
        T, R = arbitrate()
        move(T, R)

main()

  # Decide which behavior, in order of priority
  # has a recommendation for the robot

  def arbitrate():
      for behavior in behaviors:
          output, T, R = behavior()
          if output:
              return [T, R]
```

Math in Python

- Python provides a set of libraries so you don't have to write them ☺
- *math* library: *from math import* *
  - ***ceil(x)*** Returns the ceiling of *x* as a float, the smallest integer value greater than or equal to *x*
  - ***floor(x)*** Returns the floor of *x* as a float, the largest integer value less than or equal to *x*
  - ***exp(x)*** Returns e^x

Functions in Math

- ***log(x[, base])*** Returns the logarithm of x to the given base. If the base is not specified, return the natural logarithm of x (i.e., $log_e x$)
- **log10(x)** Returns the base-10 logarithm of x (i.e. $log_{10} x$)
- **pow(x, y)** Returns x^y
- **sqrt(x)** Returns the square root of x ($\sqrt{x}$)

```
import math
>>> math.ceil(5.34)
6.0
```

Summary

- Behavior based Control
- Mathematical Functions