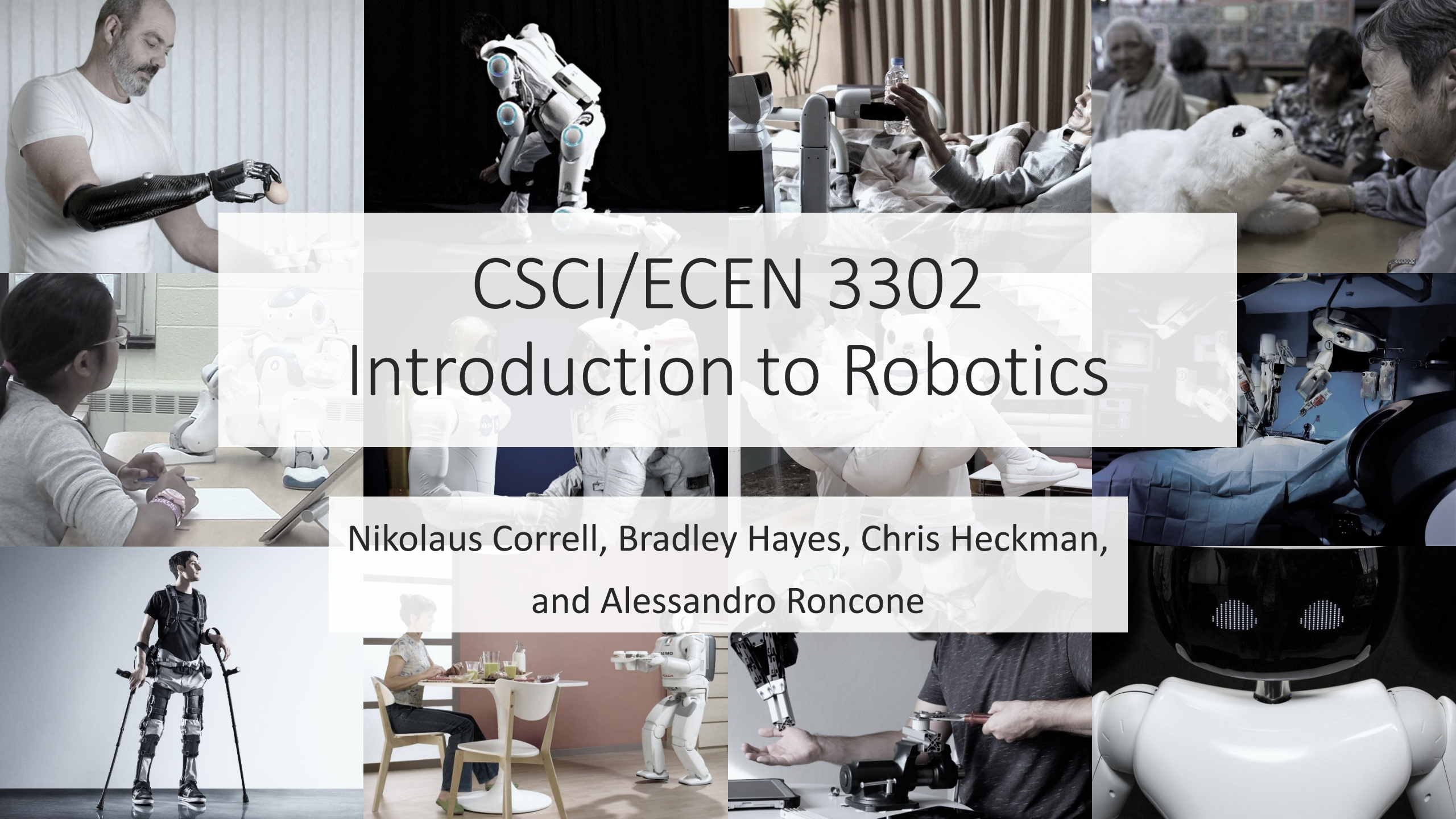
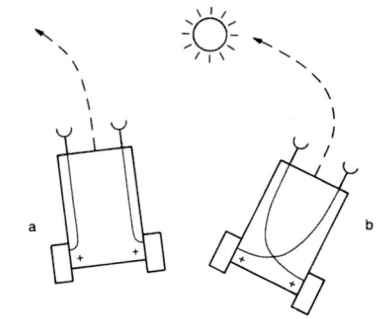
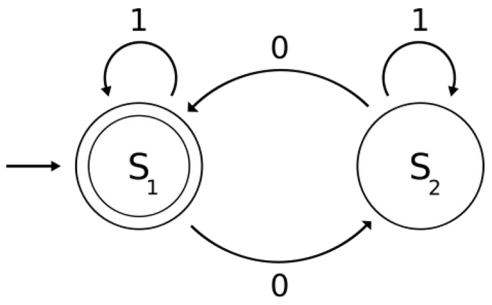


CSCI/ECEN 3302
Introduction to Robotics

Nikolaus Correll, Bradley Hayes, Chris Heckman,
and Alessandro Roncone

Lab 1
Reactive Behaviors and State Machines





Types of Robot Architectures

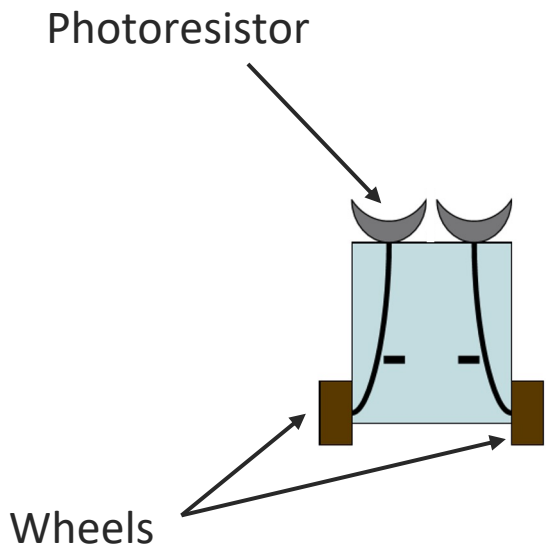
Reactive

- Connect sensing directly to action
- Stimulus-response
  (e.g., ‘knee-jerk reaction’)

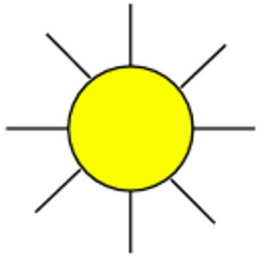
Deliberative

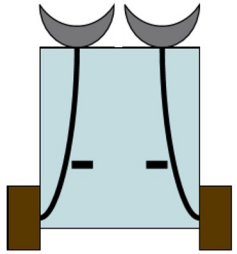
- Sense-Plan-Act
- *Classical* control systems
- Reason over abstraction of the world
  (e.g., playing chess)

Reactive Behaviors

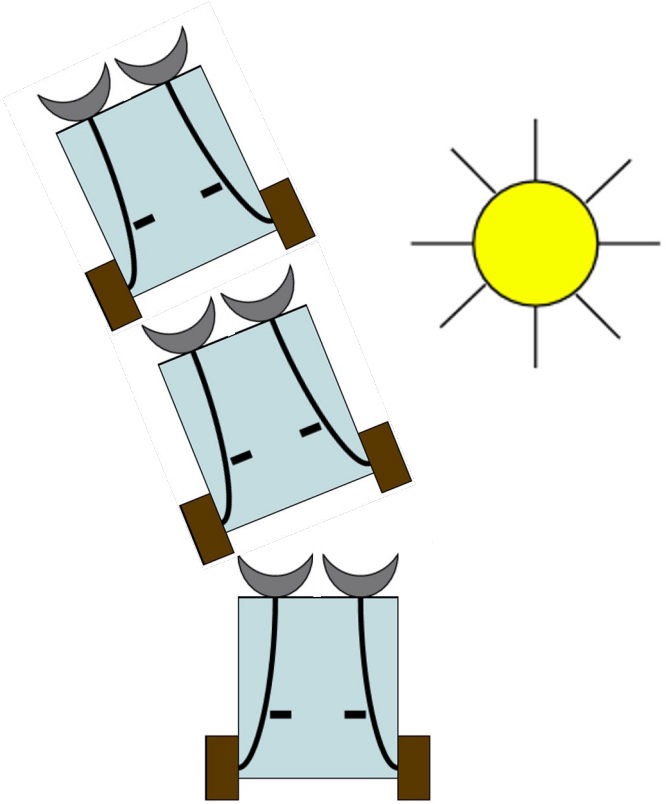


Reactive Behaviors

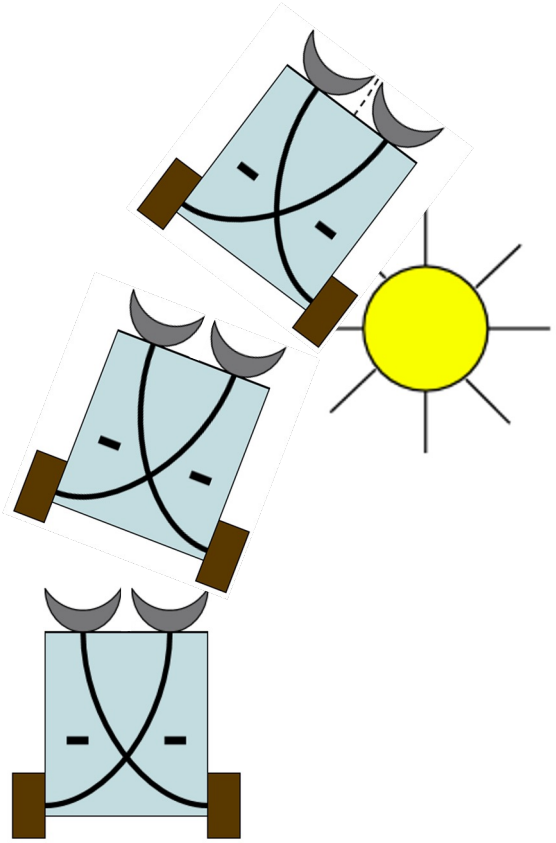




Reactive Behaviors

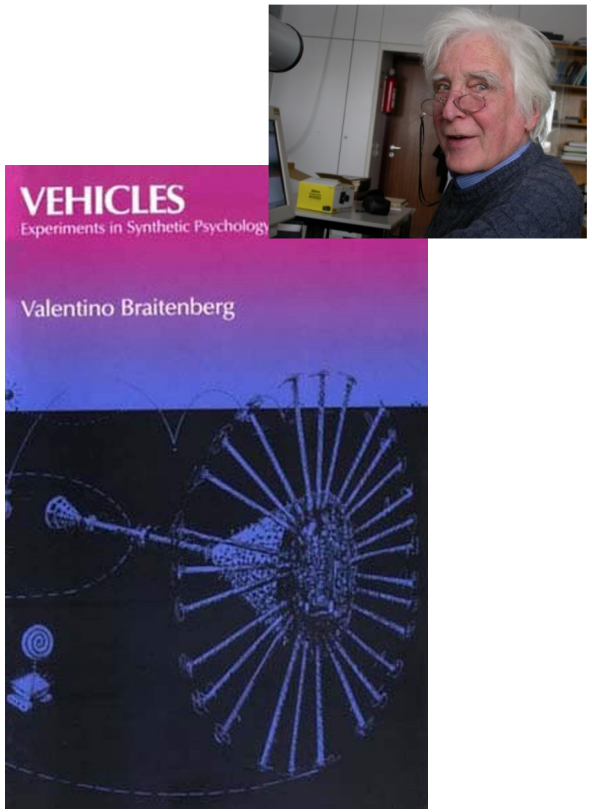


Reactive Behaviors

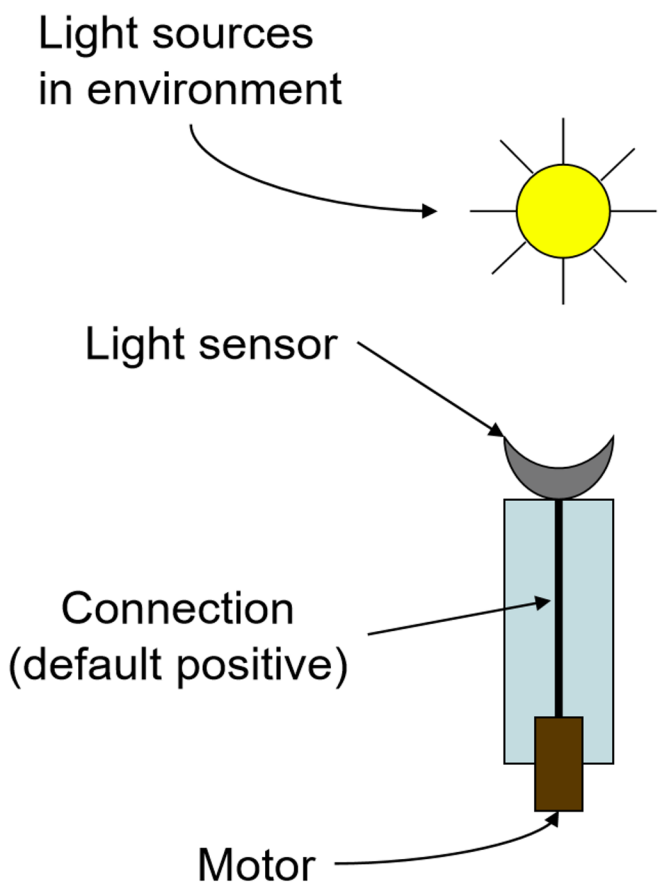


Valentino Braitenberg

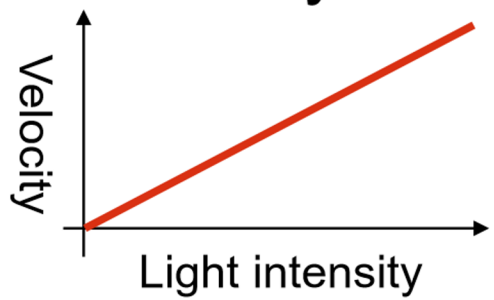
- Neuro-anatomist
- Professor of Cybernetics
- Director of Max Planck Institute of Biological Cybernetics
- Published “Vehicles” in 1984
  (based on papers from 1965)



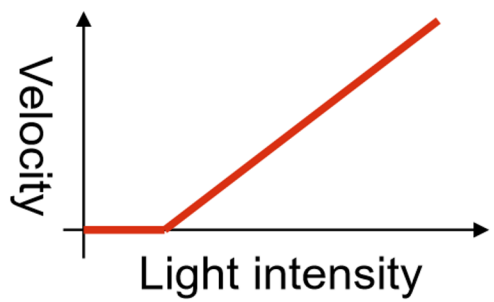
Vehicle 1



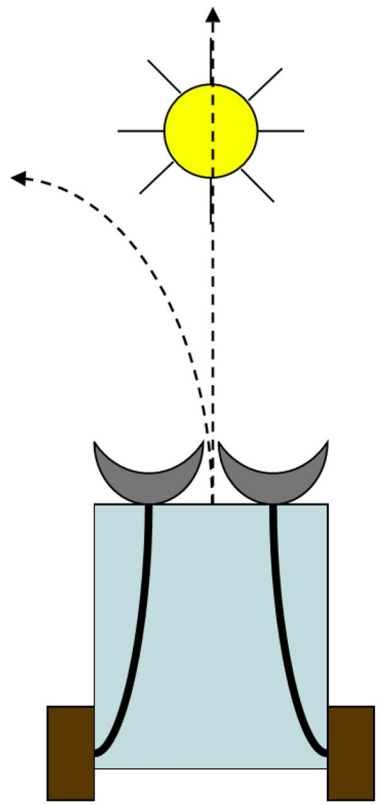
- Speed is proportional to the light received by the sensor



- Environment (friction) can introduce non-linear behavior



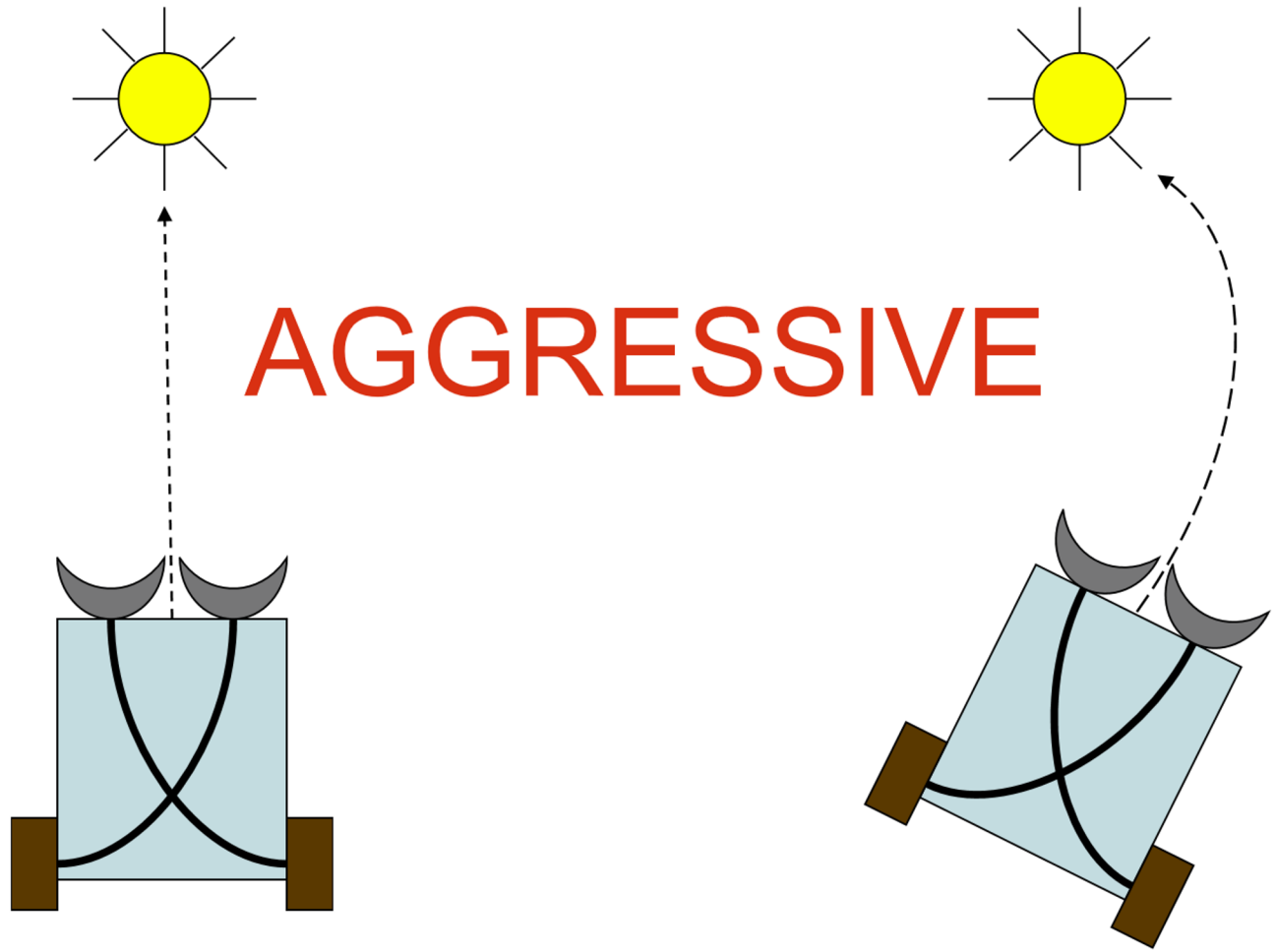
Vehicle 2a



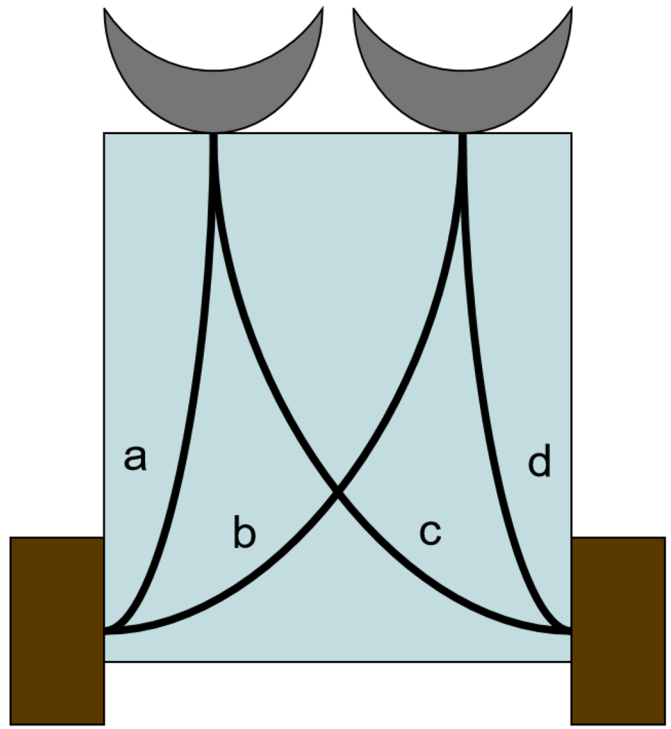
COWARD

- 2 wheels, 2 sensors
- What happens with a perfect on-axis approach?
- What happens if there is any deviation?
  - Sensor noise
  - Differences in friction
  - Transmission fluctuations/reflections

Vehicle 2b

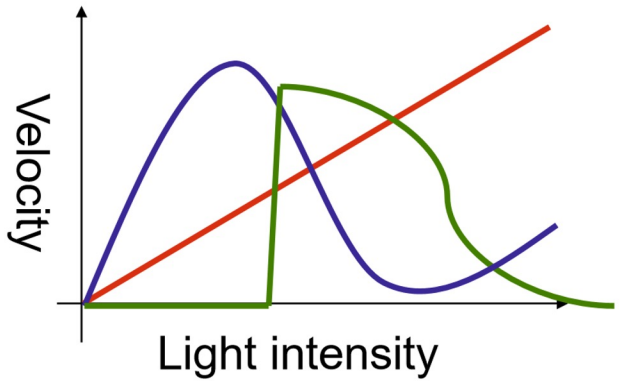


Vehicle 3c

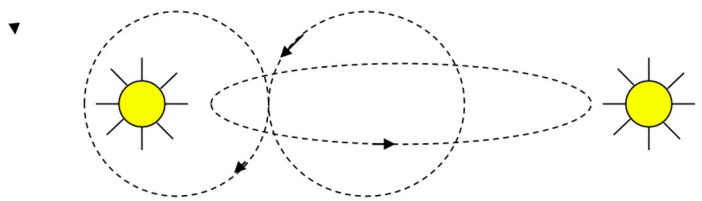


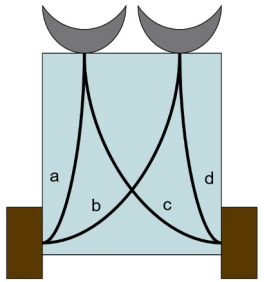
- Allow all possible connections, with arbitrary weights on each connection
- Displays a set of “preferences” or “values”

Vehicle 4



- Non-monotonic connection functions
- Beginning of “instincts” or “choice-like” behavior

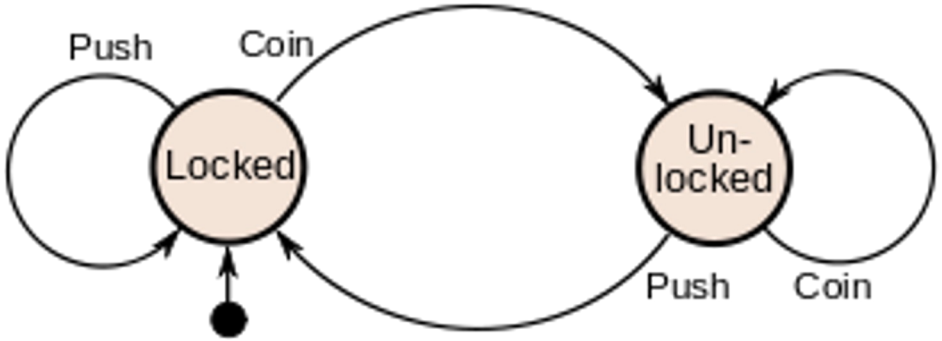




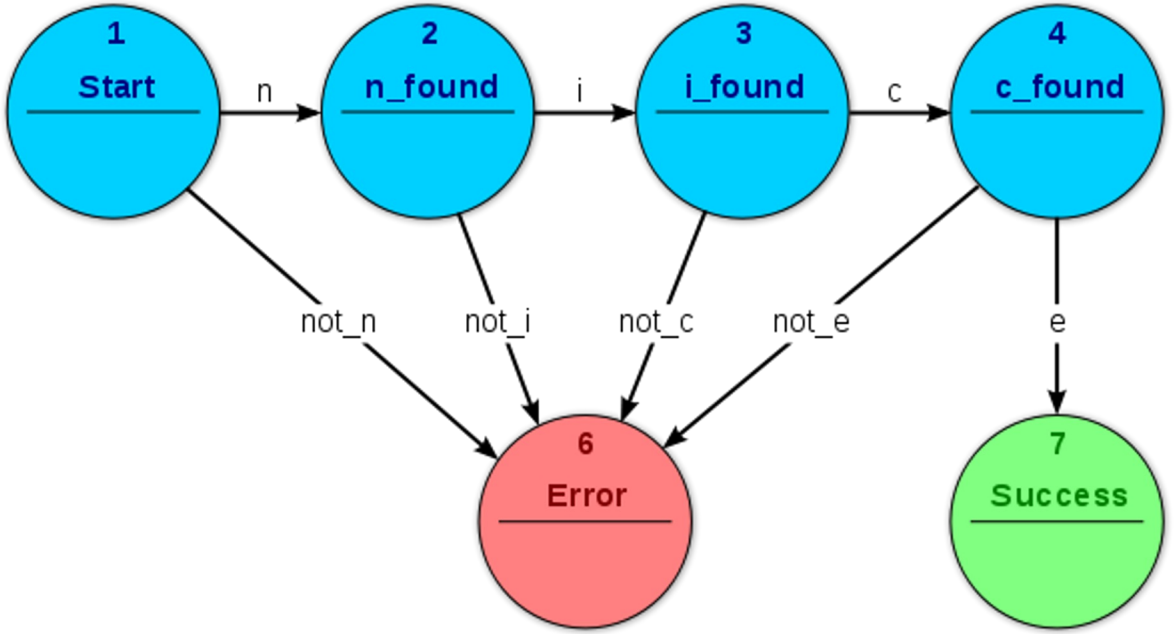
Observations

- “Law of uphill analysis and downhill invention”
  - Very challenging to figure out the internal control mechanisms of even very simple agents strictly through observation
  - It’s easier to produce complex behaviors by combining simple parts
  - When analyzing mechanisms, there is a tendency to overestimate its complexity

State Machines

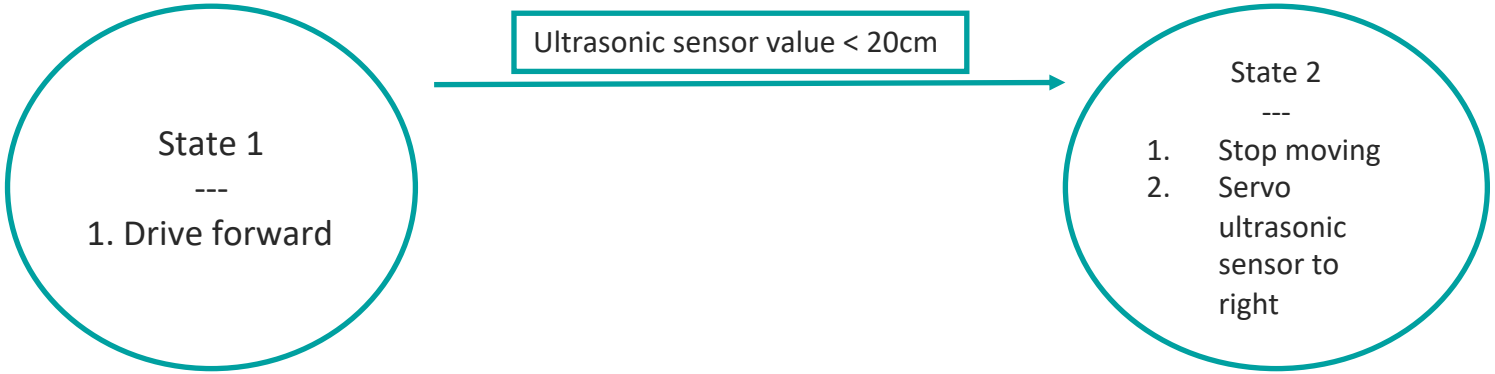


“nice”



Why use a state machine?

- They are a convenient way to define multiple behaviors and the conditions that determine when a robot should use each.
- Allows for parallel implementation on developer teams
  - Each ‘state’ of the machine (representing a behavior the robot should exhibit) can be implemented independently.



Sanity-preserving Tips:

- Plan first
- Code last
- Comment often

Need Help? Send a chat message!

Group programming IDEs: www.codeshare.io | http://collabedit.com/

FAQs:

- What's due at the end of today?
  - Nothing, this is a 2-week lab.
- What do I turn in?
  - ONE person needs to turn in the lab report and code per group.
- We're done! Can we leave?
  - Yep. Lab is meant to provide an interactive problem-solving time. If you complete the work early, you are free to go!

Code Suggestions

- Use global variables for your sensor values
- Use an if/elif chain to check which state your system is in!
- Use the provided sleep function if you need to delay your controller's execution instead of time.sleep()

```
current_state = 'STATE_1'
while robot.step(TIME_STEP) != -1:
    if current_state == 'STATE_1':
                    pass
    elif current_state = 'STATE_2':
                    pass
```

```
def sleep(duration):
    # Waits for duration seconds before returning.
    global robot
    end_time = robot.getTime() + duration
    while robot.step(TIME_STEP) != -1 and robot.getTime() < end_time:
        pass
    return
```

Questions?