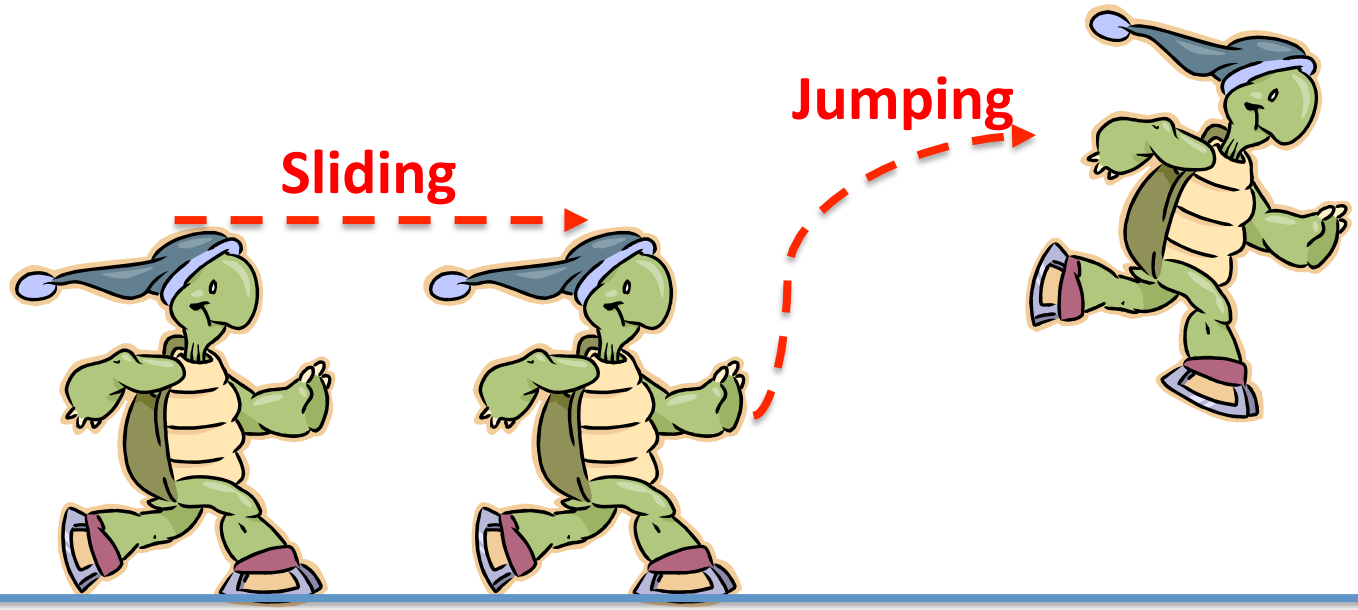
CSCI 1106
Lecture 14

State Transition Diagrams
(Modeling the Real World)

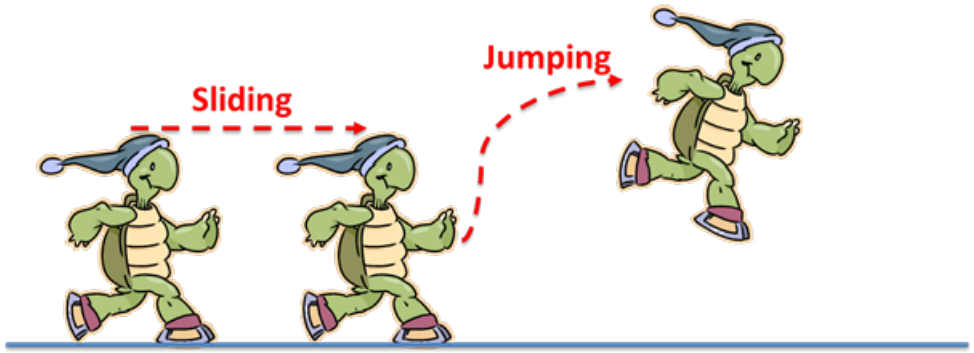


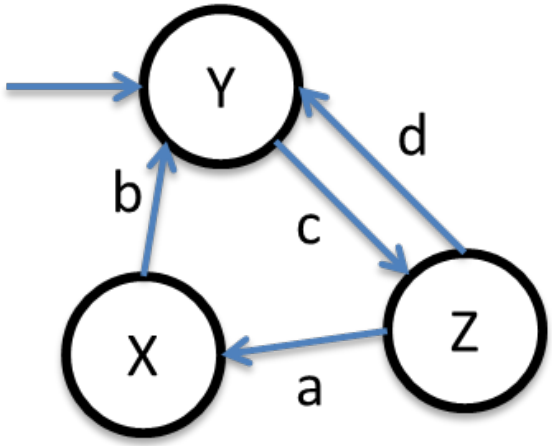
Announcements

- Today's Topics
  - Modeling Tasks
  - States and Transitions
  - State Transition Diagrams
  - Examples

State Transition Diagrams

- Modeling the real world
  - Modeling the robot's behaviour and environment.





Crossing at an Intersection

- If light is red, wait for light to turn green
- If light is yellow, wait for light to turn green
- If light is green but there is not enough time, wait for light to turn red and then green
- If light is green and there is enough time,
  - Proceed on crosswalk
  - **If a car is speeding at you, get out of the way**
- Stop crossing when other side is reached

Observations

- Even simple tasks are hard to specify
  - What are the steps?
  - When are the steps to be done?
  - Which steps need to be done?
- Specifying computer tasks is even harder
- Need a simple way to specify and model
  - Steps of a task
  - Conditions under which the steps are performed
  - Environment of the robot during the task
- Idea: Use state transition diagrams

State

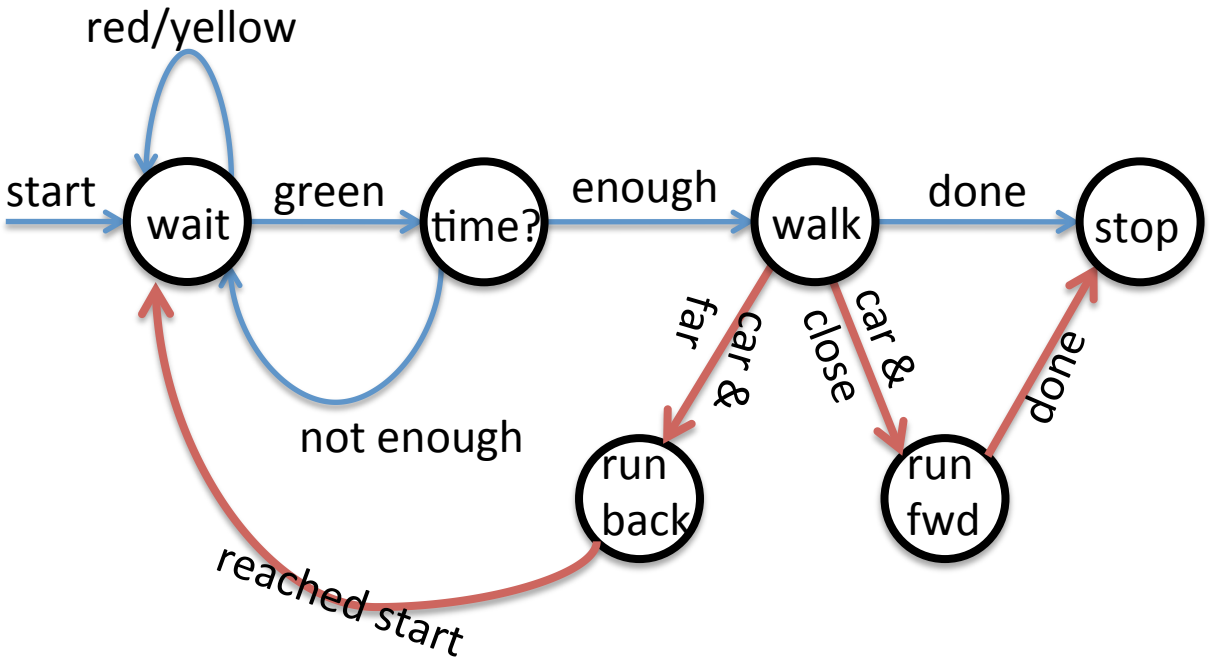
- A *state* is a unique set of conditions that hold at a given time
- Conditions include:
  - Measured or sensed properties of the environment
    - E.g., light is green and there is 20 seconds to cross
  - Current behaviour
    - E.g., Crossing the street
  - Current expectations
    - E.g., Will reach the other side without being run over
- Key Idea: A robot can be in one state at a time
- Robots can transition from one state to another state

State Transitions

- A *state transition* occurs when
  - An *event* occurs
  - One of the conditions describing the state changes
  - The state of the robot changes
- Transitions are typically caused by
  - External events
    - E.g. The stoplight changing colour
  - Internal event (Completion of a step in a task)
    - E.g. Completion of crossing the street

State Transition Diagrams

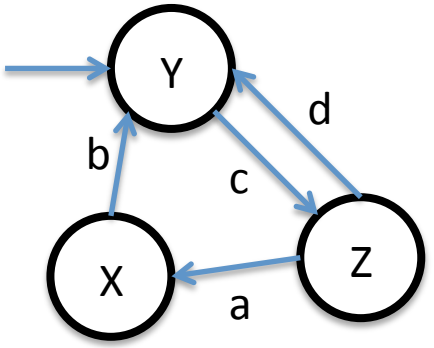
- Idea: We use a state transition diagram to model a task
- States are represented by circles
- Arrows represent transitions between states



- If light is red, wait for light to turn green
- If light is yellow, wait for light to turn green
- If light is green but there is not enough time, wait for light to turn red and then green
- If light is green and there is enough time,
  - Proceed on crosswalk
  - **If a car is speeding at you, get out of the way**
- Stop crossing when other side is reached

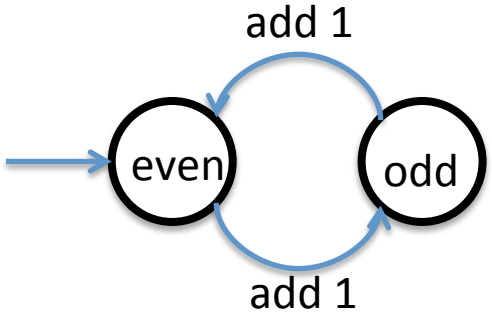
Creating State Transition Diagrams

- Identify the states (steps) of a task
  - Determine what **actions** must be performed
  - Determine groups of **unique** (relevant) conditions
  - Label each group with a **unique** name
- Identify state to state transitions
  - What is being sensed?
  - What external events will be sensed?
  - What internal events will occur?
  - What conditions will these events change?
  - Determine which conditions change?
  - Determine the corresponding states in the transition
  - Label each transition with a unique label
- Create diagram
  - Combine states and transitions
  - Refine the diagram by repeating the process
- **This diagram is a blueprint for your program!**



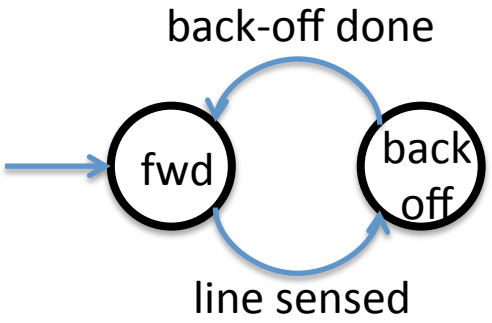
Determine if Number of People is Even

- Idea
  - Don't want to count people
  - Just keep track if # of people is odd or even
- States: (2)
  - Even
  - Odd
- Transitions:
  - Each additional person causes a transition to the other state



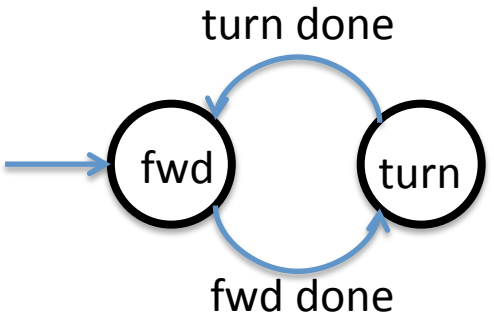
Avoid the Boundary

- Idea
  - Two actions
    - Move forward
    - Back off
  - Two events
    - Black line sensed
    - Finish back-off
- States: (2)
  - Forward
  - Back-off
- Transitions:
  - Line sensed (prox event)
  - Back-off done (timer event)



Move in a Square

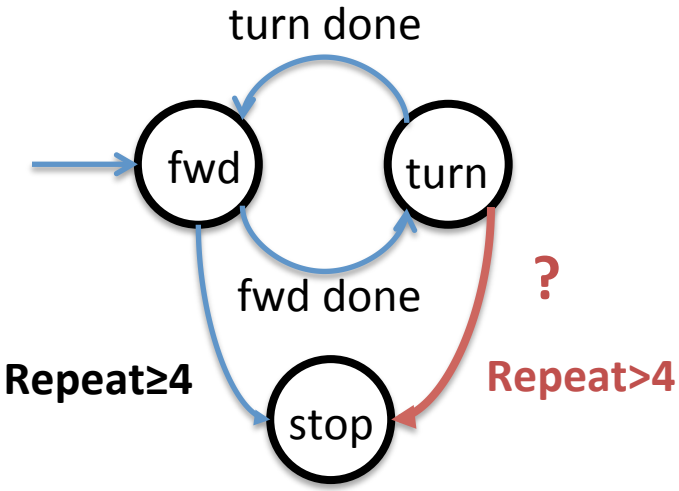
- Idea
  - Two actions
    - Move forward
    - Turn right
  - Two events
    - Finish straight move (timer expired)
    - Finish right turn (timer expired)
- States: (2)
  - Forward
  - Turn
- Transitions:
  - On timer events
    - (timers expire)




```
onevent timer0
  motor.left.target = -motor.left.target
  if motor.left.target < 0 then
    timer.period[0] = TURN_PERIOD
  else
    timer.period[0] = FWD_PERIOD
  end
```

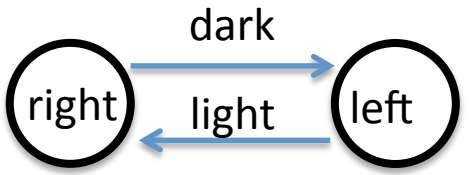
Make One Square

- Idea
  - Two actions
    - Move forward
    - Turn right
    - Repeated 4 times
  - Two events
    - Finish straight move (timer)
    - Finish right turn (timer)
- States: (?)
  - Forward?
  - Turn?
  - …
- Transitions:
  - When timers expire
  - …



Follow the Line

- Setup
  - Actions?
  - Events?
- States: (?)
- Transitions: ?



Determine if Number of People is Divisible by 3

- Idea
  - Don't want to count people
  - Just keep track if # of people is divisible by 3
- States: (?)
- Transitions:
  - Each additional person causes a transition