CSCI 1106
Lecture 17

Using State Transition Diagrams
Announcements

• Today’s Topics
  – Implementing State Transitions
  – Mapping state transition diagrams to programs
  – When to use when
Recall

States and Transitions

• State
  – Unique set of conditions
  – Describes a step of a task
  – Represented by a circle and a label

• Transition
  – Change of one or more conditions
  – Describes a change from one state to another
  – Represented by a labeled arc

Program Code

```plaintext
var state = STOPPED
motor.left.target = 0
motor.right.target = 0

onevent button.forward
  state = RIGHT

onevent button.backward
  state = STOPPED
  motor.left.target = 0
  motor.right.target = 0

onevent prox
  if state != STOPPED then
    when prox.ground.delta[0] >= THRESHOLD do
      state = RIGHT
      motor.left.target = TARGET
      motor.right.target = 0
    end

    when prox.ground.delta[0] < THRESHOLD do
      state = LEFT
      motor.left.target = 0
      motor.right.target = TARGET
    end
  end
```

right → light → left

dark
Translating State Transition Diagrams

• Problem:
  – We design our solution by creating a state transition diagram (STD)
  – We need to translate the STD into a program

• Idea: Use a standard process
  – Use a variable to encode the current state
  – Enumerate all states as constants
  – Identify events associated with each transition
  – Gather transition information
  – Implement event handlers to perform the transitions
Tracking and Enumerating States

• Use a state variable
  – Stores the current state
  – Set to an initial state, e.g., STOPPED

• Enumerate all states
  – Select state names e.g., STOPPED, RIGHT, LEFT
  – Number consecutively
  – Add states as constants

• Can be done automatically

```javascript
var state = STOPPED;
motor.left.target = 0;
motor.right.target = 0;

onevent button.forward
state = RIGHT;

onevent button.backward
state = STOPPED;
motor.left.target = 0;
motor.right.target = 0;

onevent prox
if state != STOPPED then
  when prox.ground.delta[0] >= THRESHOLD do
    state = RIGHT;
motor.left.target = TARGET;
motor.right.target = 0;
  end
end
when prox.ground.delta[0] < THRESHOLD do
  state = LEFT;
motor.left.target = 0;
motor.right.target = TARGET;
end
end
```
Identify Events

• Identify the events associated with each transition
  – `button.forward`: Forward Button pressed
  – `prox`: horizontal proximity or ground proximity sensors
  – `timer0` or `timer1`: timer has expired
  – `tap`: robot tapped
  – etc

• Add an event handler for each event
  – `onevent button.forward`
  – `onevent prox`
  – `onevent timer0`

• In each handler implement all the transitions associated with the event
Example: Identify Events

Events
- button.forward
- button.backward
- prox

onevent button.forward
...

onevent button.backward
...

onevent prox
...
Gather Transition Information

• For each transition, identify
  – States (CONSTANTS)
  – Event (handler)
  – Sensor/device
  – Change in sensor/device
  – Thresholds (if any)
  – Action to perform
• E.g., transition: fwd → left
  – States:
    • From: fwd (FORWARD)
    • To: left (LEFT)
  – Event (Handler): prox
  – Sensor: prox.ground.delta[0]
  – Change in sensor: response decreases (dark)
  – Threshold: < 500 means dark
  – Turn left
    motor.left.target = 0
    Motor.right.target = 200
• Implement the transitions in their event handlers
Implement the Transitions

• Inside the handler use template:
  
  ```java
  if state == FROM_STATE and sensor has changed then
    state = TO_STATE
    perform action
  end
  ```

• E.g., transition: fwd ➔ left

  ```java
  onevent prox
    if state == FORWARD and prox.ground.delta[0] < 500 then
      state = LEFT
      motor.left.target = 0
      motor.right.target = 200
    end
  ```
Optimizations

• Consider the handler for the "back button" transitions:
  ```
  onevent button.backward
  if state == FORWARD then
    state = STOPPED
    motor.left.target = 0
    motor.right.target = 0
  elseif state == LEFT then
    state = STOPPED
    motor.left.target = 0
    motor.right.target = 0
  elseif state == RIGHT then
    state = STOPPED
    motor.left.target = 0
    motor.right.target = 0
  end
  ```

• Why is there no AND part?
• Is this necessary?
  ```
  onevent button.backward
  state = STOPPED
  motor.left.target = 0
  motor.right.target = 0
  ```

• In many cases code can be optimized!
Another Example: Move in a Square

The Square STD

- States:
  - stop (STOPPED)
  - Fwd (FORWARD)
  - turn (TURN)

- Transitions:
  - stop $\rightarrow$ fwd
  - fwd $\rightarrow$ stop
  - turn $\rightarrow$ stop
  - fwd $\rightarrow$ turn
  - turn $\rightarrow$ fwd

- Events:
  - Forward Button
  - Backward Button
  - timer0
Timer based Transitions

Done Fwd
- States:
  - From: fwd (FORWARD)
  - To: turn (TURN)
- Event handler: timer0
- Device: timer.period[0]
- Thresholds: None
- Action:
  - Start turning
    motor.left.target = -200
    motor.right.target = 200
  - Set timer period
    timer.period[0] = 1000

Done Turn
- States:
  - From: turn (TURN)
  - To: fwd (FORWARD)
- Event handler: timer0
- Device: timer.period[0]
- Thresholds: None
- Action:
  - Start moving straight
    motor.left.target = 200
    motor.right.target = 200
  - Set timer period
    timer.period[0] = 2000
The timer0 Event Handler

```
onevent timer0
    if state == FORWARD then
        state = TURN
        timer.period[0] = 1000
        motor.left.target = -200
        motor.right.target = 200
    elseif state == TURN then
        state = FORWARD
        timer.period[0] = 2000
        motor.left.target = 200
        motor.right.target = 200
    end
```

The Other Event Handlers

```
onevent button.forward
    state = FORWARD
    timer.period[0] = 2000
    motor.left.target = 200
    motor.right.target = 200

onevent button.backward
    state = STOPPED
    timer.period[0] = 0
    motor.left.target = 0
    motor.right.target = 0
```
Using elseif

Right

onevent timer0
  if state == FORWARD then
    state = TURN
    timer.period[0] = 1000
    motor.left.target = -200
    motor.right.target = 200
  elseif state == TURN then
    state = FORWARD
    timer.period[0] = 2000
    motor.left.target = 200
    motor.right.target = 200
  end

Wrong

onevent timer0
  if state == FORWARD then
    state = TURN
    timer.period[0] = 1000
    motor.left.target = -200
    motor.right.target = 200
  end

Key Idea: Multiple transitions in one event handler should be linked with elseif
What is when?

- Consider our line follower (again)
- Which transitions occur on a prox event?
  - on black line
    - fwd ➔ left
    - right ➔ left
  - on no line
    - fwd ➔ right
    - left ➔ right
- Observation:
  - If we are not STOPPED
  - Transition to LEFT when we encounter a black line
  - Transition to RIGHT when we encounter no line
- Idea: Transitions occur when things change
- Analogy:
  - When we encounter a stop sign, we stop the car
  - We do not continue stopping the car once it has stopped
if vs when

if
• Form:
  
  if condition then
  body
  end
• If the condition is true
  the body is executed
• E.g., if we see a stop sign
  stop, regardless of whether
  we are already stopped

when
• Form:
  
  when condition do
  body
  end
• If the condition is true now
  and was not true before,
  the body is executed
• E.g., if we see a stop sign
  and we are not stopped,
  then stop
When to use when?

- **Idea:** Use *when* when the state of a sensor corresponds to a state

- **Examples:**
  - **Line following:**
    - Sensor registers dark means move left
    - Sensor registers light means move right
  - **Wall avoidance:**
    - Sensor registers an object ahead means turn
    - Sensor no registering an object means go forward
Example of when

Using ifs

onevent prox
  if state == FORWARD and prox.ground.delta[0] < 500 then
    state = LEFT
    motor.left.target = 0
    motor.right.target = 200
  elseif state == RIGHT and prox.ground.delta[0] < 500 then
    state = LEFT
    motor.left.target = 0
    motor.right.target = 200
  elseif state == FORWARD and prox.ground.delta[0] >= 500 then
    state = RIGHT
    motor.left.target = 200
    motor.right.target = 0
  elseif state == LEFT and prox.ground.delta[0] >= 500 then
    state = RIGHT
    motor.left.target = 200
    motor.right.target = 0
  end
Example of when

Using whens

onevent prox
  if state != STOPPED then
    when prox.ground.delta[0] < 500 do
      state = LEFT
      motor.left.target = 0
      motor.right.target = 200
    end
    when prox.ground.delta[0] >= 500 do
      state = RIGHT
      motor.left.target = 200
      motor.right.target = 0
    end
  end