Control of Time-Varying Behavior

Proportional-Derivative (PD) controller: react to the immediate sensory inputs

• E.g.: yaw control
• Need a reference (or “desired”) heading

Where does this reference come from?
Control of Time-Varying Behavior

Where does the reference come from?

- Determined by what our task is (or subtask)

- E.g.: at the current state of a mission, it may be appropriate to orient the craft in a particular direction so that it can fly back “home”
Control of Time-Varying Behavior

Can often express a “mission” in terms of a sequence of sub-tasks (or a plan)
• But: we also want to handle contingencies when they arrive

Finite state machines are a simple way of expressing such plans and contingencies
Finite State Machines (FSMs)

Pure FSM form is composed of:

• A set of states
• A set of possible inputs (or events)
• A set of possible outputs (or actions)
• A transition function:
  – Given the current state and an input: defines the output and the next state
Finite State Machines (FSMs)

States:

• Represent all possible “situations” that must be distinguished
• At any given time, the system is in exactly one of the states
• There is a finite number of these states
Finite State Machines (FSMs)

An example: a counter that increments when an input signal transitions from high to low
• States: ?
Finite State Machines (FSMs)

An example: a counter

• States: the different combinations of the digits: 000, 001, 010, … 111

• Inputs: ?
Finite State Machines (FSMs)

An example: a counter

• Inputs:
  – Really only one: the event associated with the clock transitioning from high to low
  – We will call this “C”

• Outputs: ?
Finite State Machines (FSMs)

An example: a counter

• Outputs: same as the set of states

• Transition function: ?
Finite State Machines (FSMs)

An example: a counter

• Transition function:
  – On the clock event, transition to the next highest value
FSM Example: Synchronous Counter

A Graphical Representation:

A set of states
FSM Example: Synchronous Counter

A transition

C/001

000

001

001

010

011

100

101

110

111
FSM Example: Synchronous Counter

A transition

The event

C/001

000

001

010

011

100

111

110

101
FSM Example: Synchronous Counter

A transition

000 → 001

The output

001 → 010 → 011

100

101

110

111
FSM Example: Synchronous Counter

The next transition
FSM Example: Synchronous Counter

The next transition

000 -> 001 (C/001)
001 -> 010 (C/010)
010 -> 011 (C/011)
011 -> 100
100 -> 101
101 -> 110
110 -> 111
111
FSM Example: Synchronous Counter

The full transition set

```
000 → 001 (C/001)
001 → 010 (C/010)
010 → 011 (C/011)
011 → 111 (C/111)
111 → 110 (C/110)
110 → 101 (C/110)
101 → 100 (C/100)
100 → 000 (C/000)
```
FSM Example: Synchronous Counter

Initial condition

- From 000, on x/000, go to 000
- From 001, on C/001, go to 001
- From 001, on C/101, go to 101
- From 101, on C/110, go to 110
- From 110, on C/111, go to 111
- From 111, on C/011, go to 011
- From 011, on C/010, go to 010
- From 010, on C/010, go to 000
Example II: An Up/Down Counter

Suppose we have two events (instead of one): Count up and count down

• How does this change our state transition diagram?
Example II: An Up/Down Counter

From state 000, there are now two possible transitions

- From state 000, an Up transition leads to state 001
- From state 000, a Down transition leads to state 111
Example II: An Up/Down Counter

Likewise for state 001…

000 → 001 → 010 → 011 → 100 → 101 → 110 → 111 → 000

U/001 → D/000 → U/010

D/111 → U/010
Example II: An Up/Down Counter

The full transition set
FSMs and Control

How do we relate FSMs to Control?

• States are?
FSMs and Control

How do we relate FSMs to Control?
• States are our memory of recent inputs

• Inputs are ?
FSMs and Control

How do we relate FSMs to Control?

• States are our memory of recent inputs

• Inputs are some processed representation of what the sensors are observing

• Outputs are?
FSMs and Control

How do we relate FSMs to Control?

• States are our memory of recent inputs

• Inputs are some processed representation of what the sensors are observing

• Outputs are the control actions
  – These are typically “high level” actions: e.g., set the goal orientation to 125 degrees
FSMs: A Control Example

Suppose we have a vending machine:

• Accepts dimes and nickels
• Will dispense one of two things once $.20 has been entered: Jolt or Buzz Water
  – The “user” requests one of these by pressing a button
• Ignores select if < $.20 has been entered
• Immediately returns any coins above $.20
Vending Machine FSM

What are the states?
Vending Machine FSM

What are the states?

• $0
• $.05
• $.10
• $.15
• $.20
Vending Machine FSM

What are the inputs/events?
Vending Machine FSM

What are the inputs/events?
• Input nickel (N)
• Input dime (D)
• Select Jolt (J)
• Select Buzz Water (BW)
Vending Machine FSM

What are the outputs?
Vending Machine FSM

What are the outputs?
- Return nickel (RN)
- Return dime (RD)
- Dispense Jolt (DJ)
- Dispense Buzz Water (DBW)
- Nothing (Z)
Vending Machine Design

What is the initial state?
Vending Machine Design

What is the initial state?
• $S = 0$
Vending Machine Design

What can happen from $S = \$0$?

<table>
<thead>
<tr>
<th>Event</th>
<th>Next State</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Vending Machine Design

What can happen from $S = 0$?

<table>
<thead>
<tr>
<th>Event</th>
<th>Next State</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>$.05</td>
<td>Z</td>
</tr>
<tr>
<td>D</td>
<td>$.10</td>
<td>Z</td>
</tr>
<tr>
<td>J</td>
<td>$0</td>
<td>Z</td>
</tr>
<tr>
<td>BW</td>
<td>$0</td>
<td>Z</td>
</tr>
</tbody>
</table>

What does this part of the diagram look like?
Vending Machine Design

A piece of the state diagram:
Vending Machine Design

What can happen from $S = $0.05?

<table>
<thead>
<tr>
<th>Event</th>
<th>Next State</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Vending Machine Design

What can happen from S = $0.05?

<table>
<thead>
<tr>
<th>Event</th>
<th>Next State</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>$.10</td>
<td>Z</td>
</tr>
<tr>
<td>D</td>
<td>$.15</td>
<td>Z</td>
</tr>
<tr>
<td>J</td>
<td>$.05</td>
<td>Z</td>
</tr>
<tr>
<td>BW</td>
<td>$.05</td>
<td>Z</td>
</tr>
</tbody>
</table>
Vending Machine Design

A piece of the state diagram:
Vending Machine Design

What can happen from $S = $0.10?

<table>
<thead>
<tr>
<th>Event</th>
<th>Next State</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Vending Machine Design

What can happen from $S = $0.10? 

<table>
<thead>
<tr>
<th>Event</th>
<th>Next State</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>$.15</td>
<td>Z</td>
</tr>
<tr>
<td>D</td>
<td>$.20</td>
<td>Z</td>
</tr>
<tr>
<td>J</td>
<td>$.10</td>
<td>Z</td>
</tr>
<tr>
<td>BW</td>
<td>$.10</td>
<td>Z</td>
</tr>
</tbody>
</table>
Vending Machine Design

A piece of the state diagram:
Vending Machine Design

What can happen from $S = 0.15$?

<table>
<thead>
<tr>
<th>Event</th>
<th>Next State</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Vending Machine Design

What can happen from S = $0.15?

<table>
<thead>
<tr>
<th>Event</th>
<th>Next State</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>$.20</td>
<td>Z</td>
</tr>
<tr>
<td>D</td>
<td>$.20</td>
<td>RN</td>
</tr>
<tr>
<td>J</td>
<td>$.15</td>
<td>Z</td>
</tr>
<tr>
<td>BW</td>
<td>$.15</td>
<td>Z</td>
</tr>
</tbody>
</table>
Vending Machine Design

A piece of the state diagram:

- From $0 to $.05: x/Z, J/Z, BW/Z, N/Z, D/Z
- From $.05 to $.10: N/Z
- From $.10 to $.15: D/Z
- From $.15 to $.20: J/Z, BW/Z
- From $.20 to $0: D/Z

Andrew H. Fagg: Embedded Real-Time Systems: FSMs
Vending Machine Design

Finally: what can happen from $S = 0.20$?

<table>
<thead>
<tr>
<th>Event</th>
<th>Next State</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Vending Machine Design

Finally, what can happen from $S = 0.20$?

<table>
<thead>
<tr>
<th>Event</th>
<th>Next State</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>$0.20$</td>
<td>RN</td>
</tr>
<tr>
<td>D</td>
<td>$0.20$</td>
<td>RD</td>
</tr>
<tr>
<td>J</td>
<td>$0$</td>
<td>DJ</td>
</tr>
<tr>
<td>BW</td>
<td>$0$</td>
<td>DBW</td>
</tr>
</tbody>
</table>
Vending Machine Design

The complete state diagram:

[State diagram with states and transitions labeled with amounts and actions like 'J/Z', 'BW/Z', 'N/Z', 'D/Z', 'J/ZBW/Z', etc.]

Andrew H. Fagg: Embedded Real-Time Systems: FSMs
Last Time

Andrew H. Fagg: Embedded Real-Time Systems: FSMs

56
FSMs and Control

How do we relate FSMs to Control?

• States are?
FSMs and Control

How do we relate FSMs to Control?

• States are our memory of recent inputs

• Inputs are?
FSMs and Control

How do we relate FSMs to Control?

• States are our memory of recent inputs

• Inputs are some processed representation of what the sensors are observing

• Outputs are?
FSMs and Control

How do we relate FSMs to Control?

• States are our memory of recent inputs

• Inputs are some processed representation of what the sensors are observing

• Outputs are the control actions
A Robot Control Example

Consider the following task:

- The robot is to move toward the first beacon that it “sees”
- The robot searches for a beacon in the following order: right, left, front

What is the FSM representation?
Robot Control Example II

Consider the following task:
• The robot must lift off to some altitude
• Translate to some location
• Take pictures
• Return to base
• Land
• At any time: a detected failure should cause the craft to land

What is the FSM representation?
FSMs As Controllers

Must bridge the gap between the FSM and the low- and mid-level controllers

• Events:
  – Abstraction of sensor or internal state

• Actions:
  – Modify mid- or low-level control behavior
FSMs in C

State state = STATE_0;  // Initial state
while(1) {
    <do some processing of the sensory inputs>
    switch(state) {
        case STATE_0:
            <handle state 0>
            break;
        case STATE_1:
            <handle state 1>
            break;
        case STATE_2: ...
    }
}
FSMs in C (some other possibilities)

State state = STATE_0; // Initial state
while(1) {
    <do some processing of the sensory inputs>
    switch(state) {
        case STATE_0:
            <handle state 0>
            break;
        :
        default:
            <handle default case>
            break;
    }
    <do some low-level control>
}
Handling Each State

• You will need to provide code that handles the event processing for each state

• Specifically:
  – You need to handle each event that can occur
  – For each event, you must specify:
    • What action is to be taken
    • What the next state is
Handling Each State

In our vending machine example:

• Events are easy to describe (only a few things can happen)
• It is convenient in this case to also “switch” on the event
FSMs in C: Processing for Individual States

case STATE_10cents:  
    // $.10 has already been deposited  
    switch(event) {  
        case EVENT_NICKEL:  // Nickel  
            state = STATE_15cents;  // Transition to $.15  
            break;  
        case EVENT_DIME:  // Dime  
            state = STATE_20cents;  // Transition to $.2  
            break;  
        case EVENT_JOLT:  // Select Jolt  
        case EVENT_BUZZ:  // Select Buzzwater  
            display_NOT_ENOUGH();  
            break;  
        case EVENT_NONE:  // No event  
            break;  // Do nothing  
    }  
    break;
FSMs As Controllers

Must bridge the gap between the FSM and the low- and mid-level controllers

• Events:
  – Abstraction of sensor or internal state

• Actions:
  – Modify mid- or low-level control behavior
Handling Each State

Some events do not fall neatly into one of several categories

• This precludes the use of the “switch” construct for events

• For example: an event that occurs when our hovercraft reaches a goal orientation

• For these continuous situations, we typically use an “if” construct …
FSMs in C: Processing for Individual States

case STATEMISSION_PHASE_3:
  if (heading_error < 100 &&
      heading_error > -100)
  {
    // Accelerate forward!
    forward_thrust = 126;
    state = STATEMISSION_PHASE_4;
  };
  break;
FSMs in C: Processing for Individual States

```c
: case STATE_MISSION_PHASE_4:
    if(distance_left < 200 ||
       distance_right < 200)
    {
        // Brake!
        forward_thrust = 0;
        middle_thrust_magnitude(300);
        middle_thrust_dir(BRAKE);
        state = STATE_MISSION_PHASE_5;
        counter = 0;     // Reset the clock
    }
    break;
: 
```

Andrew H. Fagg: Embedded Real-Time Systems: FSMs
FSMs in C: Processing for Individual States

```c
: case STATE_MISSION_PHASE_5:
    if (counter > 20) {
        // One second has gone by since we
        // started the brake: Stop the brake

        middle_thrust_magnitude(100);
        middle_thrust_dir(HOVER);
        forward_thrust = 100;
        heading_goal = 2700;
        state = STATE_MISSION_PHASE_6;
    }
    break;
:
```

NOTE: counter is being incremented once per control cycle (outside of the FSM code)
FSM Implementation Notes

- FSM code should not contain delays or waits
  - No delay_ms() or while(…){}
  - Remember that your FSM code will be called once per control cycle: use “if” to check for an event during that control cycle
- Use LEDs and/or fprintf() to indicate current state
- Implement and test incrementally