UNIT 4b: KEY CONCEPTS

Finite State Machine:
- Boolean Inputs
- Boolean Output
- Can only be in one state at a time

Two Formulations:
- Moore (which we use in this course)
- Mealy

Design Steps
- Create abstract representation
- Minimize states (optional)
- Perform state assignment (if using gate logic)
- Select flip-flop for implementation (use D flip-flop when possible)
- Generate next-state logic (if using gate logic)
- Simulate and implement the state machine
UNIT 4:
SEQUENTIAL LOGIC

PART C:
FINITE STATE MODELING
DEVELOP A CONTROL SYSTEM FOR THE iROBOT 510 PACKBOT

- The iRobot 510 Packbot is designed for bomb disposal, surveillance and reconnaissance, CBRN detection and HazMat handling operations.
- Your focus is on navigating the Packbot to an object for interaction.
THE PACKBOT BASE HAS THREE SENSORS FOR OBJECT DETECTION

- All three sensors are mounted to the robot base
- **LS** is active if an object is detected to the robot’s left side
- **RS** is active if an object is detected to the robot’s right side
- **CLOSE** is active if an object is detected within 2 inches of the robot’s front side
THE PACKBOT MOTOR CONTROLLER HAS THREE INPUTS

- **FORWARD** should be activated when the robot is to move forward and deactivated when the robot is to stop.

- **LEFT** should be activate to turn the robot to the left.

- **RIGHT** should be activated to turn the robot to the right.
CERTAIN OPERATING PRINCIPLES SHOULD BE FOLLOWED

- When **LS** is active and **RS** is not, then the robot should turn left.
- When **RS** is active and **LS** is not, then the robot should turn right.
- When both **LS** and **RS** are active, then the robot should go straight.
- When **CLOSE** is active, the robot should stop.
- The robot should start from stop.
- When none of the **LS**, **RS**, and **CLOSE** inputs are active, the robot should go forward and turn toward right to circle around.
IN ADDITION, CERTAIN CONSTRAINTS ARE IN EFFECT

- LEFT and RIGHT cannot be activated unless FORWARD is also activated.
- To stop the robot, FORWARD, LEFT, and RIGHT should be deactivated.
- To move the robot forward in the current direction, both LEFT and RIGHT need to be deactivated, and FORWARD needs to be activated.
LABEL AND IDENTIFY MEANING FOR INPUTS AND OUTPUTS

- **Inputs (LS, RS, CL)**
  - LS = 0  LS deactivated (off) (nothing is sensed to the left)
  - LS = 1  LS activated (on) (something is sensed to the left)
  - RS = 0  RS deactivated (off) (nothing is sensed to the right)
  - RS = 1  RS activated (on) (something is sensed to the right)
  - CL = 0  CL deactivated (off) (nothing in front of robot)
  - CL = 1  RS activated (on) (object detected in front of robot)

- **Outputs (L, R, F)**
  - L = 0  L is low, vehicle does not turn left
  - L = 1  L is activated, vehicle turns left
  - R = 0  R is low, vehicle does not turn right
  - R = 1  R is activated, vehicle turns right
  - F = 0  F is low, vehicle stops
  - F = 1  F is activated, vehicle moves forward in current direction
AUTONOMOUS OBJECT SEEKING

Sketch the state transition diagram for the FSM.
AUTONOMOUS OBJECT SEEKING

Sketch the state transition diagram for the FSM.

- **S0 (STOP)**
  - 000

- **S1 (LT)**
  - 101

- **S2 (RT)**
  - 011

- **S3 (FOR)**
  - 001

Inputs:
- LS
- RS
- CL

Sx (xxx)
- L
- R
- F
Sketch the state transition diagram for the FSM.

By convention, the initial (starting) state is typically named S0. If this is not the case, you should indicate the initial state for your FSM.
AUTONOMOUS OBJECT SEEKING

Sketch the state transition diagram for the FSM.

Object detected to left of robot!
AUTONOMOUS OBJECT SEEKING

Sketch the state transition diagram for the FSM.

Object detected to right of robot!
AUTONOMOUS OBJECT SEEKING

Sketch the state transition diagram for the FSM.

Object still present to left of robot
AUTONOMOUS OBJECT SEEKING

Sketch the state transition diagram for the FSM.

S0 (STOP)
0 0 0

S1 (LT)
1 0 1

S2 (RT)
0 1 1

S3 (FOR)
0 0 1

Inputs:
LS RS CL
L R F

Object directly ahead of robot!
Sketch the state transition diagram for the FSM.
AUTONOMOUS OBJECT SEEKING

Sketch the state transition diagram for the FSM.

- **Inputs:**
  - LS
  - RS
  - CL
  - L
  - R
  - F

- **States:**
  - **S0 (STOP)**
    - 0 0 0
  - **S1 (LT)**
    - 1 0 1
    - Transition on 100
    - Transition on xx1
  - **S2 (RT)**
    - 0 1 1
    - Transition on 010
  - **S3 (FOR)**
    - 0 0 1
    - Transition on 010
    - Transition on 110

- **Object detected on both sides of robot:**
  - Transition on 100
Sketch the state transition diagram for the FSM.

- **S0 (STOP)**
  - Input: 000

- **S1 (LT)**
  - Input: 100

- **S2 (RT)**
  - Input: 010

- **S3 (FOR)**
  - Input: 001

- **Inputs:**
  - LS
  - RS
  - CL
  - L
  - R
  - F

Object directly ahead of robot!
Sketch the state transition diagram for the FSM.

Object still present to right of robot

Inputs:
LS RS CL
L R F

Sx (xxx)
Sketch the state transition diagram for the FSM.

The diagram shows the transition between states S0 (STOP), S1 (LT), S2 (RT), and S3 (FOR) based on the inputs LS, RS, and CL. The states are connected by transitions labeled with input combinations and state changes.

- **Inputs:**
  - LS
  - RS
  - CL

- **States:**
  - S0 (STOP)
  - S1 (LT)
  - S2 (RT)
  - S3 (FOR)

- **Transitions:**
  - From S0 to S1 with input 100.
  - From S0 to S2 with input 010.
  - From S1 to S0 with input 100.
  - From S1 to S3 with input 110.
  - From S2 to S1 with input 010.
  - From S2 to S3 with input 010.
  - From S3 to S2 with input 010.

- **Notes:**
  - Object now on left side of robot.
Sketch the state transition diagram for the FSM.
AUTONOMOUS OBJECT SEEKING

Sketch the state transition diagram for the FSM.

Object detected on both sides of robot

Inputs:
LS RS CL
L R F

S0 (STOP)
0 0 0

S1 (LT)
1 0 1

S2 (RT)
0 1 1

S3 (FOR)
0 0 1
AUTONOMOUS OBJECT SEEKING

Sketch the state transition diagram for the FSM.
AUTONOMOUS OBJECT SEEKING

Sketch the state transition diagram for the FSM.

Object directly ahead of robot!
Sketch the state transition diagram for the FSM.

Default movement is to move forward...
AUTONOMOUS OBJECT SEEKING

Sketch the state transition diagram for the FSM.

![State Transition Diagram](image)

... and make right hand turns
AUTONOMOUS OBJECT SEEKING

Sketch the state transition diagram for the FSM.

... until one of the sensors detects an object.
AUTONOMOUS OBJECT SEEKING

We can merge two inputs for S2…
AUTONOMOUS OBJECT SEEKING

Sketch the state transition diagram for the FSM.

Are we done?
AUTONOMOUS OBJECT SEEKING

Sketch the state transition diagram for the FSM.

Are all the input combinations handled?
AUTONOMOUS OBJECT SEEKING

Sketch the state transition diagram for the FSM.
Sketch the state transition diagram for the FSM.
Sketch the state transition diagram for the FSM.
Sketch the state transition diagram for the FSM.
Sketch the state transition diagram for the FSM.
Sketch the state transition diagram for the FSM.
AUTONOMOUS OBJECT SEEKING

Sketch the state transition diagram for the FSM.

Inputs:
LS  RS  CL
L  R  F

Sx (xxx)
AUTONOMOUS OBJECT SEEKING

Sketch the state transition diagram for the FSM.
Sketch the state transition diagram for the FSM.

**Inputs:**
- LS
- RS
- CL
- L
- R
- F

**States:**
- S0 (STOP)
- S1 (LT)
- S2 (RT)
- S3 (FOR)
Sketch the state transition diagram for the FSM.
AUTONOMOUS OBJECT SEEKING

Sketch the state transition diagram for the FSM.
Sketch the state transition diagram for the FSM.
FINALLY, IMPLEMENT THE DESIGNED FSM ON ARDUINO

- Simulate operation using LEDs and switches
- Test operation with actual system
SIMPLE POSITION CONTROL

You are asked to design a controller (finite state machine) to control the motion of a robot arm. The machine has two inputs (P, N) and three outputs (Forward, Reverse, Slow).
SIMPLE POSITION CONTROL

The inputs \((P, N)\) are described as follows:

- When the robot position is less than the desired position; \(P=1, N=0\).
- When the robot position is larger than the desired position; \(P=0, N=1\).
- When the robot position is identical to the desired position; \(P=0, N=0\).
- When the robot has moved at least \(7/8\) of the way to the desired position, moving either forward or backward; \(P=1, N=1\).
SIMPLE POSITION CONTROL
SIMPLE POSITION CONTROL
SIMPLE POSITION CONTROL
SIMPLE POSITION CONTROL

Start

Target

-.7/8 mark

N

P
SIMPLE POSITION CONTROL

Target

Start

N
P
SIMPLE POSITION CONTROL

Start

Target

N

P
SIMPLE POSITION CONTROL
SIMPLE POSITION CONTROL

Start

7/8 mark

Target

N

P

Jeff Shelton – 12 February 2015
SIMPLE POSITION CONTROL

Start

Target

7/8 mark

N

P
SIMPLE POSITION CONTROL
SIMPLE POSITION CONTROL

The outputs are:
- **FORWARD (F)**: Robot moves forward when this output is asserted.
- **REVERSE (R)**: Robot moves backward when this output is asserted.
- **SLOW (S)**: Brake is applied to slow down the robot.
SIMPLE POSITION CONTROL

The operation of the machine is:

- The system starts at a stop state where all outputs are zero.
- If the current position of the robot is less than the desired position, \text{FORWARD} should be asserted (F = 1).
- If the current position of the robot is larger than the desired position, \text{REVERSE} should be asserted (R = 1).
- If the position is correct, stop the robot by setting all outputs to zero.
- If the robot has moved 7/8 the way to the target position, the \text{SLOW} output should be asserted (S = 1) to brake the motion of the arm. \text{FORWARD} or \text{REVERSE} signals must remain high while \text{SLOW} is asserted for movement to continue.
Sketch the state transition diagram for the FSM.
Sketch the state transition diagram for the FSM.

Exactly on target...
Sketch the state transition diagram for the FSM.
Sketch the state transition diagram for the FSM.
Sketch the state transition diagram for the FSM.

We shouldn't get a 7/8ths signal when we are stopped.

Treat as either a "don't care," condition or as an error...
maybe create an emergency shutdown state?
Sketch the state transition diagram for the FSM.

- **S0 (STOP)**
  - 000
  - Transition on 00 or 11
  - 11: "X"
- **S1 (FORWARD)**
  - 100
  - Transition on 10
- **S2 (REVERSE)**
  - 010
  - Transition on 01
- **S3 (SLOW FOR)**
  - 101
  - Transition on 10
- **S4 (SLOW REV)**
  - 011
  - Transition on 01

Inputs:
- P N
- F R S
Sketch the state transition diagram for the FSM.

Exactly on target…
Sketch the state transition diagram for the FSM.

Again, either a "don't care," or an error...

We're not able to jump to the other side of the target without passing through the 7/8ths zone
Sketch the state transition diagram for the FSM.

The diagram shows the different states and transitions with inputs. The states include S0 (STOP), S1 (FORWARD), S2 (REVERSE), S3 (SLOW FOR), and S4 (SLOW REV). The transitions are represented by arrows indicating the input combinations that lead from one state to another. The input combinations are indicated at the transitions, such as 00, 01, 10, and 11. The diagram also indicates when the system is not yet to target, marked as "X".
Sketch the state transition diagram for the FSM.

```
Inputs:
P N
F R S
```

```
S0 (STOP)
1 0 0
0 0 0
11

S1 (FORWARD)
1 0 0
10 0 0
10

S2 (REVERSE)
0 1 0
01 0 1

S3 (SLOW FOR)
1 0 1
11 01 00

S4 (SLOW REV)
1 0 1
01 01

7/8ths of way to target

"X"
```

Jeff Shelton – 12 February 2015
Sketch the state transition diagram for the FSM.
Sketch the state transition diagram for the FSM.

"X"

Exactly on target...
Sketch the state transition diagram for the FSM.

Another "don't care," or an error…

Shouldn't be able to jump to other side of the target without passing through target.
Sketch the state transition diagram for the FSM.

"X" shouldn't get a "P" signal when already in 7/8ths zone.
Sketch the state transition diagram for the FSM.
Sketch the state transition diagram for the FSM.
Sketch the state transition diagram for the FSM.
Sketch the state transition diagram for the FSM.

Approaching target from negative side
Sketch the state transition diagram for the FSM.

- **S0 (STOP)**
  - Transition: 00
  - Output: 000

- **S1 (FORWARD)**
  - Transition: 10
  - Input: 100

- **S2 (REVERSE)**
  - Transition: 00
  - Input: 010

- **S3 (SLOW FOR)**
  - Transition: 11
  - Input: 101

- **S4 (SLOW REV)**
  - Transition: 11
  - Input: 011

Another "don't care" or error state

Can't jump to opposite side of 7/8ths zone
Sketch the state transition diagram for the FSM.
Sketch the state transition diagram for the FSM.
Sketch the state transition diagram for the FSM.
Sketch the state transition diagram for the FSM.
Sketch the state transition diagram for the FSM.

Can't jump to other side of target

"Don't care" or error
Sketch the state transition diagram for the FSM.

Still in 7/8ths zone
Sketch the state transition diagram for the FSM.
# Simple Position Control

## Truth Table:

<table>
<thead>
<tr>
<th>Input</th>
<th>Current State</th>
<th>Next State</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P</td>
<td>N</td>
<td>F</td>
</tr>
<tr>
<td>0 0</td>
<td>S0</td>
<td>S0</td>
<td>0</td>
</tr>
<tr>
<td>0 0</td>
<td>S1</td>
<td>S0</td>
<td>1</td>
</tr>
<tr>
<td>0 0</td>
<td>S2</td>
<td>S0</td>
<td>0</td>
</tr>
<tr>
<td>0 0</td>
<td>S3</td>
<td>S0</td>
<td>1</td>
</tr>
<tr>
<td>0 0</td>
<td>S4</td>
<td>S0</td>
<td>0</td>
</tr>
<tr>
<td>0 1</td>
<td>S0</td>
<td>S2</td>
<td>0</td>
</tr>
<tr>
<td>0 1</td>
<td>S1</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>0 1</td>
<td>S2</td>
<td>S2</td>
<td>0</td>
</tr>
<tr>
<td>0 1</td>
<td>S3</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>0 1</td>
<td>S4</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>1 0</td>
<td>S0</td>
<td>S1</td>
<td>0</td>
</tr>
<tr>
<td>1 0</td>
<td>S1</td>
<td>S1</td>
<td>1</td>
</tr>
<tr>
<td>1 0</td>
<td>S2</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>1 0</td>
<td>S3</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>1 0</td>
<td>S4</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>1 1</td>
<td>S0</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>1 1</td>
<td>S1</td>
<td>S3</td>
<td>1</td>
</tr>
<tr>
<td>1 1</td>
<td>S2</td>
<td>S4</td>
<td>0</td>
</tr>
<tr>
<td>1 1</td>
<td>S3</td>
<td>S3</td>
<td>1</td>
</tr>
<tr>
<td>1 1</td>
<td>S4</td>
<td>S4</td>
<td>0</td>
</tr>
</tbody>
</table>
# Simple Position Control

## Truth Table:

<table>
<thead>
<tr>
<th>Input</th>
<th>Current State</th>
<th>Next State</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Q0</td>
<td>Q1</td>
<td>Q2</td>
</tr>
<tr>
<td>P</td>
<td>N</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
SIMPLE POSITION CONTROL

Next State Logic:

Output Logic:
# Simple Position Control

**Next State Logic:**

<table>
<thead>
<tr>
<th>(Q0^*) ((P = 0))</th>
<th>(Q1)</th>
<th>(Q2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>01</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>10</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(N)</th>
<th>(Q0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>X</td>
</tr>
<tr>
<td>01</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>X</td>
</tr>
<tr>
<td>10</td>
<td>X</td>
</tr>
</tbody>
</table>

**Output Logic:**

<table>
<thead>
<tr>
<th>(Q1^*) ((P = 0))</th>
<th>(Q0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>X</td>
</tr>
<tr>
<td>01</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>X</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(N)</th>
<th>(Q0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>X</td>
</tr>
<tr>
<td>01</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(Q2^*) ((P = 0))</th>
<th>(Q0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>X</td>
</tr>
<tr>
<td>01</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>X</td>
</tr>
<tr>
<td>10</td>
<td>X</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(N)</th>
<th>(Q0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>X</td>
</tr>
<tr>
<td>01</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>X</td>
</tr>
<tr>
<td>10</td>
<td>X</td>
</tr>
</tbody>
</table>
**SIMPLE POSITION CONTROL**

Next State Logic:

\[ Q0^* = P \cdot \overline{Q1} \]

Output Logic:
**SIMPLE POSITION CONTROL**

Next State Logic:

\[ Q_0^* = P \cdot \overline{Q_1} \]
\[ Q_1^* = N \cdot \overline{Q_0} \]

Output Logic:
SIMPLE POSITION CONTROL

Next State Logic:

\[
Q_0^* = P \cdot \overline{Q_1}
\]
\[
Q_1^* = N \cdot \overline{Q_0}
\]
\[
Q_2^* = P \cdot N
\]

Output Logic:
SIMPLE POSITION CONTROL

Next State Logic:

\[ Q_0^* = P \cdot \overline{Q_1} \]
\[ Q_1^* = N \cdot \overline{Q_0} \]
\[ Q_2^* = P \cdot N \]

Output Logic:

\[ F = Q_0 \]
\[ R = Q_1 \]
\[ S = Q_2 \]
COMING UP...

Computer Systems (Sequential Logic)
  - Finite state reduction