EE 109 Unit 15 – State Machines
What is state?

• You see delicious looking potato salad in the cafeteria? Do you eat it?
  – Ate potato salad as a kid..._________
  – As a teen, ate potato salad at a picnic after a long hot day...________
• Your decision is influenced not just on how it looks now, but the sum of all your ___________________________
  – The sum of all your previous experiences is what is known as __________
  – Your 'state' determines your __________________ of your senses and thoughts
• In a circuit, 'state' refers to all the __________ being remembered (__________________ or memory)
• In software, 'state' refers to all the __________________ values that are being used
State Machine Block Diagram

- A system that utilizes state is often referred to as a state machine
  - A.k.a. ________________________________

- Most state machines can be embodied in the following form
  - Logic examines what's happening NOW (inputs) & in the PAST (state) to...
    - Produce outputs (actions you do now)
    - Update the state (which will be used in the future to change the decision)

- Inputs will go away or change, so state needs to summarize/capture anything that might need to be _______________ and used in the future

```
Inputs
(A DC, Timer, Buttons)
```

```
Logic
```

```
State
(memory)
```

```
Outputs
```
State Diagrams

- Abstractly a state machine can be visualized and represented as a flow chart (or state diagram)
  - Circles or boxes represent _____________
  - Arrows show what input causes a ______________
  - Outputs can be generated whenever you reach a particular state or based on the combination of state + input

State Machine to check for two consecutive 1's on a digital input

- On startup

```
Input=_____  Input=_____  Input=_____  Input=___  

S0  Out=False  S1  Out=False  S2  out=True

Input=___  Input=___  Input=___  Input=___

On startup
```
Another Example

- Potato salad eating...Let's let state represent our present feeling/experience
- Notice I don't remember __________ occasion that I ate potato salad...my state summarizes ____________________ for me to make a decision
  - Sometimes you have two states with same output but it helps differentiate future action (I have to get sick twice) to not eat (if I just had a positive and negative state I would change my mind the 1st time I got sick)
Formal Definition

• Mathematically, a state machine is defined by a 6-tuple (a tuple is just several pieces of information that go together):
  – A set of possible _______________
  – A set of possible _______________
  – A set of possible _______________
  – An _____________________
  – A transition function: {_________ x __________} -> ______________
  – An output function: {States x Inputs} -> ______________________
Formal Definition

Mathematically, a state machine consists of:

- A set of possible input values: \{ ____________ \}
- A set of possible states: \{ ______________ \}
- A set of possible outputs: \{ ______________ \}
- An initial state = __________
- A transition function:
  - \{States x Inputs\} -> the Next state
- An output function:
  - \{States x Inputs\} -> Output value(s)
Washing Machine State Diagram

We move through the states based on the conditions. Outputs get asserted when the machine is in that state and the transition is true.

Stay in the initial state until there is enough money (coins) and the door is closed.
Washing Machine State Diagram

Move to the Fill state when there is enough money (coins) and the door is closed.
Washing Machine State Diagram

1. Idle
   \[ N = 2 \]
2. Fill
   \[ WV = 1 \]
3. Agitate
   \[ Motor = 1 \]
4. Drain
   \[ DV = 1 \]
5. EMPTY
   \[ N = 0 \]
6. FULL

Stay in the Fill state until it is full...also set the Water Valve Open output to be true.
Washing Machine State Diagram

Move to the Agitate state after it is full
Software vs. Hardware

• Software
  – State = just a variable(s)
  – Logic = if statements to update the next state
    • if(state == 'A' && input == 1)
      { state = 'B'; }
  – Transitions triggered by input or timers
  – In EE 109, we'll implement our state machines in SW

• Hardware
  – State = Register (D-Flip-Flops)
  – Logic = AND/OR Gates to produce the next state & outputs
  – Transitions triggered by clock signal
  – More on this in EE 154
Software Implementation

- Store 'state' in some variable and assign ______________ to represent state (0=Idle, 1=Fill, etc.)
- Use a timer or just __________ certain inputs and then make appropriate transitions

```cpp
int main()
{
    bool coins, door;
    int state = 0, n = 0;
    while(1)
    {
        _delay_ms(10);
        coins = PIND & (1 << PD0);
        door = PIND & (1 << PD1);
        if(state == 0){
            if( coins && door ){
                state = 1;
            }
        }
        else if(state == 1){
            ...
        }
        else if(state == 1){
            ...
        }
        return 0;
    }
}
```
More Implementation Tips

- Continuously loop
- Each iteration:
  - Poll inputs
  - Use ________________ to decide current state
  - In each state, update state appropriately based on desired transitions from that state
  - Produce appropriate output from that state

```c
// input = PD0, output = PD7
int main()
{
    // be sure to init. state
    unsigned char state = 0;
    unsigned char input, output;
    while(1)
    {
        _delay_ms(10);
        input = PIND & (1 << PD0);
        if(state == 0){
            PORTD &= ~(1 << PD7);
            if( input ){ state = 1; }
        }
        else if(state == 1){
            PORTD &= ~(1 << PD7);
            if( input ){ state = 2; }
            else { state = 0; }
        }
        else {
            PORTD |= (1 << PD7);
            if( !input ) { state = 0; }
        }
    }
    return 0;
}
```
MORE EXAMPLES

HW (Instruction Cycle) & Software (String Matching)
More State Machines

- State machines are all over the place in digital systems
- Instruction Cycle of a computer processor

![Diagram](image-url)
Another Example

• On the Internet, packets of data are transferred between “router” devices

• Each router receives thousands of packet per second each of 100’s-1000’s of bytes of data

• These packets may contain viruses, spam, etc.

• Given patterns (common spam words or virus definitions), can we find these in the data and filter them out?

```
1110 0010 0101 1001
0110 1011 0000 1100
0100 1101 0111 1111
1010 1100 0010 1011
0001 0110 0011 1000
```
Looking for Signatures

• Look for specific patterns (i.e. signatures) such as data that would indicate a specific virus, words that are typically spam, etc.

• Databases of these signatures are available

• We take a packet and search for the presence of any of these signatures in our database

• If we find a signature we can drop the packet and not deliver it
String/Pattern Matching

• Given a large array of data (let's say text characters) how can we efficiently find the occurrence of specific strings (patterns)?

Hello,

I am Barr. Phillip Butulezi, an attorney of law to a deceased Immigrant property Magnate, who was based in the U.K, also referred to as my client.

On the 25th of July 2000, my client, his wife, and their two Children died in the Air France concord plane crash bound for New York. They were on their way to a world cruise.
Brute Force

- Take each character in the data stream
  - Compare each string in the _____________ to the string _____________ at the character in the data stream
  - Use `strn cmp()` or `strcmp()`

Data Stream = N chars with T Targets => Run Time proportional to _____________
A Better way

- Can we avoid checking each of the T target strings for each character in the data stream
- Can we take a letter from the data stream and simultaneously track __________________________ target string matches
  - Example strings: her, hers, here, rest
  - Data Stream: heresthers
    - Don’t check all 4 target strings, just grab ‘h’ and see what options are possible and which are ruled out... (i.e. keep track of all options simultaneously)
    - h [could be her or hers or here]
    - e [could still be her or hers or here]
    - r [found her! But could also be hers or here or start of rest]
    - e [found here! Could be start of rest]
    - s [Could be rest ]
    - t [Found rest ]
    - h [Could be start of her or hers or here]
Use a state machine

- '!' represents 'null' state
  - No part of a definition found
- Slightly different notation used
  - State label indicates the input character that would put you into that state
- What state you’re in "tracks" what you’ve ______ thus far AND what target strings you ______ be about to find…
Finite State Automaton

• Data Stream: heresthers

Run-Time proportional to $N$
USING STATE MACHINES TO SIMPLIFY & ORGANIZE DESIGNS
An Example

- Let's design a stopwatch (_______ units)
- What are the inputs and outputs
  - Inputs
    - Buttons for ______________
    - ______________________
  - Outputs
    - LCD [___________ time format]
- Question:
  - What do I need state for in this design?
- Answer:
  - Anytime you provide the same input and different outputs/actions occur, there is state inside
  - Different actions for same button press
Why Use State Machines

• It can be very hard/difficult to design a system where all the inputs can communicate each of the outputs (i.e. an all-to-all relationship)
Why Use State Machines

- Easier to decouple relationship between input and output
- Let inputs ________________, then examine the state to decide what ________________ should be or do
- Similar to the popular MVC GUI & Web app design approach
  - Model->View->Controller (MVC design)
  - Model (State), View (Output), Controller (Input)
Stopwatch Application

• What states do we need to differentiate button presses

• When timer interrupt occurs examine the state to decide how to update the display (or just leave current displayed time)

• What else in this design is technically "state"?
  – Time: SS.Tenths
  – Every time the timer interrupts check to see if time needs to update & increment the time if necessary
Suggested Guidelines

• Use a timer to generate an interrupt every 0.1s
• Use the timer ISR to perform _____________ and ______________
• Use ___________ in main() to detect input button presses and update state (but not necessarily display unless necessary)

```c
int main()
{
    // be sure to init. state
    unsigned char state = 0;

    // init and start timer

    // used to check inputs
    // and perform state updates
    while(1)
    {
        // Poll inputs and updates
        // state
        return 0;
    }

    // Use to perform output tasks
    ISR(TIMER1_COMPA_vect)
    {
        // update time and
        // display based on state
    }
}"
```