Motivation

```c
void UnitControl()
{
    up = down = 0; open = 1;
    while (1) {
        while (req == floor);
        open = 0;
        if (req > floor) { up = 1;}
        else {down = 1;}
        while (req != floor);
        open = 1;
        delay(10);
    }
}
```
**Motivation**

- FSM
  - Definition, notation, usage, extensions, system behaviour
- FSMD
- FSMD execution / timing
- Modeling

---

**Finite State Machine, FSM**

- Finite state machines are used to describe the behavior of a system and is one of the most fundamental models of computation.
- A finite state machine has a set of *states*, and its control moves from state to state in response to external *inputs*.
- The term "finite" refers to the fact that the set of states Q is a finite state.
**Finite State Machine, FSM**

A finite state machine is said to be *deterministic* if for a given state $q_i$ and a given input $s_i$ there is only one possible next state.

In case of two or more possible next states for the same input $s_i$, the finite state machine is said to be *non-deterministic*.

We shall only be concerned with deterministic finite state machines.

---

**FSM definition**

$FSM = < Q, \Sigma, \delta, q_0, F >$

- A finite set of states $Q$
- A finite set of input symbols, $\Sigma$
- A transition function, $\delta: Q \times \Sigma \rightarrow Q$
- A start state $q_0$ from $Q$
- A set of final or accepting states, $F \subseteq Q$
A FSM can be used to recognize a sequence of input symbols (a string)

Used to describe **regular languages** and to show properties of such languages

if two finite state machines represents the same language.

The language of a finite state machine is the set of all strings which the finite state machines accepts.

Let $q_n$ be the state after processing each input symbol in sequence,

If $q_n$ belongs to $F$, the string is said to be **accepted**, otherwise it is **rejected**.
Example

We want to specify a FSM that accepts all and only the strings of 0's and 1's that have the sequence 01 somewhere in the string.

\( \{x01y \mid x \text{ and } y \text{ are any string of 0's and 1's} \} \)

Strings in the language

- 01, 11010 and 100011

Strings not in the language

- 0 and 111000

Example

\( \{x01y \mid x \text{ and } y \text{ are any string of 0's and 1's} \} \)

FSM = \(<\{q_0, q_1, q_2\},\{0,1\},?, q_0,\{q_1\}>\)
**FSM extensions**

- We are not always content with a machine which only accepts strings!
- We also need machines which produce some output as a reaction to the inputs
- There are two distinct approaches to include output:
  - Moore: associate the emission of an output event with a state
  - Mealy: associate the emission of an output event with a transition
- Both are equal in modeling capabilities

**FSM with output**

\[
\text{FSM} = \langle Q, \Sigma, \Gamma, \Delta, q_0, F \rangle
\]

- A finite set of states \( Q \)
- A finite set of input symbols, \( \Sigma \)
- A finite set of output symbols, \( \Gamma \)
- A transition function, \( \Delta : Q \times \Sigma \rightarrow Q \)
- A start state \( q_0 \) from \( Q \)
- A set of final or accepting states, \( F \subseteq Q \)
**FSM with output**

- **Moore machine**
  - State transition: $q_0/d_0 \rightarrow s_1 \rightarrow q_2/d_2$

- **Mealy machine**
  - State transition: $s_1/d_1 \rightarrow q_0 \rightarrow q_2$

---

**Describing behavior**

- **Seat belt alarm**
  - **Specification:**
    - "Five seconds after the key is turned on, if the belt has not been fastened, an alarm will beep for ten seconds or until the key is turned off"

- **FSM?**
  - Device a FSM with your neighbour (10 min.)
Seat belt alarm

**States:**
- Off, Wait, Alarm

**Input events:**
- Key
- Belt
- End

**Output events:**
- Alarm
- Start

![FSMD Diagram]

FSMD
FSMD execution

von Neumann machine

Controller: Instruction processor
1. Determine label of next instruction
2. Instruct memory to provide instruction
3. Receive instruction and decode it
4. Inform data processor of operation to be executed
5. Determine labels of operands and pass them to data memory
6. Receive state information from data processor

Defines the instruction sequence
von Neumann machine

**Controller:** Instruction processor

Get label → Request instruction → Receive instruction

Receive state → Operands → Instruct DP

Datapath: Data processor

Receive Inst type →Operand labels →Request operands

Store result → Return state → execute

Receive operands
FSMD execution

- Executes in a number of steps

- Steps are controlled by a clock

- Within a period we have to read, compute and save

- For now, all we need to know is that there is a clock

\[ \text{period } T \quad \text{frequency } f = \frac{1}{T} \]
Simple example

```
+1  +1  +1
0   0   0
0+1 1   1
0+1 1+1 1+1
0   1   2
0+1 1+1 1+1
0   1   2
0+1 1+1 1+1
```

Modelling
Modeling

Mathematical domain

model type

model instance

When go==1 then blink leds five times

specification

Physical domain

model type

model instance

implementation

Modeling: Hardware design

Mathematical domain

Gezel: fdlsim

FSMD

Gezel

Gezel: fdlvhd

VHDL

VHDL: Modelsim

Physical domain

Xilinx FPGA

When go==1 then blink leds five times

specification

Xilinx
Modeling: Software design

Mathematical domain

Imperative languages

C

Compiler: gcc

ASM

linker

When go==1 then blink leds five times

specification

Physical domain

Pentium III

02131 Embedded Systems