Finite State Machines (FSM)

Lecturer: Guillaume Beslon
Original Author: Lionel Morel

Computer Science and Information Technologies - INSA Lyon

Fall 2017
Previously, on AC ...

Two ways of representing a sequential behavior:

- **time-diagrams**, called **chronogrammes**
  
  Describe a specific sequence of input/output values graphically

- **State machines**: 
  
  Describe the full system’s behavior **formally**
  
  see lecture on **FSMs**
Sequential behavior: Example

**Input:** $w$

**Output:** $z$

**Behavior:** $z$ is true whenever $w$ has been true for at least 4 clock cycles in a row.
Sequential behavior: Example
A finite state machine ("Machine a États Finis") is a tuple \((Q, q_0, I, T)\) with:

- \(Q\) is the set of states ("États")
- \(q_0 \in Q\) is the initial state ("État initial")
- \(I\) is the input alphabet ("Alphabet d’entrée")
- \(T \subseteq Q \times I \times Q\) is a transition function ("Fonction de transition").
FSM: graphical representation

\[ Q = \{q_0, q_1, q_2\} \]

\[ I = \{a, b, c\} \]
Acceptors ("Accepteurs" ou ‘Reconnaisseurs")

- Special form of FSM
- Have at least one accepting state
- (Regular languages) recognizers
Acceptors — Example

\[ Q = \{ q_0, q_1, q_2 \} \]

\[ I = \{ a, b \} \]
By the way ... why are we here?

We want to describe systems that produce certain output sequences based on input sequences...

But our FSMs don’t have outputs (yet).

Two steps at which outputs can be introduced:

- when the system is transitioning from one state to another
  ⇒ Mealy Machine
- when the system lies in one state
  ⇒ Moore Machine
A Mealy machine is a tuple \((Q, q_0, I, O, T)\) with:

- \(Q\) is the set of states
- \(q_0 \in Q\) is the initial state
- \(I\) is the input alphabet
- \(O\) is the output alphabet
- \(T \subseteq Q \times I \times O \times Q\) is a transition function.

On every transition, we read one input symbol/event and produce one output symbol/event (at least)
Mealy — Graphical Example

I = \{ a, b \}

O = \{ o \}

"emit o every time you see an 'a' followed by a 'b'"
Moore Machines

A Moore machine is a tuple \((Q, q_0, I, O, T, F)\) with:

- \(Q\) is the set of states
- \(q_0 \in Q\) is the initial state
- \(I\) is the input alphabet
- \(O\) is the output alphabet
- \(T \subseteq Q \times I \times Q\) is a transition function.
- \(F \subseteq Q \rightarrow O\) is an output function.
Blackboard
Intermission

https://youtu.be/aWVywhzuHnQ?t=1m3s
Mealy or Moore?

- Equivalent in expressiveness
- Translation from one to the other is always possible (at some costs)
- Mealy describes a very “event-driven” vision of the world
- In Computer Architecture, we will usually use Moore instead: we are not only interested in rising and falling edges (events), ie registers hold their value during cycles...
Synchronous Moore Machine

We want to describe **Synchronous Circuits**
Our automata should ultimately be driven by a clock

Blackboard
Synchronous Moore Machine

Most of the time, however, we will **not include clock** in description

⇒ Diagrams are more readable.

**Blackboard!**

 clk is always here, IMPLICITELY!!!
Running Example - specification (again!?)

Remember...

**Input:** \( w \)

**Output:** \( z \)

**Behavior:** \( z \) is true whenever \( w \) has been true for at least 4 clock cycles in a row.
Running Example

Blackboard
Reactivity and Determinism

In Hardware

our FSMs need to be reactive and deterministic:

Let’s note $c_{ij}$ the condition triggering transition between state $s_i$ and $s_j$. Let $Succ_i \in Q$ the set of all successors of state $s_i \in Q$.

**Reactive:** $\sum_{s_j \in Succ_i} c_{ij} = 1$

ie, in any given state, for any input value, the next state is known

ie, the component is never blocked.

**Deterministic:** $\forall (s_j, s_k) \in Succ_i \times Succ_i$, with $j \neq k$, $c_{ij}c_{ik} = 0$

ie, in any given state, for any input value, there exist only one next possible state
Blackboard