CSCI 1010
Models of Computation

Lecture 04
Finite State Machines
Overview

• Review of languages, functions, logic circuits
• FSM models and computing functions
• Simulation of FSMs by circuits
• Computational inequalities
• Example of FSM computation
Languages and Functions

• A **language** $L$ is a set of strings over an **alphabet** $A$.
  $\quad \rightarrow L \subseteq A^*$, the Kleene closure of the set $A$.

• A **Boolean function** $f : \{0,1\}^k \rightarrow \{0,1\}$

• **Binary functions** $f : \{0,1\}^k \rightarrow \{0,1\}^m,$
  $g : \{0,1\}^* \rightarrow \{0,1\}^*$.
Boolean Straight-Line Program

• (1 READ x)
• (2 READ y)
• (3 NOT 1)
• (4 NOT 2)
• (5 AND 1 4)
• (6 AND 3 2)
• (7 OR 5 6)

The SLP computes Exclusive OR

\[ g_7 = x \oplus y = (x \land \bar{y}) \lor (\bar{x} \land y) \]

\[(r \ OP\ s\ t) \iff g_r = g_s \ OP \ g_t\]
Binary Memory Cells

a) Clocked, D-type latch built from NAND gates
   When CLK = 1, output \( \rho = D, \rho^* = \overline{\rho} \). Held when CLK = 0.

b) Master-Slave D-type latch
   Output \( \rho \) set when CLK = 0. Changes only when CLK = 1
Memory

• Memory can be assembled from latches.
• A **register** is a collection of latches, e.g. 32
• A **memory word** is equivalent to a register.
• The technology of a memory varies:
  – Latches made from gates
  – Capacitors (they hold charge) as in DRAM
  – Flash memory (charge held on gate of transistor)
  – Phase-change devices (material change occurs).
Building FSM with Logic & Memory

• **State** of finite-state machine (FSM) is in memory
• Logic circuit computes next-state function $\delta$ and output function $\lambda$.
• FSM takes input, changes state, produces output.
The Finite State Machine (FSM)

• An FSM has sets $Q$ of states, $\Sigma$ of inputs, $\Psi$ of outputs, and a start state $s$.
• An FSM has next state function $\delta : Q \times \Sigma \rightarrow Q$ and output function $\lambda : Q \rightarrow \Psi$
• FSM with output is a six-tuple $(\Sigma, \Psi, Q, \delta, \lambda, s)$.
FSM Example

This is the parity FSM. It’s output is the parity of all its inputs.
FSM Computation

• Functions computed by an FSM \((\Sigma, \Psi, Q, \delta, \lambda, s)\).
  - On \(n\) inputs the FSM computes function \(f^{(n)} : \Sigma^n \rightarrow \Psi^n\)
    where \(f^{(n)} (x_1, x_2, \ldots, x_n) = (y_1, y_2, \ldots, y_n)\)
  - The FSM computes the functions \(f = \{f^{(1)}, f^{(2)}, f^{(3)}, \ldots\}\).

• Languages
  - Let an FSM have accept states \(F\).
  - An FSM accepts those strings that take it from the initial state to an accept state.
  - It recognizes the language consisting of these strings.
    • FSM example accepts strings with odd parity (odd no. 1s)
Simulating an FSM by a Circuit

- FSM computes $f^{(T)} = (y_1, y_2, ..., y_T)$ in $T$ cycles.

- Can $f^{(T)}$ be computed by a circuit? **Unwind loop!**
A Computational Inequality

• Let $C_{\Omega}(f)$ be the size of the smallest circuit using gates in the set $\Omega$ for the function $f$. $\Omega$ is the basis for circuit.

**Theorem** Let the FSM $L$ with next-state and output functions $\delta$ and $\lambda$ compute $f^{(T)}$. Then,

$$C_{\Omega}(f^{(T)}) \leq C_{\Omega}(\delta, \lambda) \cdot T$$

**Proof** $f^{(T)}$ requires circuit of size $\geq C_{\Omega}(f^{(T)})$. Circuit simulating $f^{(T)}$ uses at most $C_{\Omega}(\delta, \lambda)T$ gates. Thus, $C_{\Omega}(f^{(T)}) \leq C_{\Omega}(\delta, \lambda) \cdot T$.

• If $C_{\Omega}(f^{(T)}) > C_{\Omega}(\delta, \lambda)T$, $f^{(T)}$ cannot be computed.
• While it may be possible to “trade” $C_{\Omega}(\delta, \lambda)$ for $T$, the inequality can’t be violated.
An FSM as Language Recognizer

• The FSM computes $f = \{f^{(1)}, f^{(2)}, f^{(3)}, \ldots \}$.

• $L_\alpha$ is the **language** of inputs that cause the last output of $f^{(T)}$ to be $\alpha$ in $\Psi$, the set of **outputs**.

• Languages associated with an FSM can also be defined using **final states** $F$.
  – Let $F$ (a subset of $Q$) be the **final states** of an FSM.
  – The inputs that cause the FSM to end on a final state in $F$ is the **language recognized** by the FSM.
Analyzing an FSM

- What are the inputs $\Sigma$, outputs $\Psi$, and states $Q$?
- What does this machine do? Adds two binary numbers.
- What is circuit for next state and output functions?
- Can you simulate the FSM with a circuit?

$Q = \{q_0, q_1, q_3, q_4\}$
$\Sigma = \{(0,0),(0,1),(1,0),(1,1)\}$
$\Psi = \{0,1\}$
Analyzing an FSM

• FSM adds binary numbers \((x_{n-1}, \ldots, x_1, x_0)\) and \((y_{n-1}, \ldots, y_1, y_0)\) to produce \((z_n, \ldots, z_1, z_0)\)
• State \(q_k\) represents \(k = (c, s)\), \(c = \text{carry}, s = \text{sum}\).
• Initially \(c = s = 0\) (state is \(q_0\))
• \(c_r\) and \(s_r\) are determined by input \((x_r, y_r)\), & previous carry \(c_{r-1}\).
• Output is \(s_r\).
Next-State & Output Functions $\delta, \lambda$
Computing $\delta$ of an FSM

- Circuit computing
  
  $(c_r, s_r) = \delta(x_r, y_r, c_{r-1})$

<table>
<thead>
<tr>
<th>$X_r$</th>
<th>$Y_r$</th>
<th>$C_{r-1}$</th>
<th>$C_r$</th>
<th>$S_r$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Introduction to Turing Machines

- It models human computation. TM has an FSM **control unit** (CU) and a potentially infinite **tape**.
- CU input from head, changes state, replaces cell value & moves head by at most one cell.
Standard TM Protocol

• Input written left-adjusted on non-blank tape
• If TM halts, the output is the left-adjusted non-blank string on the tape.
• The TM computes the function from inputs to outputs. This function may be partial, that is, not defined for all strings.
A Turing Machine Computation

• Recognizing balanced strings \( \{0^m1^m \mid m \geq 0\} \).

1. If current symbol is? (Testing first symbol)
   - 1: stop and reject.
   - 0: write blank \( \beta \), shift right, and go to #2
   - \( \beta \): stop and accept.

2. If current symbol is? (Move right scanning 0s)
   - 0: write 0, shift right, and go to #2.
   - \( \beta \): stop and reject.
   - 1: write 1, move right, go to #3.
A Turing Machine Computation

- Recognizing balanced strings \( \{0^m1^m \mid m \geq 1\} \).

3. If current symbol is? (Move right scanning 1s)
   - 0: stop and reject.
   - 1: write 1, shift right, and go to #3
   - \( \beta \): write \( \beta \), move left, write \( \beta \), move left, go to #4

4. If current symbol is? (Move left scanning 1s)
   - If 1, write 1, move left, and go to #4.
   - If 0, write 0, move left, and go to #5.
   - If \( \beta \), move right, go to #1
A Turing Machine Computation

• Recognizing balanced strings \( \{0^m1^m \mid m \geq 1\} \).
  
5. If current symbol is? (Move left scanning 0s)
  • 0: write 0, move left, go to #5.
  • \( \beta \): move right, go to #1

• This algorithm can be mapped to a state diagram for an FSM. Each step corresponds to one state.
TM State Diagram for \( \{0^m1^m \mid m \geq 1\} \)

- **Notation**: \( u/v, M \) means on input \( u \), write \( v \), move right (\( M=R \)), left (\( M=L \)), or not at all (\( M=N \)). \( z/H \) means write \( z \) and halt.
Review

• Languages, Boolean functions, logic circuits
• FSM models and computing functions
• Simulation of FSMs by circuits
• Computational inequalities
• Example of FSM computation
• The Turing Machine (TM)
• Example of TM computation
• Goal: classify languages by time and space