16.070

Introduction to Computers & Programming

Theory of computation: What is a computer? FSM, Automata
What is a computer?

- If you can’t measure it it has no value…
  - Quantitative (numerical)
  - Qualitative
- Can we model a computer as we know it today?
## Models of Computation

<table>
<thead>
<tr>
<th>Uncomputable</th>
<th>Complex</th>
<th>Crude</th>
</tr>
</thead>
<tbody>
<tr>
<td>Turing Machines</td>
<td>Phrase Structure</td>
<td></td>
</tr>
<tr>
<td>Linear bounded automata</td>
<td>Context-sensitive</td>
<td></td>
</tr>
<tr>
<td>Pushdown automata</td>
<td>Context-free</td>
<td></td>
</tr>
<tr>
<td>Finite state automata</td>
<td>Regular</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Machines</th>
<th>Grammars/Languages</th>
</tr>
</thead>
</table>

- Regular
- Finite state automata
- Linear bounded automata
- Pushdown automata
- Turing Machines
- Phrase Structure
- Context-sensitive
- Context-free
- Regular
Finite Machines

- Think of a black box which takes inputs from the environment and produces some kind of observable response. Examples are e.g., vending machine, dish washer, automatic door opener.
- A finite machine has a finite memory. It can only distinguish between a finite number of input histories.
- Each class of equivalent histories corresponds to a state of the machine.

A Finite State Automata (FSA) is an abstract finite machine.
One entry found for automaton.

Main Entry: au·tom·a·ton
Pronunciation: o-'tä-m&-t&n, -m&-"tän
Function: noun
Inflected Form(s): plural -atons or au·tom·a·ta /-m&-t&, -m&-"tä/
Etymology: Latin, from Greek, neuter of automatos
Date: 1645
1 : a mechanism that is relatively self-operating; especially :
   ROBOT
2 : a machine or control mechanism designed to follow
   automatically a predetermined sequence of operations or respond to
   encoded instructions
3 : an individual who acts in a mechanical fashion
Theory of Computation

1. **Finite state automata**: deterministic and non-deterministic state machines, regular expressions and languages. Techniques for identifying and describing regular languages; techniques for showing that a language is not regular. Properties of such languages.

2. **Context-free languages**: Context-free grammars, parse trees, derivations and ambiguity. Relation to pushdown automata. Properties of such languages and techniques for showing that a language is not context-free.
3. **Turing Machines**: Basic definitions and relation to the notion of an algorithm or program. Power of Turing Machines.

4. **Undecidability**: Recursive and recursively enumerable languages. Universal Turing Machines. Power of Turing Machines.

5. **Computational Complexity**: Decidable problems for which no sufficient algorithms are known. Polynomial time computability. The notion of NP-completeness and problem reductions. Example of hard problems.
A simple finite automaton; an on/off-switch

- Circles represent **states**. In this case named *On* and *Off*.
- Edges (arcs) represent **transitions** or **input** to the system.
- Start arrow indicates which state we start in.

Initially the switch is Off. When the switch encounters a Push it changes state into the On state. When another Push is encountered we switch into the Off state and so on.
Finite Automata

- Software to design and verify circuit behavior
- Lexical analyzer of a typical compiler
- Parser for natural language processing
- An efficient scanner for patterns in large bodies of text (e.g. text search on the web)
- Verification of protocols (e.g. communications, security).
- …
Moore and Mealy Machines

- Two types of machines: **Moore** and **Mealy**. The difference lies in the outputs.
  - **Mealy Machines**
    - The output is a function of the present state and all the inputs
    - Input change causes an immediate output change
  - **Moore Machines**
    - The output is a function of the present state only
    - Outputs change synchronously with state changes
Finite Automata

![Diagram of Finite Automata]

- Lab 1
- Lab 2
A finite automaton called $M_1$

The figure is called the *state diagram* of $M_1$

It has 5 *states* labeled $q_0, q_1, q_2, q_3, q_4$,

The *start state* is labeled $q_0$

The *accept state* $q_2$, is the one with double circles

The arrows going from one state to another are called *transitions*
Formal Definition of a Finite Automaton

- An FSA is a 5-tuple \((Q, \Sigma, \delta, q_0, F)\)
  1. \(Q\) is a finite set called the **states**
  2. \(\Sigma\) is a finite set called the **alphabet**
  3. \(\delta: Q \times \Sigma \rightarrow Q\) is the **transition function**
  4. \(q_0 \in Q\) is the **start state**
  5. \(F \subseteq Q\) is the **set of accept states** (final states)

\[
\begin{align*}
Q &= \{q_1, q_2, q_3\} \\
\Sigma &= \{0, 1\} \\
\delta &\text{ is described as}
\end{align*}
\]

\[
\begin{array}{c|cc}
q_1 & q_1 & q_2 \\
q_2 & q_3 & q_2 \\
q_3 & q_2 & q_2 \\
\end{array}
\]

- \(q_1\) is the start state
- \(F = \{q_2\}\)
Formal Definition of a Finite Automaton

- If $A$ is the set of all strings that machine $M$ accepts, we say that $A$ is the **language of machine $M$** and write $L(M) = A$.

- We say that $M$ recognizes $A$ (or that $M$ accepts $A$).

- A machine may accept several strings, but it only recognizes **one** language.
Finite Automaton $M_2$

- State diagram of finite automaton $M_2$

$M_2 = (\{q_1, q_2\}, \{0, 1\}, \delta, q_1, \{q_2\})$

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_1$</td>
<td>$q_1$</td>
<td>$q_2$</td>
</tr>
<tr>
<td>$q_2$</td>
<td>$q_1$</td>
<td>$q_2$</td>
</tr>
</tbody>
</table>

What strings does $M_2$ accept?
Finite Automaton $M_3$

- State diagram of finite automaton $M_3$

\[ L(M_3) = \{ \omega \mid \omega \text{ is the empty string } \epsilon \text{ or ends in a } 0 \} \]
Alphabet $\Sigma = \{a, b\}$

What does $M_4$ accept?

All strings that start and end with $a$, or that start and end with $b$. In other words, $M_4$ accepts strings that start and end with the same symbol.
Finite Automaton $M_5$

Alphabet $\Sigma = \{<\text{reset}>, 0, 1, 2\}$

What does $M_5$ accept?

$M_5$ keeps a running count of the sum of the numerical input symbols it reads, modulo 3. Every time it receives the $<\text{reset}>$ symbol it resets the count to 0.
Finite Automaton $M_6$

- Is it possible to describe all finite automata by a state diagram?
- No: if diagram is too large to draw
- No: if description depends on some unspecified parameter

$$B_i = (Q_i, \Sigma, \delta_i, q_0, \{q_0\})$$
Formal Definition of Computation

- Let $M = (Q, \Sigma, \delta, q_0, F)$ be a finite automaton
- Let $w = w_1w_2…w_n$ be a string over the alphabet $\Sigma$

Then $M$ accepts $w$ if a sequence of states $r_0, r_1, \ldots, r_n$ exists in $Q$ with the following three conditions:

1. $r_0 = q_0$
2. $\delta(r_i, w_{i+1}) = r_{i+1}$ for $i = 0, \ldots, n-1$
3. $r_n \in F$

- $M$ recognizes language $A$ if $A = \{w \mid M$ accepts $w\}$
A language is called a **regular language** if some finite automaton recognizes it.