Find your way here

CSC 250: Foundations of Computer Science I

Fall 2023 - Lecture 16

Amitabha Dey

Department of Computer Science
University of North Carolina at Greensboro

State Machines

Automata theory is the study of abstract machines and automata, as well as the computational problems that can be solved using them. It is a theory in theoretical computer science with close connections to mathematical logic.
A **state machine** is a mathematical abstraction used to design algorithms.

In simpler terms, a state machine will read a series of inputs. When it reads an input, it will switch to a different state. Each state specifies which state to switch to, for a given input. This sounds complicated but it is really quite simple.

Imagine a device that reads a long piece of paper. For every inch of paper there is a single letter printed on it–either the letter ‘a’ or the letter ‘b’. As the state machine reads each letter, it changes state. Here is a very simple state machine:

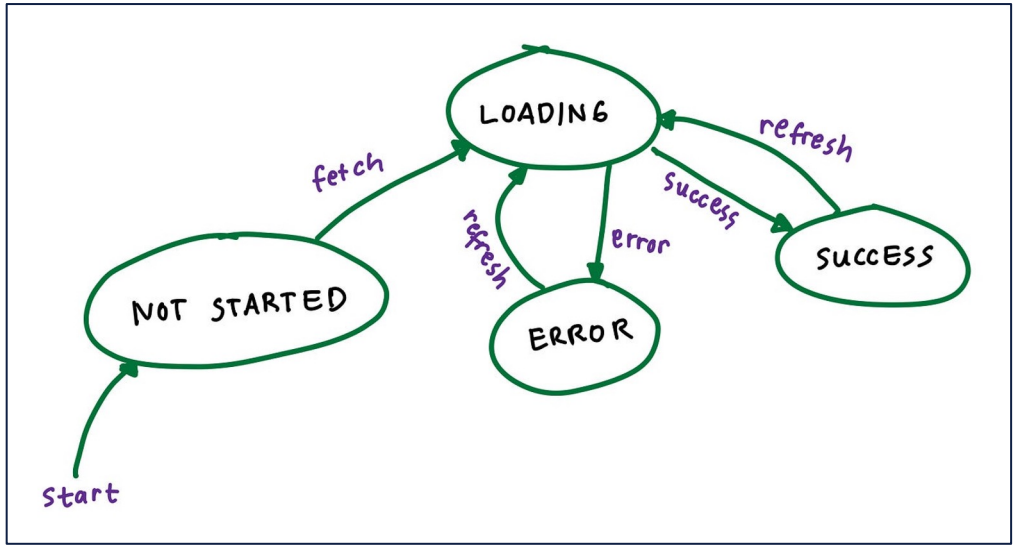

Image credit - Yolanda Septiana @ Medium.com

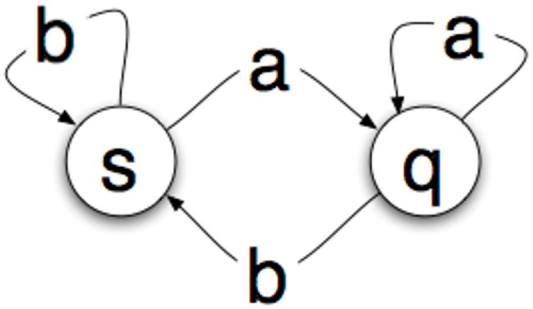

Image credit - https://www.freecodecamp.org/

The circles are “**states**” that the machine can be in. The arrows are the **transitions**. So, if you are in state *s* and read an ‘a’, you’ll transition to state *q*. If you read a ‘b’, you’ll stay in state *s*.

Finite State Machines

A Finite State Machine, or FSM, is a computation model that can be used to simulate sequential logic, or, in other words, to represent and control execution flow. Finite State Machines can be used to model problems in many fields, including mathematics, artificial intelligence, games or linguistics.

A Finite State Machine is a model of computation based on a hypothetical machine made of one or more states. Only one single state of this machine can be active at the same time. It means the machine has to transition from one state to another in to perform different actions.

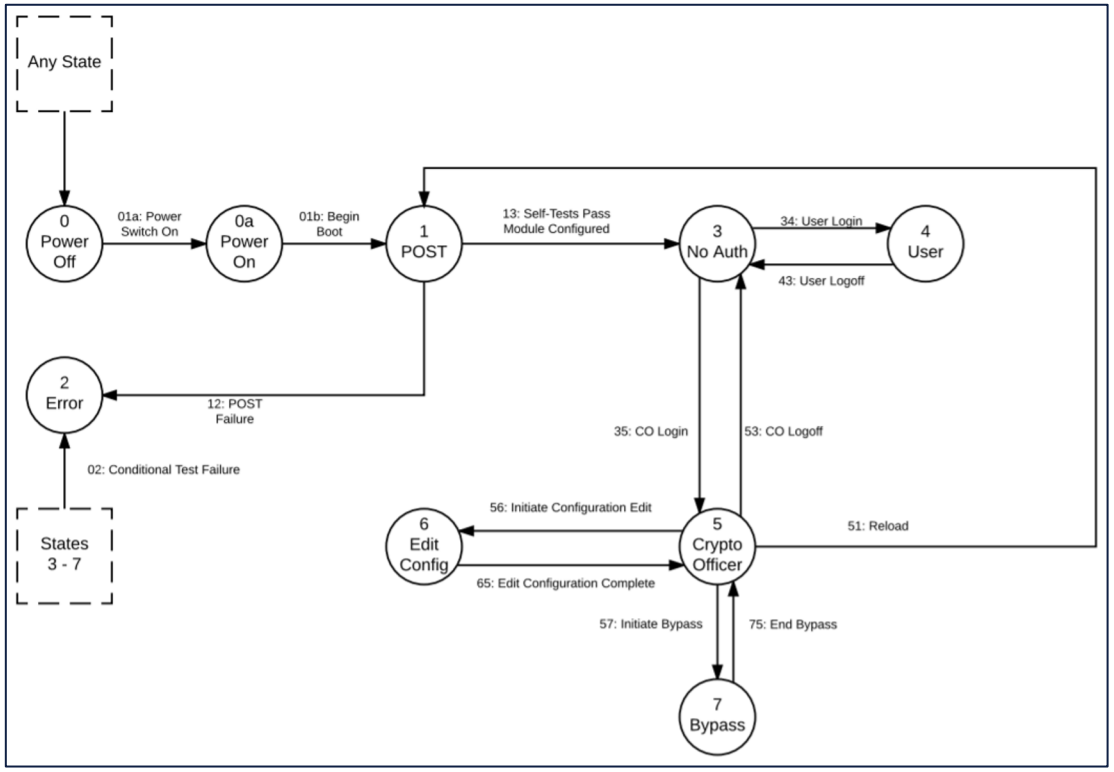

Image credit - https://docs.oracle.com/

The important points here are the following:

- We have a fixed set of states that the machine can be in
- The machine can only be in one state at a time
- A sequence of inputs is sent to the machine
- Every state has a set of transitions and every transition is associated with an input and pointing to a state

Turing Machine

A Turing machine is an abstract computational model that performs computations by reading and writing to an infinite tape. Turing machines provide a powerful computational model for solving problems in computer science and testing the limits of computation — are there problems that we simply cannot solve?

Image credit - https://blogs.scientificamerican.com/

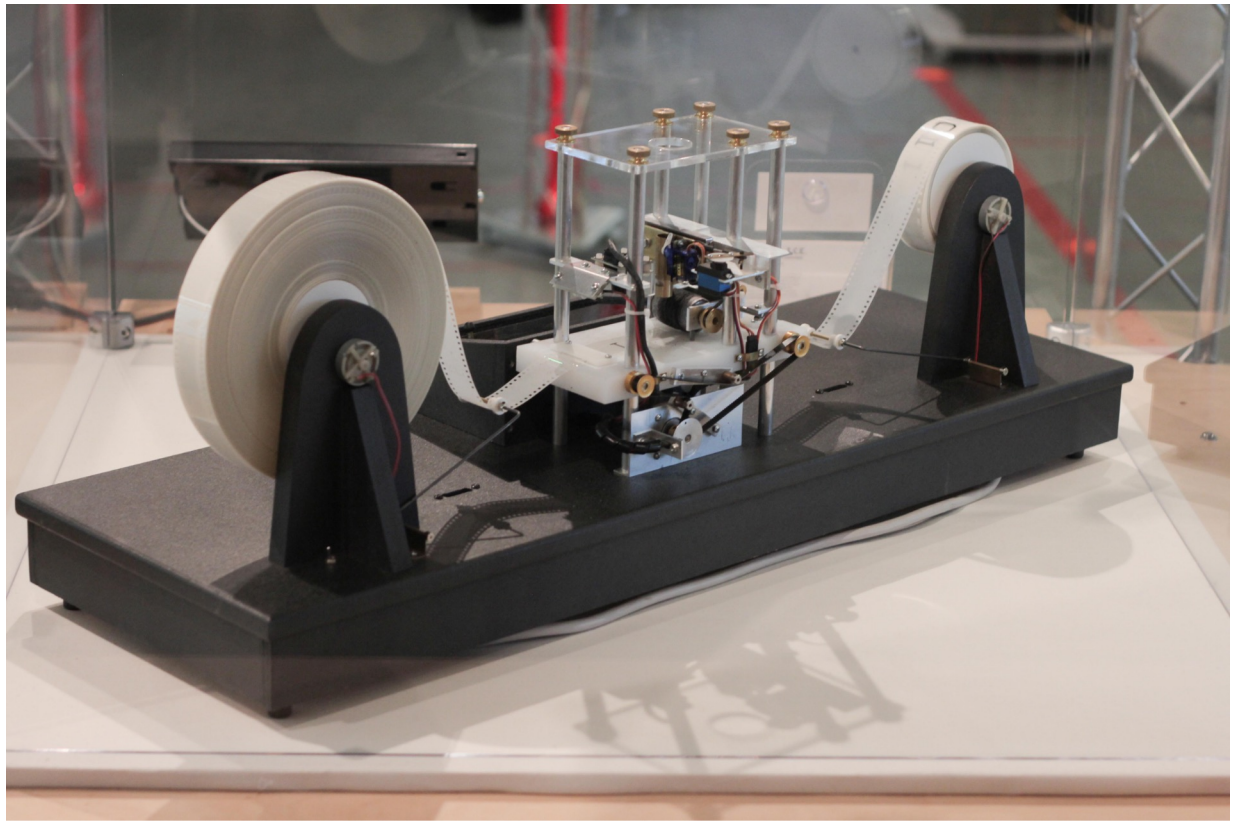

Image credit - https://blogs.scientificamerican.com/

A Turing machine consists of an infinite tape (as the memory), a tape head (a pointer to the currently inspected cell of memory), and a state transition table (to govern the behavior of the machine). During operation, the tape head is in a certain state. Every step, it consults the table (the set of transition functions), based on only the state it's in and the symbol underneath the head, to obtain its next choice: either halt (end the operation), or resume by writing a symbol to the current cell, changing its state, and moving to the left or the right. This way, A Turing machine can simulate the fact that a program is made of many lines and thus it depends on what line a program is executing, and it can also simulate the fact that a program can react differently with different data in memory.

Deterministic Finite Automata (DFA)

- DFA refers to deterministic finite automata. Deterministic refers to the uniqueness of the computation. The finite automata are called deterministic finite automata if the machine is read an input string one symbol at a time.
- In DFA, there is only one path for specific input from the current state to the next state.
- DFA does not accept the null move, i.e., the DFA cannot change state without any input character.
- DFA can contain multiple final states. It is used in Lexical Analysis in Compiler.

A deterministic finite automaton (DFA) consists of five things:

- an input alphabet
- a finite set S whose elements are called states,
- a distinguished state called starting state,
- a set distinguished states, called accepting states (or final states),
- a partial function called the transition function

A ***finite automaton*** is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set called the ***states***,
2. Σ is a finite set called the ***alphabet***,
3. $\delta\colon Q \times \Sigma \longrightarrow Q$ is the ***transition function***,[1]
4. $q_0 \in Q$ is the ***start state***, and
5. $F \subseteq Q$ is the ***set of accept states***.[2]

Deterministic Finite Automata (DFA)