CS301: Theory of Computation
Section 1.1 - Finite Automata

Daniel White

Gettysburg College
Computer Science

Spring 2024

What is it possible for a computer to do? What is outside the realm of computation?

...What *is* a computer? What *is* computation?

It will be our goal this semester to establish a manageable mathematical theory of "computer" and "computation" to answer these questions.

Definition (pg. 3)

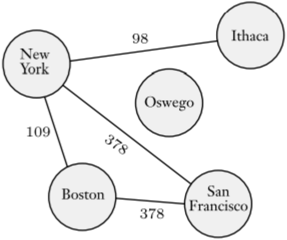
A **set** is a group of objects represented as a unit. The objects in a set are called its **elements** or **members**.

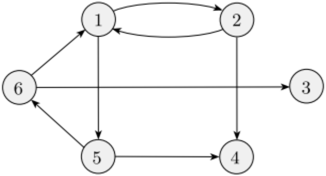
exercise: Recall (and formally express) some common sets found in mathematics. *e.g.* $\mathbb{Z}$ and $\mathbb{R}^2$.

exercise: What is a Cartesian product? What is a k-tuple?

Definition (pg. 10)

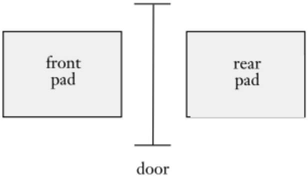
A **graph** is a set of vertices (nodes) connected by either directed or undirected edges.

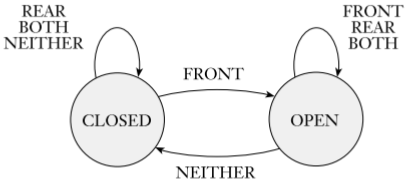




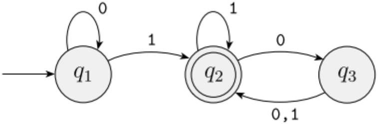
Sometimes the vertices and/or the edges of a graph will be labeled.

The following example, with its state diagram, illustrates how a controller for a one-way automatic swinging door would make decisions to open and close.

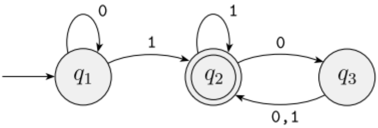




Before we provide a formal definition of **finite automaton**, consider the following **state diagram** of machine M_1.



- The **start state** q_1 is indicated by the arrow pointing at it from nowhere.
- The **accept state** q_2 is the one with a double circle.
- The directed edges leading from one state to another are called **transitions**.



exercise: Describe the set of "input values" that the machine M_1 can function over.

exercise: Walk through the computation over "input values" 1100 and 00010.

exercise: Describe the set of all "input values" that end at the accept state q_2.

$\boxed{\text{homework 1.1}}$

exercise: Create the state diagram of a machine M_4 that accepts a non-empty string of a's and b's if and only if the first and last characters are equal.

Definition 1.5 (pg. 35)

A **finite automaton** (DFA) 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. $\delta : Q \times \Sigma \to Q$ is the **transition function**,
4. $q_0 \in Q$ is the **start state**, and
5. $F \subseteq Q$ is the **set of accept states**.

exercise: Express the machine M_1 from the previous slides formally as a finite automaton.

homework 1.2

Definition (pg. 13)

An **alphabet** is a non-empty finite set. The members of the alphabet are the **symbols** of the alphabet.

We generally use Σ and Γ to designate alphabets.

Definition (pg. 14)

A **string over an alphabet** is a finite sequence of symbols from that alphabet.

For example, consider the alphabet $\Sigma = \{\mathtt{x}, \mathtt{y}, \#\}$.

- The strings $\#\mathtt{yyx}$ and $\#\#\#$ are strings over Σ. The empty string ε is also a string over Σ.
- The string $\mathtt{xyz}$ is not a string over Σ.

Definition (pg. 14)

A **language** is a set of strings over some alphabet.

Definition (pg. 40)

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.

So, for example, the machine M_4 recognizes the language A, where A is the set of all non-empty strings over the alphabet $\{\mathtt{a}, \mathtt{b}\}$ with matching first and last character.

Definition 1.16 (pg. 40)

A language is called a **regular language** if some finite automaton recognizes it.

exercise: Consider the alphabet $\Gamma = \{0, 1, 2\}$ and the language $A = \{x_1 \ldots x_k \mid x_i \in \Gamma, k \geq 1, \text{ and } x_1 + \ldots + x_k \equiv 0 \pmod 3\}$. Show that A is a regular language.

$\boxed{\text{homework: 1.5ab}}$

Definition 1.23 (pg. 44)

Let A and B be languages. We define the regular operations as follows:

1. **union:** $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$
2. **concatenation:** $A \circ B = \{xy \mid x \in A \text{ and } y \in B\}$
3. **star:** $A^* = \{x_1 x_2 \ldots x_k \mid k \geq 0 \text{ and each } x_i \in A\}$

exercise: Consider the languages $A = \{\mathtt{a}, \mathtt{aa}, \mathtt{aac}\}$ and $B = \{\mathtt{b}, \mathtt{bc}\}$. Can you explicate the result of any of the operations above on these languages?

Theorem 1.25 (pg. 45)

The class of regular languages is closed under the union operation. In other words, if A_1 and A_2 are regular languages, so is $A_1 \cup A_2$.

Proof. Let A_1 and A_2 be regular languages. By definition, there exist finite automata M_1 and M_2 that recognize them, respectively. We proceed via construction to show that there exists a finite automata M that uses both M_1 and M_2 together to recognize $A_1 \cup A_2 \ldots$

exercise: Finish this proof. *(pgs. 45-46)*

homework: 1.4bd

Theorem 1.26 (pg. 47)

The class of regular languages is closed under the concatenation operation. In other words, if A_1 and A_2 are regular languages, so is $A_1 \circ A_2$.

To prove this theorem, we'd hope to try something similar to the previous proof. However, we would need the machine to know when to "stop testing" for a string from A_1 and to "start testing" for a string from A_2. This could happen at any point in an input string!

To make this problem easier, we will introduce the notion of **nondeterminism** in the next section.