Strings & Languages

Sequences or strings of symbols and characters are fundamental in computer science. We use strings of symbols to represent data and most importantly we use them to represent algorithms.

We formalize these structures as follows:

An **alphabet** is any non-empty, finite set. The elements of an alphabet are called **symbols**.

Examples:

$$\Sigma = \{0, 1\}$$

$$\Sigma' = \{a, b, c, d\}$$

$$\Gamma = \{this, and, that\}$$

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

Example: Given the alphabet $\Sigma = \{0, 1\}$ then the following are strings over this alphabet: 011001, 000000, 1.

Operations on Strings

If w is a string over Σ, then the **length** of w, written $|w|$, is the number of symbols w contains.

The **empty string**, usually written as ϵ, is a string where $|\epsilon| = 0$.

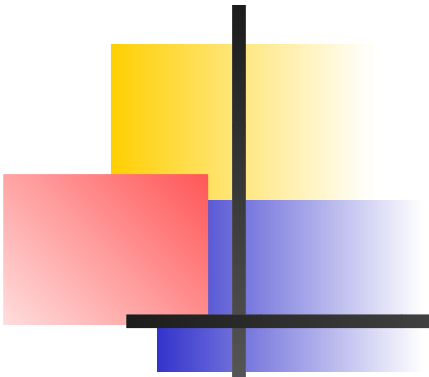
Given a string w of length n, we can write $w = w_1 w_2 \dots w_n$ where $w_i \in \Sigma$. Furthermore, we can define the **reverse** of w by $w^R = w_n w_{n-1} \dots w_1$.

Some string z is a **substring** of w if z appears consecutively within w, e.g., cd is a substring of $abcdef$.

Given two strings $x = x_1 x_2 \dots x_n$ and $y = y_1 y_2 \dots y_m$ where $x_i, y_j \in \Sigma$ for $i = 1 \dots n$ and $j = 1 \dots m$, then the **concatenation** of x and y is obtained by appending y to the end of x written as $xy = x_1 x_2 \dots x_n y_1 y_2 \dots y_m$ with $|xy| = n + m$.

Given a string that consists of k copies of the same symbol, we use a superscript notation for compactness,

$$\overbrace{xx \dots x}^{k} = x^k.$$



Languages

Definition: *A* **language** *is a set of strings.*

Languages

Example: The language L_2 of all 2 bit words. Let $\Sigma = \{0, 1\}$ be the alphabet, then we can construct strings such as 01, 0001, *etc.*. But our language is only the set of all 2 bit words, thus

$$L_2 = \{00, 01, 10, 11\}.$$

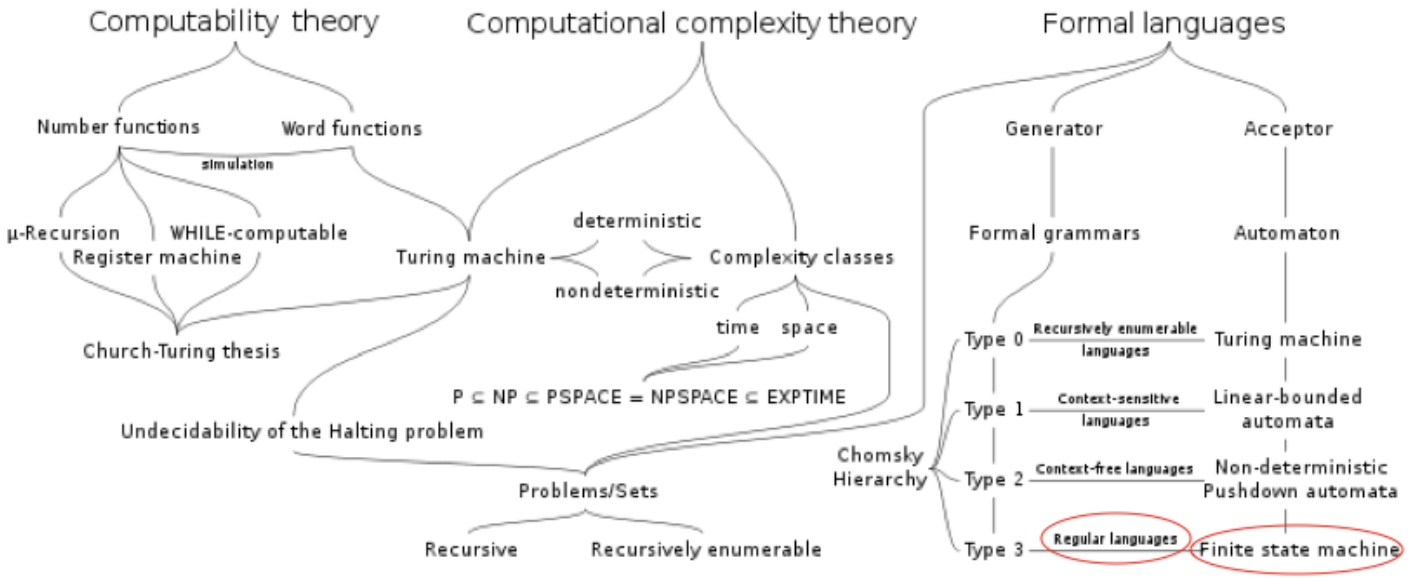
Example: The language L_a consisting of words of five or less a's. Let $\Sigma = \{a\}$ be the alphabet, then we construct strings such as a, aaa, then our language is

$$L_a = \{a, aa, aaa, aaaa, aaaaa\}.$$

Note: The empty language is not equivalent to the language that has the empty string as its only member,

$$L_\emptyset = \{\} \not\equiv L_\epsilon = \{\epsilon\}.$$

Course Road Map



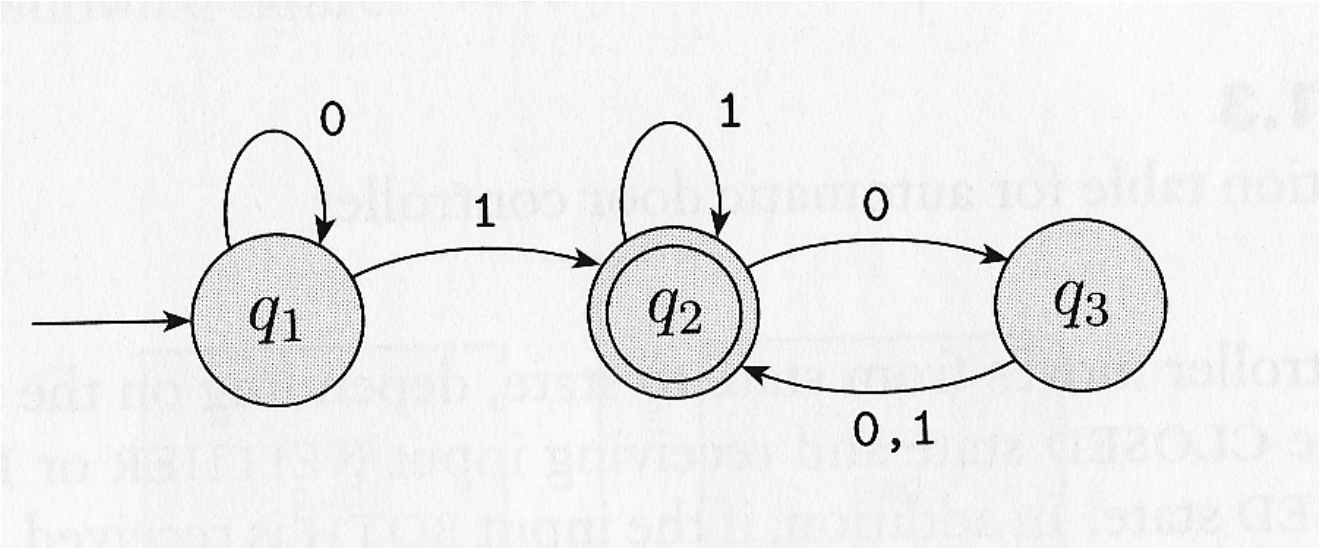
Type 3: Regular Expressions

Regular Languages

- The simplest model of computation we know is the **finite automaton**.
- As we will see, this machine computes via **state transitions** given **input strings**.
- We say that a computation is successful if a FA **accepts** a particular string.
- If all the strings of a particular language are accepted then we say that the FA **recognizes** that language.

In the following we formally define these notions.

FA - Example



Terms: states, start state, accept states, state transitions, alphabet.

Which of the strings would the above machine accept: 000, 01, 010?

What would a machine look like for L_2 and L_a, respectively?

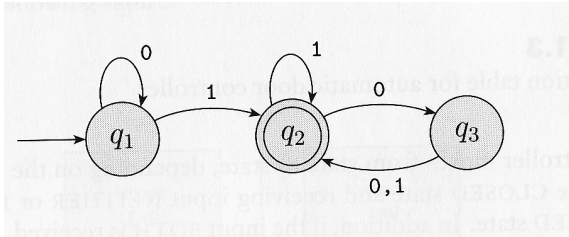
Formal Def. of FA

Pictures are nice but they are not very useful when we try to develop a mathematical theory of computing. We need a more formal definition:

Definition: 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. $\delta : Q \times \Sigma \longrightarrow Q$ is the **transition function**,
4. $q_0 \in Q$ is the **start state**, and
5. $F \subseteq Q$ is the **set of accept states**.

FA - Example, again



1. $Q = \{q_1, q_2, q_3\}$,
2. $\Sigma = \{0, 1\}$,
3. δ : $Q \times \Sigma \rightarrow Q$ can be described by a table,

	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

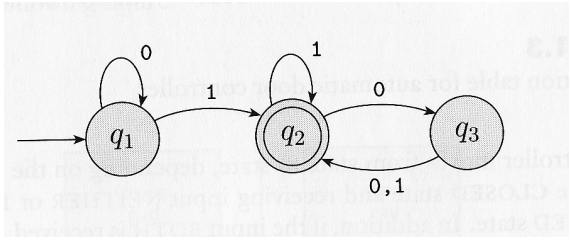
4. $q_1 \in Q$ is the **start state**, and
5. $F = \{q_2\}$.

Language of a Machine

Let M be some FA and A be the set of all strings that the machine accepts, then we say that A is the **language of machine** M or, alternatively, that M **recognizes** A and we write

$$L(M) = A.$$

Example: Let M_1 be our machine



we can determine that this machine accepts all the strings in the set

$$B = \{w \mid w \text{ contains at least one } 1 \text{ followed by a combination of } 1,\ 00, \text{ or } 01\},$$

then we say that $L(M_1) = B$ or M_1 recognizes B.

Formal Def. of Computation

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a FA and let $w = w_1 w_2 \dots w_n$ be a string with $w_i \in \Sigma$, then M **accepts** w if a sequence of states $q_0, q_1, \dots, q_n$ with $q_i \in Q$ exists with conditions:

1. $\delta(q_i, w_{i+1}) = q_{i+1}$, for $i = 0, \dots, n-1$, and
2. $q_n \in F$.

Sometimes we write

$$q_0 w_1 w_2 \dots w_n \vdash w_1 q_1 w_2 \dots w_n \vdash \dots \vdash w_1 w_2 \dots q_{n-1} w_n \vdash w_1 w_2 \dots w_n q_n$$

for the computation.[a]

We say that M **recognizes language** A if $A = \{w \mid M \text{ accepts } w\}$.

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

[a] Here the symbol $\vdash$ means 'entails'.