Lec 01. Intro & DFA

Eunjung Kim

Fundamental questions for CS

- What do we mean by "computation"?
  $\rightsquigarrow$ "Computation is to solve a problem by an effective manner."
  Vague.
- What is a computer? Whay all 'computers' are all called computers?
- What can it do and cannot?
- Computable vs not computable problems? 'Efficientlly' computable problems and those which are not?
- Can anyone on earth devise a fundamentally more powerful computer? (An alien? In another universe?)

How to express the object for computation: alphabet

Alphabet

- Alphabet, usually denoted as Σ, is a finite and nonempty set of symbols.
- Examples of alphabet: $\Sigma = \{0, 1\}$, $\{a, b, \ldots, z\}$, the set of all ASCII characters, etc.

How to express the object for computation: string

String

- String (a.k.a. word) is a finite sequence of symbols over Σ.
- Length of a string: number of symbols.
- Length-0 is a string itself, often denoted as ϵ.
- Σ^i: the set of all strings of length i.

How to express the object for computation: concatenation

Concatenation

- Operation on two strings.
- x, y are strings $\rightsquigarrow$ their concatenation xy is a string.
- $\epsilon x = x\epsilon = ?$

How to express the object for computation: language

Language

- Language (over alphabet Σ) is a set of strings over Σ.
- Simply put, $L \subseteq \Sigma^*$.
- Here, Σ^* is a set of all strings of finite length, $\Sigma^* := \bigcup_{i \geq 0} \Sigma^i$.
- Examples of languages: ...
- Both $\emptyset$ and $\{\epsilon\}(= \Sigma^0)$ are languages.

Computation: what and how

- Any well-formulated information can be represented as a string of 0 and 1, or any finite alphabet Σ.
- The object for computation can be stated as a function.

Compute WHAT

computational problem $\Leftrightarrow$ compute a function $f : \Sigma^* \to \Sigma^*$.

Decision problem and Language

Compute WHAT

a decision problem $\Leftrightarrow$ compute a function $f : \Sigma^* \rightarrow \{0, 1\}$.
$\Leftrightarrow$ given $x \in \Sigma^*$, decide if $x \in L$
where $L = \{f(s) = 1\} \subseteq \Sigma^*$.

- Decision problem, or equivalently "membership test for a language", is easier to handle.
- ...while being capable of capturing the essence of important computational problems.

Computation: what and how

Compute **HOW**

- Let us agree: "computing a function f" means "there is an ~~effective method~~algorithm which outputs $f(x)$ for each input x".
- The concept of "algorithm" is still vague.

Computation: what and how

What do we expect for an algorithm, intuitively?

- a finite number of finitely describable instructions.
- each instruction and what to do next are unambiguous.
- all the basic operation should be executable by the concerned executor.
- terminates at some point (i.e. in finite number of steps)

Toward a rigorous notion of algorithm

A mathematically rigorous description of an executor (computing device/machine...) and instructions is needed.

(Our) Model of Computation

Exercutor(machine) constituents:

- an alphabet Σ it recognizes,
- a gadget to read an input $x \in \Sigma^*$,
- a finite set of states to recognize its status ("where am I?"),
- memory to write and read later.

Basic operation:

- read one alphabet from input tape (or from memory),
- update its internal state,
- move the header (only in one fixed direction, or both direction, or neither) on input tape or memory,
- write/change on memory tape.

Set-up

- Concatenation xy of x and y.
- Cartesian product $A \times B$
- Notations: Σ, Σ^i, ϵ, Σ^*.
- Computing a function $f : \Sigma^* \to \Gamma^*$ means ...
- Special function $f : \Sigma^* \to \{0, 1\} \rightsquigarrow$ language.
- Language: a subset A of Σ^*, indicator function f_A.
- Computing $f_A \Leftrightarrow$ membership test for A

Finite (state) automata

Example: automatic door

Model of computation mimicking a simple computing device

- no/limited memory,
- basic operations: read one symbol from the input, update the state, and move to the next position in input.

(State) Transition Diagram

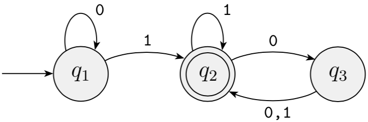


Figure 1.4, Sipser 2012.

Strings accepted by M

The set of all $w \in \{0, 1\}^*$ such that...

Formal definition

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

- Q a finite set called the states,
- Σ a finite set called the alphabet,
- δ a function from $Q \times \Sigma$ to Q called the transition function,
- $q_0 \in Q$ the start state,
- $F \subseteq Q$ the set of accept states.

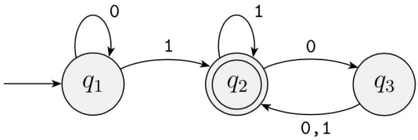


Figure 1.4, Sipser 2012.

Transition Diagram, Transition Table

(Other than listing the transition function) two common ways to express transition function.

Language recognized by FA

Definition

- Let M be a finite automata.
- A string $w \in \Sigma^*$ is accepted by a finite automata M if M ends in an accept state upon reading the entire w.
- $L(M)$ denotes the set of all strings accepted by M.
- A language A is said to be recognized by M if $A = L(M)$.

Examples of finite automata

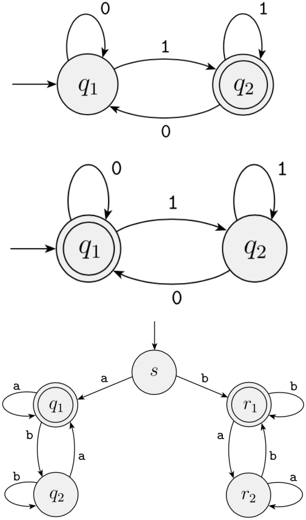


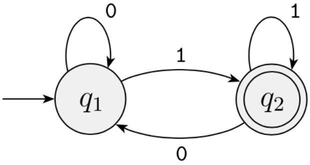
Figure 1.7, 9, 12 from Sipser 2012.

Formal definition of computation

- Let $w = w_1 w_2 \cdots w_n \in \Sigma^*$, where $w_i \in \Sigma$.

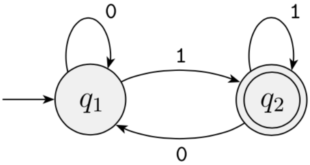
- The extended transition function δ^* is a mapping from $Q \times \Sigma^*$ to Q defined as: $\delta^*(q, w) = q'$ if there is a sequence of states $r_0, \ldots, r_n$ in Q such that
  - $r_0 = q$,
  - $r_i = \delta(r_{i-1}, w_i)$ for every $1 \leq i \leq n$,
  - $r_n = q'$

- Equivalently, there is a walk in the transition diagram of M from q to q' labelled by w.



Formal definition of computation

- Let $w_1 w_2 \cdots w_n$ be a string in Σ^*.
- $M = (Q, \Sigma, \delta, q_0, F)$ accepts w if
  - $\delta^*(q_0, w) \in F$, or equivalently
  - In the transition diagram of M, there is an walk from q_0 to an accept state labelled by w.



Language recognized by DFA

Definition: language recognized by DFA

- Let $M = (Q, \Sigma, \delta, q_0, F)$ be a finite automata.
- A string $w \in \Sigma^*$ is accepted by M if
  - $\delta^*(q_0, w) \in F$, or equivalently
  - in the transition diagram of M, there is an walk from q_0 to an accept state labelled by w.
- Let $L(M)$ be the set of all strings which are accepted by M.
- A language A is said to be recognized by M if $A = L(M)$.

Regular Language

Regular language = recognized by some DFA

- A language L over a finite alphabet is said to be regular if there is a finite-state automaton M which recognizes L.

From Languages to DFA: examples

Show that the following language is regular.

- $L = \{$all 0,1-strings containing 01$\}$
- $L = \{$ all 0,1-strings containing exactly even numbers of 0's and 1's respectively $\}$.