Theory of Computation

L.EIC, 2nd Year

João M. P. Cardoso
Email:jmpc@acm.org

U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO
DEI DEPARTAMENTO DE ENGENHARIA INFORMÁTICA

Outline

- Sets, Alphabets, Strings, Languages
- Languages and Problems
- Concepts about Finite Automata (FAs)
- Deterministic Finite Automata (DFAs)
- Notion of regular languages
- Operations with FAs

Alphabet and String

- **Alphabet** (Σ) is a non-empty finite set of symbols:
  - $\Sigma = \{0, 1\}$, binary alphabet
  - $\Sigma = \{a, b, \ldots, z\}$, set of lower-case letters
  - Set of ASCII chars
- **String** is a finite sequence of symbols over an alphabet
  - 01101 is a string over $\Sigma = \{0, 1\}$
  - Empty string (ε) has zero occurrences of symbols
  - Length of a string is the number of symbols over Σ: $|01101| = 5$, $|\varepsilon| = 0$
  - Power of an alphabet Σ^k is the set of strings, with length k, over Σ
    - $\Sigma^0 = \{\varepsilon\}$
    - If $\Sigma = \{0, 1\}$ then $\Sigma^1 = \{0, 1\}$, $\Sigma^2 = \{00, 01, 10, 11\}$, $\Sigma^3 = \{000, 001, \ldots, 111\}$
    - Distinction between $\Sigma = \{0, 1\}$, set of symbols, and $\Sigma^1 = \{0, 1\}$, set of strings

Language

- The set of all strings over an alphabet Σ is denoted as Σ^*, the Kleene-star closure on Σ (* is known as the Kleene-star)
  - $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \ldots$
  - $\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \ldots$
  - $\Sigma^* = \Sigma^0 \cup \Sigma^+$
- **Language** L over an alphabet Σ is the subset of Σ^* ($L \subseteq \Sigma^*$)
- Examples of Languages:
  - Language of the strings with n 0s, followed by n 1s, and $n \geq 0$:
    - $\{\varepsilon$, 01, 0011, 000111, …}
  - Set of prime binary numbers
    - {10, 11, 101, 111, 1011, …}
  - Empty language:
    - $\varnothing$ or {}
  - Language with only the empty string:
    - $\{\varepsilon\}$

Problem

- Decide if a given string belongs to a language
  - Given $w \in \Sigma^*$ and $L \subseteq \Sigma^*$, $w \in L$?
- It is common to describe a language using a set constructor notation:
  - {w | w consists of an equal number of 0s and 1s}, i.e., Set of strings referred as w such that w….
  - {w | w is a program in C syntactically correct}
- Example: primality testing
  - $w \in L_p$?   Where w is a string with the binary representation of a number and L_p is the language that contains all the strings representing the prime numbers in binary

Language or a Problem?

- Problem in a common sense:
  - Request to calculate (e.g., calculator) or transform an input (e.g., compiler)
  - Usually, not a yes/no decision
- In the context of complexity study, defining a problem in terms of a language is adequate
  - It is of similar difficulty to either solve the decision or the problem
  - If it is as difficult as to decide if a string belongs to language L_X (set of valid strings in language X) than to translate programs in X to object code

> If it was not, we could execute the translator, and then decide if the string belongs to L_X according to the success of the translator to produce object code. The problem of the decision would be easier which contradicts the supposition (proof by contradiction).

Language or a Problem?

- Languages and problems are essentially the same thing!
- Any **Problem** can be converted to a **Language**, and vice-versa:
  - Problem: Determine if a number is prime
  - Language: given L = {p : p is prime}, verify if a number belongs to L

Finite Automata (FAs)

Deterministic Finite Automaton

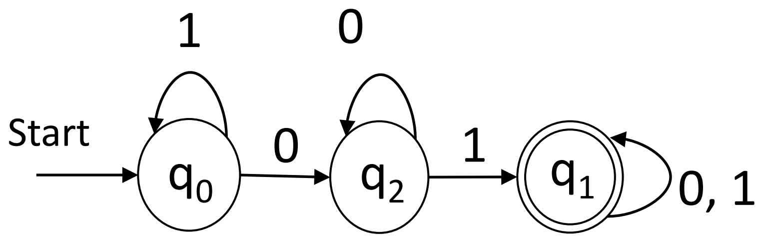
- a 5-tuple $(Q, \Sigma, \delta, q_0, F)$
  - a finite set of states Q
  - a finite set of input symbols called the alphabet Σ
  - a transition function $\delta : Q \times \Sigma \rightarrow Q$
  - an initial or start state $q_0 \in Q$
  - a set of accept states $F \subseteq Q$
- Deterministic: $\forall\ q \in Q, a \in \Sigma : |\delta(q, a)| \leq 1$ (or = 1 for complete DFAs)

Language of a DFA

- The language of a DFA $A = (Q, \Sigma, \delta, q_0, F)$ is the set of all the strings accepted/recognized by the DFA A
  - Input string: $a_1a_2\ldots a_n$
  - Initial state: q_0
  - First step: $\delta(q_0,a_1) = q_j$, $0 \le j \le n(Q)-1$
  - Step: $\delta(q_i,a_k) = q_j$, $0 \le i,j \le n(Q)-1$ and $2 \le k \le n$
  - If $\delta(q_i, a_n) \in F$, then the string is accepted

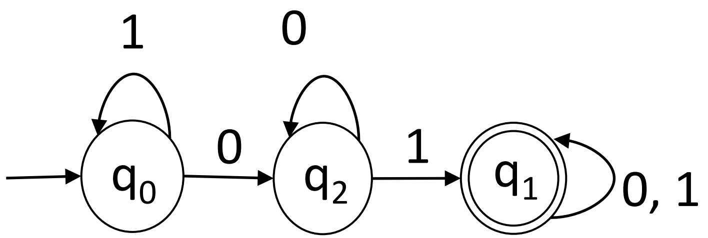
Defining a DFA: Example

- **Recognizer of the binary strings that contain the substring 01**
  - L = {x01y | x, y are empty strings or strings over the alphabet {0,1} }
  - Σ = {0,1}



The same as: L = {x01y | x and y $\in$ {0,1}*}

Defining a DFA: Example

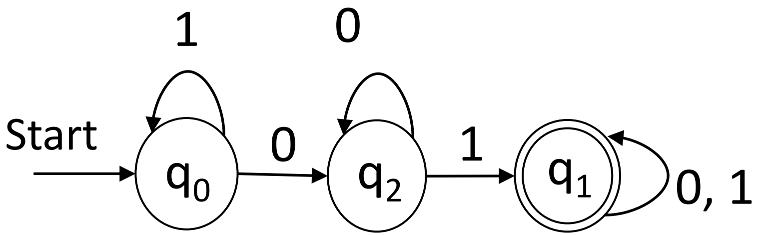


- **Recognizer of the binary strings that contain the substring 01**
  - $\Sigma = \{0,1\}$
  - $Q=\{q_0, q_1, q_2\}$ has to memorize if it has already seen 01 (q_1), if the last one was 0 (q_2), or if didn't see nothing relevant (q_0)
  - Start state: q_0
  - Transition function:
    - $\delta(q_0,1) = q_0$ $\delta(q_0,0) = q_2$ $\delta(q_2,0) = q_2$ $\delta(q_2,1) = q_1$ $\delta(q_1,0) = q_1$ $\delta(q_1,1) = q_1$
  - Final states: $\{q_1\}$
  - DFA $\mathbf{A = (Q, \Sigma , \delta, q_0, F)}$ = $(\{q_0, q_1, q_2\}, \{0,1\}, \delta, q_0, \{q_1\})$ = $(\{q_0, q_1, q_2\}, \{0,1\}, \{\delta(q_0,1) = q_0, \delta(q_0,0) = q_2 , \delta(q_2,0) = q_2 , \delta(q_2,1) = q_1 , \delta(q_1,0) = q_1 , \delta(q_1,1) = q_1\}, q_0, \{q_1\})$

Transition (state) Diagrams

- The transition diagram of a DFA A = (Q, Σ , δ, q_0, F) is a graph
  - State in Q $\Rightarrow$ node/vertex
  - δ(q, a) = p where q, p $\in$ Q and a $\in$ Σ $\Rightarrow$ edge from q to p with label a
  - Initial state $\Rightarrow$ arrow with Start (it is ok to omit the "Start" label)
  - States in F $\Rightarrow$ double circle in the node

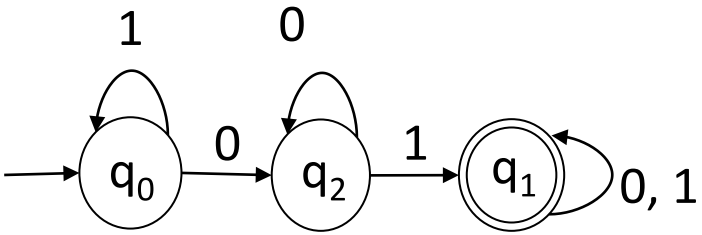
Example: recognizer of the binary strings that contain the substring 01



Transition Tables

Example: recognizer of the binary strings that contain the substring 01

- A transition table is the tabular representation of the δ function
  - States $\Rightarrow$ rows
  - Inputs $\Rightarrow$ columns
  - Initial State $\Rightarrow$ arrow
  - Final States $\Rightarrow$ *



	0	1
$\rightarrow q_0$	q_2	q_0
$*q_1$	q_1	q_1
q_2	q_2	q_1

Exercise 2

- Give a DFA that accepts the following language
  - L = { w | w has an even number of 0s and an even number of 1s}

Extended Transition Function $\hat{\delta}$

- Extended transition function, $\hat{\delta}(q, w) = p$
  - q: state
  - w: input string
  - p: reached state when we start in q and process w
- Inductive definition in $|w|$
  - Basis: $\hat{\delta}(q, \varepsilon) = q$
  - Induction: assuming **w=xa** then $\hat{\delta}(q, w) = \delta(\hat{\delta}(q, x), a)$
    - If $\hat{\delta}(q, x) = p$ and $\delta(p, a) = r$, to go from q to r, we go from q to p and then with a step to r
    - $\hat{\delta}(q, w) = \delta(p, a)$

“The Language of a DFA consists of the set of strings formed by the sequences of symbols for all the paths from the start node to each accept/final node”

Language of a DFA

- Processing in the DFA that recognizes strings with even number of 0s and 1s for the input w = 110101 (use DFA of Exercise 2)
  - $\hat{\delta}(q_0, \varepsilon) = q_0$
  - $\hat{\delta}(q_0, 1) = \delta(\hat{\delta}(q_0, \varepsilon), 1) = \delta(q_0, 1) = q_1$
  - $\hat{\delta}(q_0, 11) = \delta(\hat{\delta}(q_0, 1), 1) = \delta(q_1, 1) = q_0$
  - …
  - $\hat{\delta}(q_0, 110101) = \delta(\hat{\delta}(q_0, 11010), 1) = \delta(q_1, 1) = q_0$
- Language of a DFA $A = (Q, \Sigma, \delta, q_0, F)$ is
  - **$L(A) = \{w \mid \hat{\delta}(q_0, w) \in F\}$**
- If a language L is L(A) for a DFA A then it is a **regular language**

Exercise 3

- Give a DFA to recognize strings over the alphabet {0,1} with a ‘1’ in the third from last position.

Operations over Finite Automata

- Example of a Cartesian (cross) product
- Discuss the possible applications of the Cartesian product between finite automata

- Example (see slides “Operations Over FAs”):
  - DFA1 that recognizes: $\{w \in \{0,1\}^* \mid n_1(w) \text{ is even}\}$
  - DFA2 that recognizes: {x01y | x and y are strings of 0’s and 1’s}
  - What can give the Cartesian product of the two DFAs?

Summary

- Deterministic Finite Automata (DFAs)
- Use of DFAs to recognize strings
- Use of DFAs to represent regular languages
- Product of DFAs