Theory of Computation

MIEIC, 2nd Year

João M. P. Cardoso

U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Dep. de Engenharia Informática
Faculdade de Engenharia (FEUP)
Universidade do Porto
Porto
Portugal

Email: jmpc@acm.org

Outline

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

Concepts

- **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 selected from 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 occurrences of symbols: $|01101| = 5$, $|\varepsilon| = 0$
  - Power of an alphabet Σ^k is the set of string with length k, consisting of symbols of Σ (Cartesian product)
    - $\Sigma^0 = \{\varepsilon\}$
    - 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^*
  - $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \ldots$
  - $\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \ldots$
  - $\Sigma^* = \Sigma^0 \cup \Sigma^+$
- **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:
    - {ε, 01, 0011, 000111, …}
  - Set of binary prime numbers s
    - {10, 11, 101, 111, 1011, …}
  - Empty language:
    - ∅
  - Language with only the empty string:
    - {ε}

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}
    - 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 or transform an input (e.g., compiler)
  - Not a decision yes/no
- In the context of complexity study, defining a problem in terms of a language is adequate
  - It is of similar difficulty to solve the decision as the problem
  - If it is as difficult as to decide if a string belongs to language LX (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: L = {p : p is prime}

Finite Automata (FAs)

Example 1

(a) Let's think of an automaton that recognizes only strings over {1}

with an even number of 1s

Set constructor notation: {w | w $\in$ {1}* and |w| is even} or {w $\in$ {1}* | |w| is even}

(b) Let's think of an automaton that recognizes only strings over {0, 1}

with an even number of 1s

Set constructor notation: {w $\in$ {0,1}* | n_1(w) is even}

Deterministic Finite Automata (DFAs)

- Deterministic
  - In a state, for each input there is only a single possible transition
- A DFA is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where:
  - Q is the set of finite states
  - Σ is the set of finite input symbols (the alphabet)
  - δ is the transition function, of states and inputs to states (e.g., $p = \delta(q, a)$)
    - $\delta : Q \times \Sigma \to Q$
  - $q_0 \in Q$ is the start state
  - $F \subseteq Q$ is the set of final (accept) states
- DFA: $A = (Q, \Sigma, \delta, q_0, F)$

Tuple: a finite ordered list of elements

String Processing

- The language of a DFA A = (Q, Σ , δ, q_0, F) is the set of all the strings accepted/recognized by the DFA A
  - Input string: $a_1 a_2 \ldots a_n$
  - Initial state: q_0
  - Step: $\delta(q_{i-1}, a_i) = q_i$
  - If $q_n \in F$ then the string is accepted

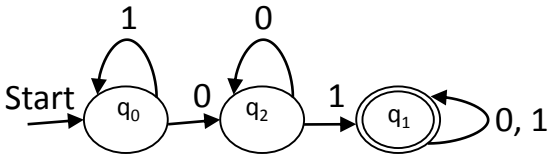
Defining an DFA

- Example: recognizer of the binary strings that contain the substring 01
  - {x 01y | x and y are strings of 0s and 1s}
  - $\Sigma = \{0,1\}$
  - $Q=\{q_0, q_1, q_2\}$ 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
    - $\delta(q_0,1) = q_0$ $\quad\quad$ $\delta(q_0,0) = q_2$
    - $\delta(q_2,0) = q_2$ $\quad\quad$ $\delta(q_2,1) = q_1$
    - $\delta(q_1,0) = q_1$ $\quad\quad$ $\delta(q_1,1) = q_1$
  - Final states: $\{q_1\}$
  - $A = (Q, \Sigma, \delta, q_0, F) = (\{q_0, q_1, q_2\}, \{0,1\}, \delta, q_0, \{q_1\})$

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

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
  - $\delta(q, a) = p$ where q, p $\in$ Q and a $\in \Sigma$ $\Rightarrow$ edge from q to p with label a
    - Edges from q to p can be merged and represented using a list of labels
  - Initial state $\Rightarrow$ arrow with Start (we will sometimes 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

- 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 Funtion $\hat{\delta}$

- Language of a DFA
  - Set of strings formed by the sequence of labels for all the paths from the start node to one of the accept nodes
- Extended transition function, $\hat{\delta}(q,w) = p$
  - q , state
  - w, input strings
  - 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
    - $\hat{\delta}(q,w) = \delta(p, a)$

Language of a DFA

- Processing in the DFA that recognizes strings with even number of 0s and 1s for the input w = 110101
  - $\hat{\delta}(q_0,\varepsilon) = q_0$
  - $\hat{\delta}(q_0,1) = \delta(\hat{\delta}(q_0, \varepsilon),1) = \delta(q_0,1) = q_1$
  - $\hat{\delta}(q_0,11) = \delta(\hat{\delta}(q_0,1),1) = \delta(q_1,1) = q_0$
  - …
  - $\hat{\delta}(q_0,110101) = \delta(\hat{\delta}(q_0,11010),1) = \delta(q_1,1) = q_0$
- 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 {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:
  - DFA1 that recognizes: $\{w \in \{0,1\}^* \mid n_1(w) \text{ is even}\}$
  - DFA2 that recognizes: {x 01y | 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