Chapter 3
Automata

Mathematical Modeling

(Materials drawn from this chapter in:
- Peter Linz. *An Introduction to Formal Languages and Automata*, (5th Ed.), Jones & Bartlett Learning, 2011.
- John E. Hopcroft, Rajeev Motwani and Jeffrey D. Ullamn. *Introduction to Automata Theory, Languages, and Computation* (3rd Ed.), Prentice Hall, 2006.
- Antal Iványi *Algorithms of Informatics*, Kempelen Farkas Hallgatói Információs Központ, 2011.)

HTNguyen, NAKhuong, LHTrang
Faculty of Computer Science and Engineering
University of Technology, VNU-HCM

Contents

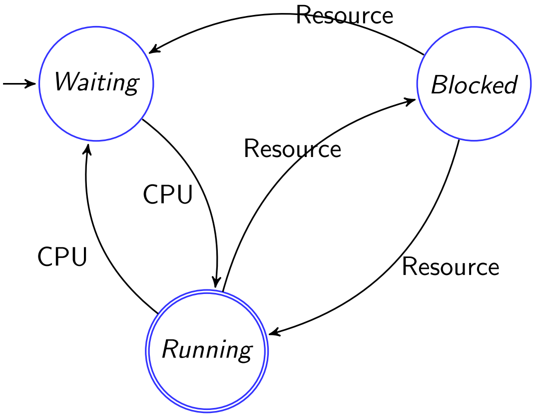
Course outcomes

	Course learning outcomes
L.O.1	Understanding of predicate logic
	L.O.1.1 – Give an example of predicate logic L.O.1.2 – Explain logic expression for some real problems L.O.1.3 – Describe logic expression for some real problems
L.O.2	Understanding of deterministic modeling using some discrete structures
	L.O.2.1 – Explain a linear programming (mathematical statement) L.O.2.2 – State some well-known discrete structures L.O.2.3 – Give a counter-example for a given model L.O.2.4 – Construct discrete model for a simple problem
L.O.3	Be able to compute solutions, parameters of models based on data
	L.O.3.1 – Compute/Determine optimal/feasible solutions of integer linear programming models, possibly utilizing adequate libraries L.O.3.2 – Compute/ optimize solution models based on automata, . . . , possibly utilizing adequate libraries

Introduction

Standard states of a process in operating system

- O with label: states
- $\rightarrow$: transitions



Why study automata theory?

A useful model
for many important kinds of software and hardware

1. designing and checking the behaviour of digital circuits
2. lexical analyser of a typical compiler: a compiler component that breaks the input text into logical units
3. scanning large bodies of text, such as collections of Web pages, to find occurrences of words, phrases or other patterns
4. verifying pratical systems of all types that have a finite number of distinct states, such as communications protocols of protocols for secure exchange information, etc.

Alphabets, symbols

Definition

Alphabet Σ (*bảng chữ cái*) is a finite and non-empty set of symbols (or characters).
For example:

- $\Sigma = \{a, b\}$
- The binary alphabet: $\Sigma = \{0, 1\}$
- The set of all lower-case letters: $\Sigma = \{a, b, \ldots, z\}$
- The set of all ASCII characters.

Remark

Σ is almost always all available characters (lowercase letters, capital letters, numbers, symbols and special characters such as space or newline).
But nothing prevents to imagine other sets.

Strings (words)

Definition

- A string/word u (*chuỗi/từ*) over Σ is a finite sequence (possibly empty) of symbols (or characters) in Σ.
- A empty string is denoted by ε.
- The length of the string u, denoted by $|u|$, is the number of characters.
- All the strings over Σ is denoted by Σ^*.
- A language L over Σ is a sub-set of Σ^*.

Remark

The purpose aims to analyze a string of Σ^* in order to know whether it belongs or not to L.

Example

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

- ε is a string with length of 0.
- 0 and 1 are the strings with length of 1.
- 00, 01, 10 and 11 are the strings with length of 2.
- $\emptyset$ is a language over Σ . It's called the empty language.
- Σ^* is a language over Σ . It's called the universal language.
- $\{\varepsilon\}$ is a language over Σ .
- $\{0, 00, 001\}$ is also a language over Σ .
- The set of strings which contain an odd number of 0 is a language over Σ.
- The set of strings that contain as many of 1 as 0 is a language over Σ.

String concatenation

Intuitively, the concatenation of two strings 01 and 10 is 0110. Concatenating the empty string ε and the string 110 is the string 110.

Definition

String concatenation is an application of $\Sigma^* \times \Sigma^*$ to Σ^*. Concatenation of two strings u and v in Σ is the string $u.v$.

Languages

Specifying languages

A language can be specified in several ways:

a) enumeration of its words, for example:

- $L_1 = \{\varepsilon, 0, 1\}$,
- $L_2 = \{a, aa, aaa, ab, ba\}$,
- $L_3 = \{\varepsilon, ab, aabb, aaabbb, aaaabbbb, \ldots\}$,

b) a property, such that all words of the language have this property but other words have not, for example:

- $L_4 = \{a^n b^n | n = 0, 1, 2, \ldots\}$,
- $L_5 = \{uu^{-1} | u \in \Sigma^*\}$ with $\Sigma = \{a, b\}$,
- $L_6 = \{u \in \{a, b\}^* | n_a(u) = n_b(u)\}$ where $n_a(u)$ denotes the number of letter 'a' in word u.

c) its grammar, for example:

- Let $G = (N, T, P, S)$ where
  $N = \{S\}$, $T = \{a, b\}$, $P = \{S \to aSb, S \to ab\}$
  i.e. $L(G) = \{a^n b^n | n \geq 1\}$ since
  $S \Rightarrow aSb \Rightarrow a^2Sb^2 \Rightarrow \ldots \Rightarrow a^nSb^n$

Operations on languages

L, L_1, L_2 are languages over Σ

- *union*

$$L_1 \cup L_2 = \{u \in \Sigma^* \mid u \in L_1 \ \text{ or } \ u \in L_2\},$$

- *intersection*

$$L_1 \cap L_2 = \{u \in \Sigma^* \mid u \in L_1 \ \text{ and } \ u \in L_2\},$$

- *difference*

$$L_1 \setminus L_2 = \{u \in \Sigma^* \mid u \in L_1 \ \text{ and } \ u \notin L_2\},$$

- *complement*

$$\overline{L} = \Sigma^* \setminus L,$$

- *multiplication*

$$L_1 L_2 = \{uv \mid u \in L_1, v \in L_2\},$$

- *power*

$$L^0 = \{\varepsilon\}, \qquad L^n = L^{n-1}L \text{ , if } n \geq 1,$$

- *iteration* or *star operation*

$$L^* = \bigcup_{i=0}^{\infty} L^i = L^0 \cup L \cup L^2 \cup \cdots \cup L^i \cup \cdots,$$

We will use also the notation L^+

$$L^+ = \bigcup_{i=1}^{\infty} L^i = L \cup L^2 \cup \cdots \cup L^i \cup \cdots .$$

The union, product and iteration are called regular operations.

Example

Let $\Sigma = \{a, b, c\}$, $L_1 = \{ab, aa, b\}$, $L_2 = \{b, ca, bac\}$

a) $L_1 \cup L_2 =?$ $L_1 \cup L_2 = \{ab, aa, b, ca, bac\}$,

b) $L_1 \cap L_2 =?$ $L_1 \cap L_2 = \{b\}$,

c) $L_1 \setminus L_2 =?$ $L_1 \setminus L_2 = \{ab, aa\}$,

d) $L_1L_2 =?$ $L_1L_2 = \{abb, aab, bb, abca, aaca, bca, abbac, aabac, bbac\}$,

e) $L_2L_1 =?$ $L_2L_1 = \{bab, baa, bb, caab, caaa, cab, bacab, bacaa, bacb\}$.

Let $\Sigma = \{a, b, c\}$ and $L = \{ab, aa, b, ca, bac\}$

$L^2 =?$ $L^2 = u.v$, with u, $v \in L$ including the following strings:

- $abab$, $abaa$, abb, $abca$, $abbac$,
- $aaab$, $aaaa$, aab, $aaca$, $aabac$,
- bab, baa, bb, bca, $bbac$,
- $caab$, $caaa$, cab, $caca$, $cabac$,
- $bacab$, $bacaa$, $bacb$, $bacca$, $bacbac$.

Regular expressions

Regular expressions (biểu thức chính quy)

Permit to specify a language with strings consist of letters and ε, parentheses $()$, operating symbols $+$, $.$, $*$. This string can be empty, denoted $\emptyset$.

Regular operations on the languages

- **union** $\cup$ or $+$
- **product of concatenation**
- **transitive closure** $*$

Example on the aphabet set $\Sigma = \{a, b\}$

- $(a+b)^*$ represent all the strings
- $a^*(ba^*)^*$ represent the same language
- $(a+b)^*aab$ represent all strings ending with aab.

Regular expressions

- $\emptyset$ is a regular expression representing the empty language.
- ε is a regular expression representing language $\{\varepsilon\}$.
- If $a \in \Sigma$, then a is a regular expression representing language $\{a\}$.
- If x, y are regular expressions representing languages X and Y respectively, then $(x + y)$, (xy), x^* are regular expression representing languages $X \bigcup Y$, XY and X^* respectively.

$$
\begin{aligned}
x + y &\equiv y + x \\
(x + y) + z &\equiv x + (y + z) \\
(xy)z &\equiv x(yz) \\
(x + y)z &\equiv xz + yz \\
x(y + z) &\equiv xy + xz \\
(x + y)^* \equiv (x^* + y)^* &\equiv (x + y^*)^* \equiv (x^* + y^*)^* \\
(x + y)^* &\equiv (x^*y^*)^* \\
(x^*)^* &\equiv x^* \\
x^*x &\equiv xx^* \\
xx^* + \varepsilon &\equiv x^*
\end{aligned}
$$

Regular expressions

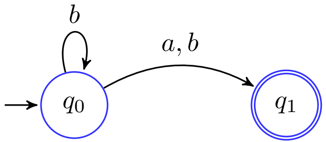
Kleene's theorem

Language $L \subseteq \Sigma^*$ is regular if and only if there exists a regular expression over Σ representing language L.

Finite automata

Finite automata (Automat hữu hạn)

- The aim is representation of a process system.
- It consists of states (including an initial state and one or several (or one) final/accepting states) and transitions (events).
- The number of states must be finite.

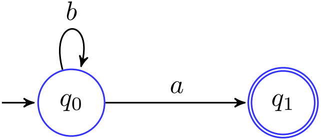


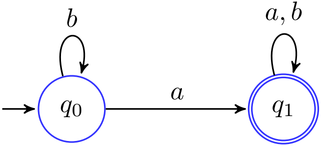
Regular expression

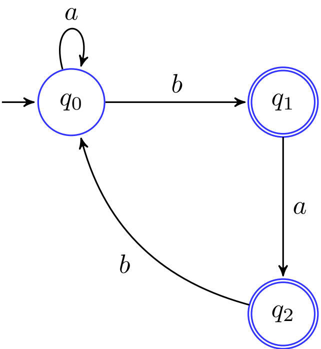
$b^*(a+b)$

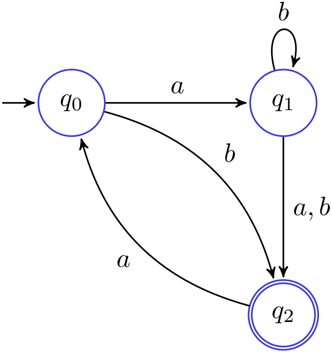
Exercise

Give regular expression for the following finite automata.









Nondeterministic finite automata

Definition

A nondeterministic finite automata (**NFA**, *Automat hữu hạn phi đơn định*) is mathematically represented by a 5-tuples $(Q, \Sigma, q_0, \delta, F)$ where

- Q a finite set of states.
- Σ is the alphabet of the automata.
- $q_0 \in Q$ is the initial state.
- $\delta : Q \times \Sigma \rightarrow Q$ is a transition function.
- $F \subseteq Q$ is the set of final/accepting states.

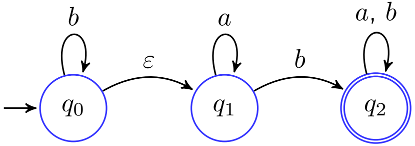
Remark

According to an event, a state may go to one or more states.

NFA with empty symbol ε

Other definition of NFA

Finite automaton with transitions defined by character x (in Σ) or empty character ε.



Deterministic finite automata

Definition

A deterministic finite automata (**DFA**, *Automat hữu hạn đơn định*) is given by a 5-tuplet $(Q, \Sigma, q_0, \delta, F)$ with

- Q a finite set of states.
- Σ is the input alphabet of the automata.
- $q_0 \in Q$ is the initial state.
- $\delta : Q \times \Sigma \to Q$ is a transition function.
- $F \subseteq Q$ is the set of final/accepting states.

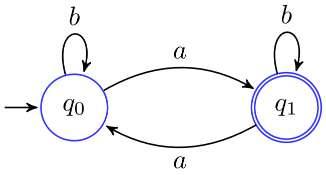
Condition

Transition function δ is an application.

Example

Let $\Sigma = \{a, b\}$

Hereinafter, a deterministic and complete automata that recognizes the set of strings which contain an odd number of a.



- $Q = \{q_0, q_1\}$,
- $\delta(q_0, a) = q_1$, $\delta(q_0, b) = q_0$, $\delta(q_1, a) = q_0$, $\delta(q_1, b) = q_1$,
- $F = \{q_1\}$.

Configurations and executions

Let $A = (Q, \Sigma, q_0, \delta, F)$

A configuration (*cấu hình*) of automata A is a couple (q, u) where $q \in Q$ and $u \in \Sigma^*$.
We define the relation $\rightarrow$ of derivation between configurations :
$(q, a.u) \rightarrow (q', u)$ iif $\delta(q, a) = q'$

An execution (*thực thi*) of automata A is a sequence of configurations
$(q_0, u_0) \ldots (q_n, u_n)$ such that
$(q_i, u_i) \rightarrow (q_{i+1}, u_{i+1})$, for $i = 0, 1, \ldots, n-1$.

Recognized languages

Một ngôn ngữ L trên một bảng chữ cái sigma, được định nghĩa là tập con của sigma*, là recognized nếu tồn lại một automata đơn định chấp nhận tất cả các chuỗi trong L

Definition

A language L over an alphabet Σ, defined as a sub-set of Σ^*, is recognized if there exists a finite automata accepting all strings of L.

Proposition

If L_1 and L_2 are two recognized languages, then

- $L_1 \cup L_2$ and $L_1 \cap L_2$ are also recognized;
- $L_1.L_2$ and L_1^* are also recognized.

Example

Sub-string ab

Construct a DFA that recognizes the language over the alphabet $\{a, b\}$ containing the sub-string ab.

Regular expression

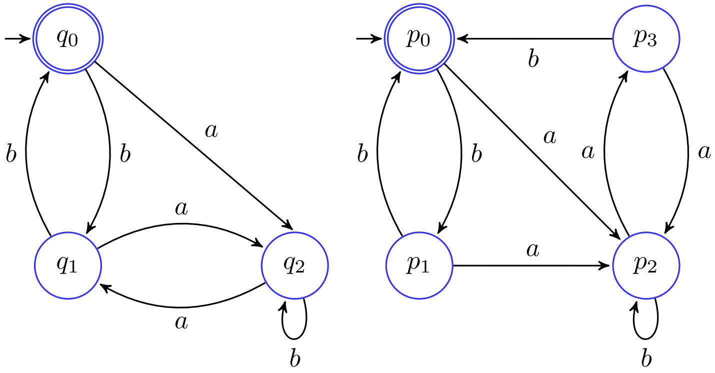
$(a+b)^*ab(a+b)^*$

Transition table

	a	b
$\rightarrow q_0$	q_1	q_0
q_1	q_1	q_2
q_2*	q_2	q_2

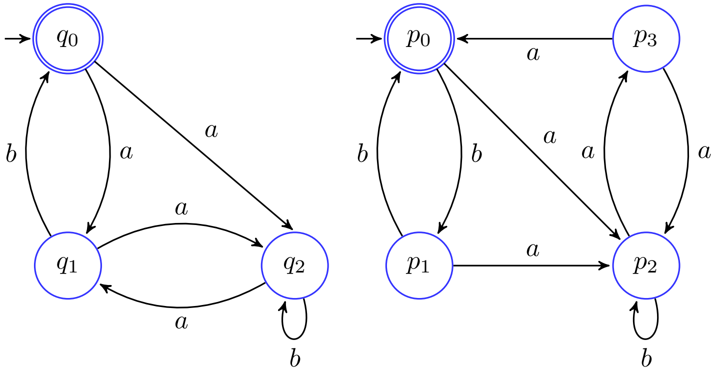
Equivalent automatons

Two following DFAs are equivalent?



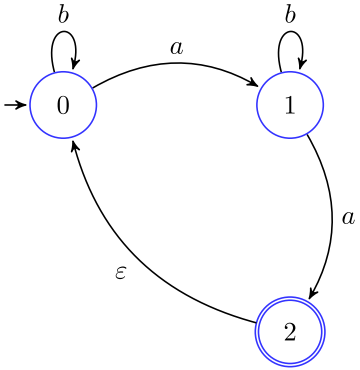
Equivalent automatons

Two following DFAs are equivalent?



From NFA to DFA

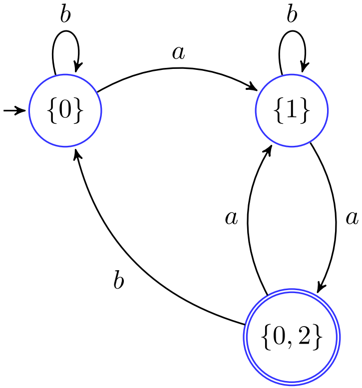
Given a NFA



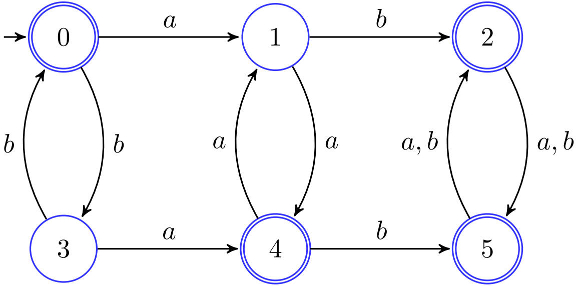
Transition table

	a	b
$\rightarrow \{0\}$	$\{1\}$	$\{0\}$
$\{1\}$	$\{0,2\}$	$\{1\}$
$\{0,2\}^*$	$\{1\}$	$\{0\}$

Corresponding DFA



From a DFA to a smaller DFA



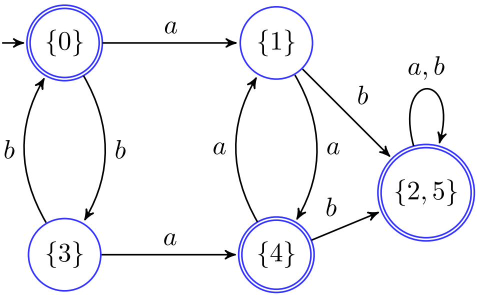
equivalence relationships

s	0	1	2	3	4	5
cl(s)	I	II	I	II	I	I
cl($s.a$)	II	I	I	I	II	I
cl($s.b$)	II	I	I	I	I	I

0	1	2	3	4	5
I	II	III	II	IV	III
II	IV	III	IV	II	III
II	III	III	I	III	III

s	0	1	2	3	4	5
cl(s)	I	II	III	V	IV	III
cl($s.a$)	II	IV	III	IV	II	III
cl($s.b$)	V	III	III	I	III	III

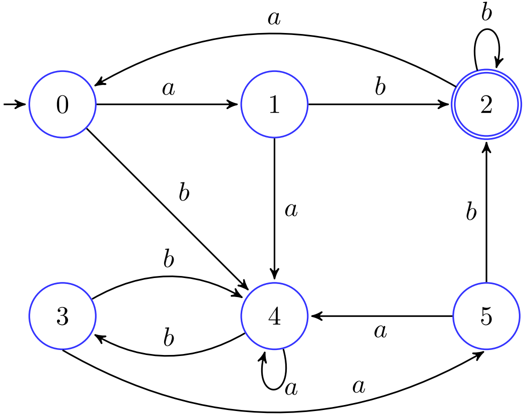
From a DFA to a smaller DFA



equivalence relationships

s	0	1	2	3	4	5
cl(s)	I	II	III	V	IV	III
cl($s.a$)	II	IV	III	IV	II	III
cl($s.b$)	V	III	III	I	III	III

Another example of minimization

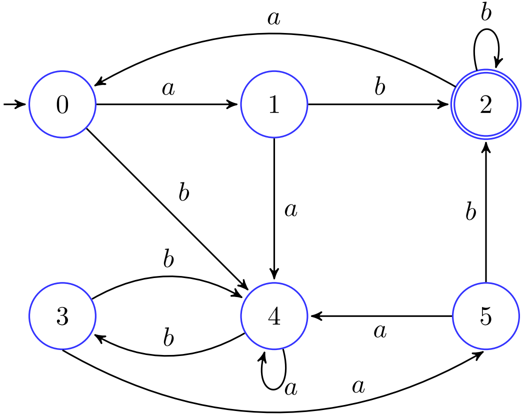


equivalence relationships

s	0	1	2	3	4	5
cl(s)	I	I	II	I	I	I
cl($s.a$)	I	I	I	I	I	I
cl($s.b$)	I	II	II	I	I	II

0	1	2	3	4	5
I	III	II	I	I	III
III	I	I	III	I	I
I	II	II	I	I	II

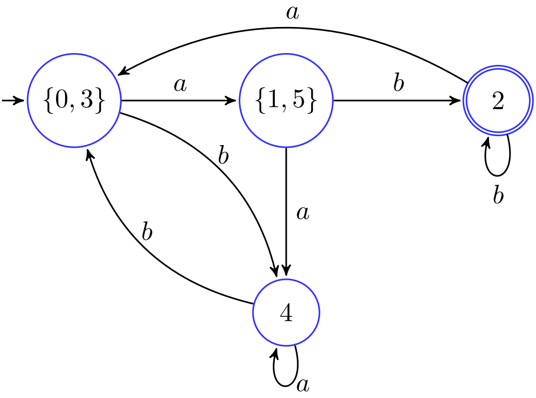
Another example of minimization



equivalence relationships

s	0	1	2	3	4	5	0	1	2	3	4	5
cl(s)	I	III	II	I	I	III	I	III	II	I	IV	III
cl($s.a$)	III	I	I	III	I	I	III	IV	I	III	IV	IV
cl($s.b$)	I	II	II	I	I	II	IV	II	II	IV	I	II

Another example of minimization



equivalence relationships

s	0	1	2	3	4	5
cl(s)	I	III	II	I	IV	III
cl($s.a$)	III	IV	I	III	IV	IV
cl($s.b$)	IV	II	II	IV	I	II