TUP-Springer Project

Renji Tao

有限自动机及在密码学中的应用

Finite Automata and Application to Cryptography

清华大学出版社

Springer

Renji Tao

有限自动机及在密码学中的应用

Finite Automata and Application to Cryptography

TSINGHUA UNIVERSITY PRESS
Springer

内 容 简 介

本书主要研究有限自动机的可逆性理论及其在密码学上的应用。此外，也讨论自治有限自动机和拉丁阵，它们与有限自动机单钥密码的标准形有关。

有限自动机是被认为是密码的自然模型。本书作者提出并发展了 $R_a\,R_b$ 变换方法，用它彻底解决了有限域上（拟）线性有限自动机的结构问题。与经典的线性系统“传输函数方法”不同，$R_a\,R_b$ 变换方法可推广到非线性有限自动机；大量弱可逆有限自动机及其弱逆可用它产生，这就导致基于有限自动机的公开钥密码（简记为 FAPKC）的提出。

本书可用作计算机科学和数学专业高年级和研究生课程的参考书。

图书在版编目(CIP)数据

有限自动机及在密码学中的应用：英文/陶仁骥著. 北京：清华大学出版社，2008.9
ISBN 978-7-302-17530-8

Ⅰ.有… Ⅱ.陶… Ⅲ.①有限自动机－理论－英文 ②有限自动机－应用－密码－理论－英文 Ⅳ.TP23 TN918.1

中国版本图书馆 CIP 数据核字(2008)第 063101 号

责任编辑：薛　慧
责任校对：刘玉霞
责任印制：孟凡玉

出版发行：清华大学出版社　　**地　址：**北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　编：**100084
社 总 机：010-62770175　　**邮　购：**010-62786544
投稿与读者服务：010-62776969, c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015, zhiliang@tup.tsinghua.edu.cn

印 刷 者：北京雅昌彩色印刷有限公司
经　　销：全国新华书店
开　　本：153×235　**印张：**26.25　**字数：**546 千字
版　　次：2008 年 9 月第 1 版　**印次：**2008 年 9 月第 1 次印刷
书　　号：ISBN 978-7-302-17530-8
ISBN 978-3-540-78256-8
e ISBN 978-3-540-78257-5
印　　数：1～2000
定　　价：98.00 元

Foreword

The important summarizing work of RENJI TAO appears now in book form. It is a great pleasure for me to see this happen, especially because I have known Professor Tao as one of the very early contributors to public-key cryptography. The research community has missed a book such as the present one now published by Tsinghua University Press and Springer. The book will be of special interest for students and researchers in the theories of finite automata, cryptography and error correcting codes.

One of the phenomena characterizing the second half of the last century is the rapid growth of computer science and informatics in general. The theory of finite automata, models of computing devices with a finite non-extensible memory, was initiated in the 1940s and 1950s, mainly by McCulloch, Pitts and Kleene. It has found numerous applications in most diverse areas, as exemplified by the series of yearly international conferences in implementation and applications of finite automata. The present work by Professor Tao develops a theory and contains strong results concerning invertible finite automata: the input sequence can be recovered from the output sequence. This is a desirable feature both in cryptography and error correcting codes. The book considers various types of invertibility and, for instance, the effect of bounded delay to invertibility.

Cryptography, secret writing, has grown enormously both in extent and importance and quality during the past few decades. This is obvious in view of the fact that so many transactions and so much confidential information are nowadays sent over the Internet. After the introduction of public-key cryptography by Diffie and Hellman in the 1970s, many devices were tried and applied for the construction of public-key cryptosystems. Professor Tao was one of such initiators in applying invertible finite automata. Although mostly in Chinese, his work was known also in the West. I referred to it already some twenty years ago. Later on, for instance, a PhD thesis was written about this topic in my university.

Many of the results in this book appear now for the first time in book form. The book systematizes important and essential results, as well as gives a comprehensive list of references. It can be used also as a starting point for further study. Different parts of the book are of varying importance for students and researchers, depending on their particular interests. Professor Tao gives useful guidelines about this in his Preface.

Much of the material in this book has not been previously available for western researchers. As a consequence, some of the results obtained by Professor Tao and his group already in the late 1970s have been independently rediscovered later. This concerns especially shift register sequences, for instance, the decimation sequence and the linear complexity of the product sequence.

I feel grateful and honored that Professor Tao has asked me to write this preface. I wish success for the book.

Turku, Finland, January 2008 *Arto Salomaa*

Preface

Automata theory is a mathematical theory to investigate behavior, structure and their relationship to discrete and digital systems such as algorithms, nerve nets, digital circuits, and so on. The first investigation of automata theory goes back to A. M. Turing in 1936 for the formulation of the informal idea of algorithms. Finite automata model the discrete and digital systems with finite "memory", for example, digital circuits. The theory of finite automata has received considerable attention and found applications in areas of computer, communication, automatic control, and biology, since the pioneering works of Kleene, Huffman, and Moore in the 1950s. Among others, autonomous finite automata including shift registers are used to generate pseudo-random sequences, and finite automata with invertibility are used to model encoders and decoders for error correcting and cipher as well as to solve topics in pure mathematics such as the Burnside problem for torsion groups. This book is devoted to the invertibility theory of finite automata and its application to cryptography. The book also focuses on autonomous finite automata and Latin arrays which are relative to the canonical form for one key cryptosystems based on finite automata.

After reviewing some basic concepts and notations on relation, function and graph, Chap. 1 gives the concept of finite automata, three types of "invertibility" for finite automata, and proves the equivalence between "feedforward invertibility" and "boundedness of decoding error propagation" which is the starting point of studying one key cryptosystems based on finite automata; a tool using labelled trees to represent states of finite automata is also given. In addition, some results on linear finite automata over finite fields are reviewed, in preparation for Chap. 7. Chapter 2 analyzes finite automata from the aspects of minimal output weight and input set. Results for weakly invertible finite automata are in return applied to establish the mutual invertibility for finite automata, and to evaluate complexity of searching an input given an output and an initial state for a kind of weakly invertible finite automata. In Chap. 3 the R_a R_b transformation method is presented for generating a kind of weakly invertible finite automata and correspondent weak inverse finite automata which are used in key generation in Chap. 9; this method is also used to solve the structure problem for quasi-linear finite automata over finite fields. Chapter 4 first discusses the relation between two linear R_a R_b transformation sequences and "composition" of R_a R_b trans-

formation sequences, then the relation of inversion by R_a R_b transformation method with inversion by reduced echelon matrix method and by canonical diagonal matrix polynomial method. Chapter 5 deals with the structure problem of feedforward inverse finite automata. Explicit expressions of feedforward inverse finite automata with delay $\leqslant 2$ are given. The result for delay 0 lays a foundation for the canonical form of one key cryptosystems based on finite automata in Chap. 8. In Chap. 6, for any given finite automaton which is invertible (weakly invertible, feedforward invertible, an inverse, or a weak inverse, respectively), the structure of all its inverses (weak inverses, weak inverses with bounded error propagation, original inverses, or original weak inverses, respectively) is characterized. Chapter 7 deals with autonomous linear finite automata over finite fields. Main topics contain representation of output sequences, translation, period, linearization, and decimation. The final two chapters discuss the application to cryptography. A canonical form for one key cryptosystems which can be implemented by finite automata without plaintext expansion and with bounded decoding error propagation is given in Chap. 8. As a component of the canonical form, the theory of Latin array is also dealt with. Chapter 9 gives a public key cryptosystem based finite automata and discusses its security. Some generalized cryptosystems are also given.

The material of this book is mainly taken from the works of our research group since the 1970s, except some basic results, for example, on linear finite automata and on partial finite automata. Of course, this book does not contain all important topics on invertibility of finite automata which our research group have investigated such as decomposition of finite automata and linear finite automata over finite rings. Results presented here other than Chaps. 1 and 7 are appearing for the first time in book form; Chapter 7 is appearing for the first time in English which is originally published in [97] and in Chap. 3 of the monograph [98]. This book is nearly self-contained, but algebra is required as a mathematical background in topics on linear finite automata, linear R_a R_b transformation, and Latin array; the reader is referred to, for example, [16], or [42] for matrix theory, [142] for finite group.

This book pursues precision in logic, which is extremely important for a mathematical theory. For automata theorists and other mathematicians interested merely in the invertibility theory of finite automata, the readers may read Chap. 1 to Chap. 6 and propose easily open problems on topics concerned. For an algebraist interested in the theory of shift register sequences, taking a glance at Chap. 7 is, at least to avoid overlap of research, harmless. A mathematician majoring in combinatory theory may be interested in Sects. 8.2 and 8.3 of Chap. 8.

For readers interested merely in one key cryptography, it is enough to read Chap. 1 (except Subsect. 1.2.3 and Sect. 1.6), the first two sections of Chap. 5, Chap. 7, and Chap. 8.

For readers interested merely in public key cryptography, they may read Chap. 1 (except Subsect. 1.2.4 and Sects. 1.3 and 1.5), Chap. 2 (except Sect. 2.2), Chap. 3 (except Sect. 3.3), Chap. 4, Sects. 6.1 and 6.5 of Chap. 6, and Chap. 9. They may skip over all proofs if they believe them to be correct; but a generation algorithm of finite automata satisfying the condition PI is directly obtained from several proofs in the first two sections of Chap. 3.

I would like to thank Zuliang Huang for his continuous encouragement and suggestions about the investigation on finite automata since the 1960s. Thanks also go to Peilin Yan, the late first director of Institute of Computing Technology, Chinese Academy of Sciences, and to Kongshi Xu, the first director of Institute of Software, Chinese Academy of Sciences, for their support and providing a suitable environment for me to do theoretical research since the 1970s. I am also grateful to many of my colleagues and students for various helpful discussions and valuable suggestions. My thanks go to Hongji Wang for his careful reading and commenting on the manuscript. Naturally, I have to take responsibility for any errors that may occur in this book. My special thanks go to Hui Xue for her continuing thorough and helpful editorial commentary, and careful polishing the manuscript. Finally, I thank my wife Shihua Chen and my daughter Xuemei Chen for their patience and continuous encouragement.

Beijing, May 2007 *Renji Tao*

Contents

Foreword by Arto Salomaa i

Preface iii

1. Introduction 1
- 1.1 Preliminaries 2
  - 1.1.1 Relations and Functions 2
  - 1.1.2 Graphs 5
- 1.2 Definitions of Finite Automata 6
  - 1.2.1 Finite Automata as Transducers 6
  - 1.2.2 Special Finite Automata 12
  - 1.2.3 Compound Finite Automata 14
  - 1.2.4 Finite Automata as Recognizers 16
- 1.3 Linear Finite Automata 16
- 1.4 Concepts on Invertibility 26
- 1.5 Error Propagation and Feedforward Invertibility 34
- 1.6 Labelled Trees as States of Finite Automata 41

2. Mutual Invertibility and Search 47
- 2.1 Minimal Output Weight and Input Set 48
- 2.2 Mutual Invertibility of Finite Automata 54
- 2.3 Find Input by Search 56
  - 2.3.1 On Output Set and Input Tree 56
  - 2.3.2 Exhausting Search 67
  - 2.3.3 Stochastic Search 74

3. R_a R_b Transformation Method 77
- 3.1 Sufficient Conditions and Inversion 78
- 3.2 Generation of Finite Automata with Invertibility 86
- 3.3 Invertibility of Quasi-Linear Finite Automata 95
  - 3.3.1 Decision Criteria 95
  - 3.3.2 Structure Problem 100

4. Relations Between Transformations 109
4.1 Relations Between R_a R_b Transformations 110
4.2 Composition of R_a R_b Transformations 115
4.3 Reduced Echelon Matrix 128
4.4 Canonical Diagonal Matrix Polynomial 132
4.4.1 R_a R_b Transformations over Matrix Polynomial 132
4.4.2 Relations Between R_a R_b Transformation and Canonical Diagonal Form 136
4.4.3 Relations of Right-Parts 139
4.4.4 Existence of Terminating R_a R_b Transformation Sequence 144

5. Structure of Feedforward Inverses 153
5.1 A Decision Criterion 154
5.2 Delay Free 157
5.3 One Step Delay 160
5.4 Two Step Delay 165

6. Some Topics on Structure Problem 177
6.1 Some Variants of Finite Automata 178
6.1.1 Partial Finite Automata 178
6.1.2 Nondeterministic Finite Automata 184
6.2 Inverses of a Finite Automaton 185
6.3 Original Inverses of a Finite Automaton 198
6.4 Weak Inverses of a Finite Automaton 201
6.5 Original Weak Inverses of a Finite Automaton 205
6.6 Weak Inverses with Bounded Error Propagation of a Finite Automaton 208

7. Linear Autonomous Finite Automata 215
7.1 Binomial Coefficient 216
7.2 Root Representation 224
7.3 Translation and Period 245
7.3.1 Shift Registers 245
7.3.2 Finite Automata 252
7.4 Linearization 254
7.5 Decimation 265

8. One Key Cryptosystems and Latin Arrays 273
8.1 Canonical Form for Finite Automaton One Key Cryptosystems 274
8.2 Latin Arrays 279
8.2.1 Definitions 279

8.2.2 On (n, k, r)-Latin Arrays 280
8.2.3 Invariant 284
8.2.4 Autotopism Group 288
8.2.5 The Case $n = 2, 3$ 291
8.2.6 The Case $n = 4$, $k \leqslant 4$ 294
8.3 Linearly Independent Latin Arrays 327
8.3.1 Latin Arrays of Invertible Functions 327
8.3.2 Generation of Linearly Independent Permutations 331

9. Finite Automaton Public Key Cryptosystems 347
9.1 Theoretical Fundamentals 348
9.2 Basic Algorithm 351
9.3 An Example of FAPKC 356
9.4 On Weak Keys 362
9.4.1 Linear R_a R_b Transformation Test 362
9.4.2 On Attack by Reduced Echelon Matrix 362
9.4.3 On Attack by Canonical Diagonal Matrix Polynomial 363
9.5 Security 364
9.5.1 Inversion by a General Method 365
9.5.2 Inversion by Decomposing Finite Automata 365
9.5.3 Chosen Plaintext Attack 366
9.5.4 Exhausting Search and Stochastic Search 367
9.6 Generalized Algorithms 372
9.6.1 Some Theoretical Results 372
9.6.2 Two Algorithms 387

References 395

Index 403

1. Introduction

Renji Tao

Institute of Software, Chinese Academy of Sciences
Beijing 100080, China trj@ios.ac.cn

Summary.

Finite automata are a mathematical abstraction of discrete and digital systems with finite "memory". From a behavior viewpoint, such a system is a transducer which transforms an input sequence to an output sequence with the same length. Whenever the input sequence can be retrieved by the output sequence (and initial internal state), the system is with invertibility and may be used as an encoder in application to cipher or error correcting.

The invertibility theory of finite automata is dealt within the first six chapters of this book. In the first chapter, the basic concepts on finite automata are introduced. The existence of (weak) inverse finite automata and boundedness of delay for (weakly) invertible finite automata are proven in Sect. 1.4, and the coincidence between feedforward invertibility and bounded error propagation is presented in Sect. 1.5. In Sect. 1.7, we characterize the structure of (weakly) invertible finite automata by means of their state tree. In addition, there is a section that reviews some basic results of linear finite automata, as an introduction to Chap. 7.

Key words: *finite automata, invertible, weakly invertible, feedforward invertible, inverse, weak inverse, feedforward inverse, error propagation, state tree*

Finite automata are a mathematical abstraction of discrete and digital systems with finite "memory". From a structural viewpoint, such a system has an input and an output as well as an "internal state". Its time system is discrete (say, moments $0, 1, \ldots$). Only finite possible values can be taken by the input (output and internal state, respectively) at each moment. And, the output at the current moment and the internal state at the next moment can be uniquely determined by the input and the internal state at the current moment. From a behavior viewpoint, such a system is a transducer which transforms an input sequence to an output sequence with the same length.

Whenever the input sequence can be retrieved by the output sequence (and the initial internal state), the system is with invertibility and may be used as an encoder in application to cipher or error correcting.

The invertibility theory of finite automata is dealt within the first six chapters. In the first chapter, the basic concepts on finite automata are introduced. The existence of (weak) inverse finite automata and boundedness of delay for (weakly) invertible finite automata are proven in Sect. 1.4, and the coincidence between feedforward invertibility and bounded error propagation is presented in Sect. 1.5. In Sect. 1.6, we characterize the structure of (weakly) invertible finite automata by means of their state tree. In addition, there is a section that reviews some basic results of linear finite automata, as an introduction to Chap. 7.

1.1 Preliminaries

We begin with a brief excursion through some fundamental concepts. A reader acquainted with the notation used may skip this section. We will assume a familiarity with the most basic notions of set theory, such as membership $\in$, set-builder notation $\{\cdots \mid \cdots\}$ or $\{\cdots : \cdots\}$, empty set $\varnothing$, subset $\subseteq$, union $\cup$, intersection $\cap$, difference $\setminus$.

1.1.1 Relations and Functions

For any sets $A_1, A_2, \ldots, A_n$, the *Cartesian product* of $A_1, A_2, \ldots, A_n$ is the set

$$\{(a_1, a_2, \ldots, a_n) \mid a_i \in A_i, i = 1, 2, \ldots, n\},$$

denoted by $A_1 \times A_2 \times \cdots \times A_n$ (sometimes $(a_1, a_2, \ldots, a_n)$ is replaced by $\langle a_1, a_2, \ldots, a_n \rangle$). In the case of $A_i = A, i = 1, 2, \ldots, n$, $A_1 \times A_2 \times \cdots \times A_n$ is called the n-fold Cartesian product of A and is abbreviated to A^n. For any $i, 1 \leqslant i \leqslant n$, the i-th component of an element $(a_1, a_2, \ldots, a_n)$ in $A_1 \times A_2 \times \cdots \times A_n$ means a_i.

Let A and B be two sets. A *relation* R from A to B is a subset R of $A \times B$. If (a, b) is in the relation R, it is written as aRb. If (a, b) is not in the relation R, it is written as $a \not R b$. In the case of $A = B$, R is also called a *relation* on A.

A relation R on a set A is an *equivalence relation*, if the following conditions hold: (a) R is reflexive, i.e., $(a, a) \in R$ for any a in A; (b) R is symmetric, i.e., $(a, b) \in R$ implies $(b, a) \in R$ for any a and b in A; and (c) R is transitive, i.e., $(a, b) \in R$ and $(b, c) \in R$ imply $(a, c) \in R$ for any a, b and c in A.

Let R be an equivalence relation on A. For any a in A, the set $[a]_R = \{b \mid b \in A, (a,b) \in R\}$ is called the *equivalence class* containing a. The set $\{[a]_R \mid a \in A\}$ is called the *equivalence classes* of R.

Let A be a set and $\pi = \{H_i \mid i \in I\}$ be a family of subsets of A. If (a) $\cup_{i \in I} H_i = A$ and (b) $H_i \cap H_j = \varnothing$ for any different i and j in I, π is called a *partition* of A, and H_i, $i \in I$ are called *blocks* of the partition π.

Clearly, the equivalence classes of an equivalence relation on A define a partition. Conversely, a partition $\{H_i \mid i \in I\}$ of A determines an equivalence relation R on A in the following way:

$$(a,b) \in R \Leftrightarrow \exists i \in I(a \in H_i \ \& \ b \in H_i),$$
$$a, b \in A.$$

It is convenient to identify an equivalence relation with its partition.

Let R be a relation from A to B. The subset

$$\{a \in A \mid \exists b \in B((a,b) \in R)\}$$

of A is called the *domain* of R, and the subset

$$\{b \in B \mid \exists a \in A((a,b) \in R)\}$$

of B is called the *range* of R.

Suppose that R is a relation from A to B. Define a relation R^{-1} from B to A as follows:

$$(a,b) \in R^{-1} \Leftrightarrow (b,a) \in R,$$
$$a \in A, b \in B.$$

R^{-1} is called the *inverse relation* of R. Clearly, the domain of R and the range of R^{-1} are the same; the domain of R^{-1} and the range of R are the same.

Let R be a relation from A to B. If, for any a in A, any b and b' in B, $(a,b) \in R$ and $(a,b') \in R$ imply $b = b'$, R is called a *partial function* from A to B.

A single-valued *function* (*mapping*) from A to B is a partial function R from A to B such that the domain of R is A. A single-valued function from a set to itself is also called a *function* or a *transformation* on the set.

Let f be a single-valued mapping or partial function from A to B. For any a in the domain of f, the unique element in B, say b, satisfying $(a,b) \in f$ is written as $f(a)$, and is called the *value* of f at (the point) a. For any a not in the domain of f, we say that the value of f at (the point) a is *undefined.* For any relation R from A to B and any a in A, we also use $R(a)$ to denote the set $\{b \in B \mid (a,b) \in R\}$. Clearly, $R^{-1}(b) = \{a \in A \mid (b,a) \in R^{-1}\} = \{a \in A \mid (a,b) \in R\}$ for any b in B.

Let f be a single-valued mapping from A to B. If the range of f is B, f is called a *surjection*, or to be *surjective*, or a single-valued mapping from A *onto* B. If $f(a) \neq f(a')$ holds for any different elements a and a' in A, f is called an *injection*, or to be *injective*, or to be *one-to-one*. If f is injective and surjective, f is called a *bijection*, or to be *bijective*, or a *one-to-one* mapping from A *onto* B. If there exists a one-to-one mapping from A onto B, A is said to be *one-to-one correspondent* with B. A bijection from a finite set to itself is also called a *permutation* on the set, or of its elements.

If f is a partial or single-valued function from A to B, the inverse relation f^{-1} is also called *the inverse function* of f. Thus $f^{-1}(b) = \{a \in A \mid (a, b) \in f\} = \{a \in A \mid f(a) = b\}$. For any b in B, whenever $|f^{-1}(b)| = 1$, we also use $f^{-1}(b)$ to denote the unique element, say a, in $f^{-1}(b)$, where $f(a) = b$; from the context, the reader can easily understand the meaning of the notation without ambiguity. It is easy to see that if f is a bijection from A to B, then f^{-1} is a bijection from B to A and $f^{-1}(f(a)) = a$ for any $a \in A$, $f(f^{-1}(b)) = b$ for any $b \in B$.

An injection f from A to B is also called an *invertible* function, or an *invertible* transformation in the case of $A = B$; a partial or single-valued function g from B to A is called *an inverse function*, or *an inverse transformation* in the case of $A = B$, of f, if $g(f(a)) = a$ holds for any $a \in A$. For any partial or single-valued function f_i from A_i to B_i, $i = 1, 2$, if $A_2 \subseteq A_1$, $B_2 \subseteq B_1$ and $f_1(a) = f_2(a)$ for any $a \in A_2$, f_2 is called a *restriction* of f_1 (on A_2). We use $f_1|_{A_2}$ to denote a restriction of f_1 on A_2. Clearly, if g is an inverse function of f, then the inverse function f^{-1} of f is a restriction of g. We also use f^{-1} to denote an inverse function of f; from the context, the reader can easily understand the meaning of the notation without ambiguity.

A *vector function* of dimension n in s variables over F means a single-valued function from the s-fold Cartesian product of F (respectively an s-dimensional vector space over F) to the n-fold Cartesian product of F (respectively an n-dimensional vector space over F). For a vector function φ of dimension n in s variables over F, its value at the point $(x_1, \ldots, x_s)$ is usually expressed as $\varphi(x_1, \ldots, x_s)$; for any i, $1 \leqslant i \leqslant n$, the i-th component function of φ is a single-valued function from the s-fold Cartesian product of F (respectively an s-dimensional vector space over F) to F of which the value at each point $(x_1, \ldots, x_s)$ is the i-th component of $\varphi(x_1, \ldots, x_s)$. A vector function over $\{0, 1\}$ is called a *Boolean vector function*. A Boolean function means a Boolean vector function of dimension 1. A Boolean function $\varphi(x_1, \ldots, x_s)$ in s variables can be expressed by a polynomial of $x_1, \ldots, x_s$; if the degree of the polynomial is greater than 1, φ is said to be *nonlinear*. The Boolean function in s variables of which all values are 0 is called the *zero Boolean*

function in s variables. The function on A of which the value at each a in A equals a is called the *identity function* on A.

1.1.2 Graphs

We will discuss some fundamental concepts of graph. (V, Γ) is called a *(directed) graph*, if $\Gamma \subseteq V \times V$ for a nonempty set V. V is called the *vertex set*, and elements in V are called *vertices*. Γ is called the *arc set* or the *directed edge set*, and elements in Γ are called *arcs* or *directed edges*. For an arc $u = (a, b) \in \Gamma$, a is called the *initial vertex* of u, and b the *terminal vertex* of u.

Let $w = u_1 u_2 \dots u_i \dots$ be a finite or infinite sequence of arcs, where $u_i \in \Gamma$, $i = 1, 2, \dots$ If the terminal vertex of u_i is the initial vertex of u_{i+1} for any u_i, u_{i+1} in w, w is called a *path* of the graph (V, Γ). The number of arcs in w is called the *length* of the path w. The initial vertex of u_1 is called the *initial vertex* of the path w; and the terminal vertex of u_n is called the *terminal vertex* of the path w if the length of the path w is n.

If $w = u_1 u_2 \dots u_n$ is a path of the graph (V, Γ) and the terminal vertex of the arc u_n is the initial vertex of the arc u_1, the path w is called a *circuit* of the graph (V, Γ). Evidently, if there exists a circuit, then there exists a path of infinite length.

For any vertex a, the set $\{b|(b, a) \in \Gamma, b \in V\}$ is called the *incoming vertex set* of a, and the set $\{b|(a, b) \in \Gamma, b \in V\}$ is called the *outgoing vertex set* of a.

A vertex of which both the incoming vertex set and the outgoing vertex set are empty is called an *isolated vertex*.

We define recurrently the levels of vertices as follows. For any vertex a in V, if the incoming vertex set of a is empty, the *level* of a is defined to be 0. For any vertex a in V, if the levels of all vertices in the incoming vertex set of a have been defined and the maximum is h, the *level* of a is defined to be $h + 1$.

For any arc $u = (a, b)$, if levels of a and b have been defined, the *level* of the arc u is defined to be the level of the vertex a.

If the level of each vertex of (V, Γ) is defined and the maximum is h, we say that the graph *has level*, and the *level* of the graph is defined to be $h - 1$.

Clearly, if each vertex of (V, Γ) is an isolated vertex, then the level of the graph is -1.

If V is finite, the graph (V, Γ) is said to be *finite*.

Notice that for a finite graph, it has no circuit if and only if it has level, and the maximum of its path-lengths equals its level plus 1 if it has level.

It is convenient for some applications to introduce the *empty graph*. The vertex set and the arc set of the empty graph can be regarded as the empty set. The level of the empty graph is defined to be -2.

Two graphs (V, Γ) and (V', Γ') are said to be *isomorphic*, if there exists a one-to-one mapping φ from V onto V' such that (a, b) is an arc of (V, Γ) if and only if $(\varphi(a), \varphi(b))$ is an arc of (V', Γ'). Any such mapping φ is called an *isomorphism* from (V, Γ) to (V', Γ'). An isomorphism from a graph to itself is called an *automorphism* of the graph.

A graph (V', Γ') is called a *subgraph* of a graph (V, Γ), if $V' \subseteq V$ and $\Gamma' \subseteq \Gamma$.

A graph (V, Γ) is called a *tree* with root v, if the following conditions hold: (a) each vertex $(\neq v)$ is a terminal vertex of a unique arc; (b) v is not a terminal vertex of any arc; and (c) (V, Γ) has no circuit.

A vertex of a tree is called a *leaf*, if no arc emits from the vertex, i.e., the outgoing vertex set of the vertex is empty.

Let (V, Γ) and (V', Γ') be two trees. If (V', Γ') is a subgraph of (V, Γ), (V', Γ') is called a *subtree* of (V, Γ).

Let G be a (directed) graph (respectively tree). If an element of some set is assigned to each arc of G, or if an element of some set is assigned to each arc of G and an element of some set is assigned to each vertex of G, G is called a *labelled graph* (respectively *labelled tree*). The element assigned to an arc (respectively a vertex) is referred to as the arc (respectively vertex) label of the arc (respectively vertex).

1.2 Definitions of Finite Automata

1.2.1 Finite Automata as Transducers

For any set A, the concatenation of elements in A, say $a_0 a_1 \ldots a_{l-1}$, is called a *word* (or a *finite sequence*) over A, and l its *length*, where $a_0, a_1, \ldots, a_{l-1}$ are elements in A. In the case of $l = 0$, $a_0 a_1 \ldots a_{l-1}$ is a void sequence which contains no element. The void sequence is called the *empty word* and its length is 0. We use ε to denote the empty word (void sequence), and $|\alpha|$ the length of a word α. The set of all the words over A including the empty word is denoted by A^*. If $a_0, a_1, \ldots, a_n, \ldots$ are elements in A, the concatenation of the infinite elements $a_0 a_1 \ldots a_n \ldots$ is called an *infinite-length word* or an *ω-word* (or an *infinite sequence*) over A. We use A^ω to denote the set of all infinite-length words over A. We also use A^n to denote the set of all words over A of length n for any nonnegative integer n.

Let $\alpha = a_0 a_1 \ldots a_{m-1}$ and $\beta = b_0 b_1 \ldots b_{n-1}$ be two words in A^*. The concatenation of α and β is $a_0 a_1 \ldots a_{m-1} b_0 b_1 \ldots b_{n-1}$, which is also a word in