Finite Automata

Behavior and Synthesis

B. A. TRAKHTENBROT
Institute of Mathematics, Novosibirsk

and

YA. M. BARZDIN'
Riga

Translated from the Russian by
D. LOUVISH
Israel Program for Scientific Translations

English translation edited by
E. SHAMIR
Hebrew University, Jerusalem

and

L. H. LANDWEBER
University of Wisconsin, Madison

1973

NORTH-HOLLAND PUBLISHING COMPANY–AMSTERDAM · LONDON
AMERICAN ELSEVIER PUBLISHING COMPANY, INC.–NEW YORK

Library of Congress Catalog Card Number: 72-88504

North-Holland ISBN: 0 7204 8021 3

American Elsevier ISBN: 0 444 10418 6

Publishers:

NORTH-HOLLAND PUBLISHING COMPANY – AMSTERDAM

NORTH-HOLLAND PUBLISHING COMPANY, LTD. – LONDON

Sole Distributors for U.S.A. and Canada:

AMERICAN ELSEVIER PUBLISHING COMPANY, INC.

52 VANDERBILT AVENUE

NEW YORK, N.Y. 10017

PRINTED IN ISRAEL

PREFACE

This book is devoted to the behavioral theory (or the abstract theory) of automata, in which the behavior of the automaton is divorced, as far as possible, from its constructional features. In this context the term "synthesis of an automaton" means the construction of a program (rather than a circuit diagram, as is the case in the structural theory of automata).

A large part of the book (Chapter 0—Introduction, Chapters I, II and V) is devoted to various aspects of the behavior of automata: the representation of languages and ω-languages, the realization of operators, the description and estimation of various behavioral parameters and spectra (= sequences of parameters). Synthesis proper is discussed only in Chapters III and IV. Since all the requisite auxiliary facts, even certain components of the basic algorithms, will have been presented in sufficient detail beforehand, the exposition in these chapters is relatively concise. An alternative classification of the material might assign the first three chapters to the traditional approach in automata theory, Chapters IV and V to the statistical approach. The first, rather arbitrarily named, traditional approach deals with rules, algorithms and constructions relating to all automata, while the statistical (frequency) approach discusses principles which, though not valid for all automata, are nevertheless frequently encountered.

Recent years have seen the publication of several monographs and review articles containing a wealth of material on the theory of automata in general and the behavioral theory in particular. It should be clear from the table of contents that this book is quite different from its predecessors. A more detailed survey of the contents may be found in Section 0.5.

The senior author (B. A. Trakhtenbrot), whose systematic work in automata theory began over ten years ago, investigated the synthesis problem for automata whose initial specification is formulated in the language of predicate logic. Concurrently (and independently), similar work was being done in the U.S. by Church and (somewhat later) by Büchi. Despite considerable accomplishments, the problem in its most general and natural formulation (cf. the logical metalanguage in Chapter III) remained open.

Only quite recently Büchi and Landweber, using a game-theoretic interpretation suggested by McNaughton, established results that lead, in a certain sense, to a definitive theory of behavior and synthesis for finite automata. These results also provide a unified treatment of many previously known facts, presenting them in a compact and, thanks to the game-theoretic interpretation, lucid manner. This is done in Chapters I to III of the present book, which come under the heading of the "traditional" approach. We present existence proofs and descriptions of algorithms for the construction of finite automata. Some of these algorithms (even the most important of them) are prohibitively complex and, in this form, quite impracticable. Though this might seem rather disappointing, one must remember that the very existence of these algorithms is far from trivial. This will be borne out by examples showing that, under seemingly minor and harmless modifications of the problem, there exists no solving algorithm at all. In other words, the situations studied in this book lie at the border of the no man's land in regard to the existence of an algorithm. Yet the algorithms described may be used as a starting point for more practical procedures.

In spirit, the "statistical" portion of the book approaches the theory of experiments whose foundations were laid as early as 1956 by Moore. Moore proved, in particular, that the behavior of an automaton with k states can be reconstructed by a multiple experiment of length $2k - 1$. Trakhtenbrot, who established the same result independently, also pointed out that "in the majority of cases" the so-called degree of reconstructibility is much smaller than $2k - 1$; he conjectured that it was of the same order as $\log k$. It became important to verify this conjecture for complexity estimates of the synthesis process in machine identification, when the designer augments his information about the projected automaton by appropriate interrogations of the customer (in so doing, the designer, so to speak, experiments with a "black box" and tries to guess at its behavior). Only in recent papers of Barzdin' and Korshunov was this conjecture proved. Barzdin' also proposed the idea of a frequency algorithm for synthesis and identification: the only requirement from the algorithm is that it produce correct results with a certain prescribed frequency. In particular, it proves possible to construct frequency algorithms which identify "most" automata, using only such information as can be gained by applying input words and observing the corresponding output words (with no upper bound on the number of states). A description of frequency algorithms and an estimate of their complexity, using the most probable values for the behavioral parameters of the automaton, is the topic of Chapters IV and V.

On the whole, one might say that Chapters I to III summarize the "old" parts of the theory, while Chapters IV and V represent the first encouraging steps of a new trend, which we have called "statistical."

Though the book deals with finite automata, wherever finiteness is inessential the exposition also includes the case of infinite automata.

The most frequently treated case in the literature is that of the finite behavior of automata, corresponding to the reception of finite (though arbitrarily long!) sequences of input signals. We shall also devote much attention to infinite behavior, corresponding to an idealized situation in which the automaton operates for an infinitely long time, receiving infinite sequences of input symbols. McNaughton has shown that abstract infinite behavor enables one to make use of certain highly efficient "limit" criteria and proves to be extremely fruitful.

Our book does not claim to present all achievements to date in the behavioral theory of automata. It omits many facts and procedures, relating both to theory and, especially, to engineering practice, which have received excellent and detailed coverage in the available monograph literature (thus, for example, minimization of automata is touched upon only in passing). We have endeavored to compensate the reader for this in the supplementary material and problems at the end of each chapter.

The main text contains no bibliographic references. These are given in the Notes at the end of each chapter; the Notes also provide other historical and bibliographic data.

The Introduction and Chapters I through III, written by Trakhtenbrot, constitute a revised version of his lectures at Novosibirsk University during the spring semester of 1966. Chapters IV and V, written by Barzdin', contain both his own results and results obtained with A. D. Korshunov and M. P. Vasilevskii. The material has been discussed in seminars on automata theory at Novosibirsk and the Latvian State University.

During our work on the book we were assisted by many individuals. Z. K. Litvintseva placed her lecture notes at our disposal and, together with N. G. Shcherbakova, helped to put them in order. We received very helpful remarks and advice from Yu. I. Lyubich, G. S. Plesnevich, A. D. Korshunov, V. A. Nepomnyaschii and M. P. Vasilevskii. The considerable task of editing the authors' manuscript was undertaken by B. Yu. Pil'chak and N. A. Karpova. We are deeply indepted to all these colleagues.

B. Trakhtenbrot
Ya. Barzdin'

CHAPTER 0

INTRODUCTION

0.1. The concept of an automaton

The automata studied in this book are in effect mechanisms consisting of a control block capable of assuming various states (the so-called internal states of the automaton), an input channel and an output channel. The input channel receives (reads) input signals from the environment, while the output channel sends output signals to the environment. The nature of the states and the signals is immaterial; they may be regarded as certain symbols (letters), which make up a state alphabet (or internal alphabet) Q, an input alphabet X and an output alphabet Y, respectively. The alphabets X and Y are always assumed finite, Q at most denumerable. The automaton functions at discrete instants of time $t = 1, 2, 3, \ldots$, called sampling times, according to a definite program or, what is the same, system of instructions. Each instruction may be written in the form

$$q_i x_r \rightarrow q_j y_s,$$

where q_i, q_j are internal states, x_r an input symbol and y_s an output symbol. It is assumed that the program does not contain different instructions $q_i x_r \rightarrow q_j y_s$, $q_i x_r \rightarrow q_v y_\tau$ with identical left-hand sides and different right-hand sides (uniqueness condition); however, the program need not contain an instruction with left-hand side $q_i x_r$ for every such pair.

Assume that at some sampling time t_0 the control block is in state q_i and the input channel receives a symbol x_r. If the program contains an instruction with left-hand side $q_i x_r$, say $q_i x_r \rightarrow q_j y_s$, the output channel emits the symbol y_s at the same time t_0 and, at the following sampling time $t_0 + 1$, the control block passes into state q_j. But if the program contains no such instruction (the pair $q_i x_r$ is forbidden), the automaton is blocked, it makes no response to the symbol received at the instant t_0, and also stops receiving symbols at following instants. Without substantial loss of generality, we may confine ourselves to automata whose programs contain no forbidden pairs (condition of complete specification), and make

no further mention of incompletely specified automata, which admit forbidden pairs.

Thus, suppose that the control block of the automaton is set to its initial state $q(t_0)$, and symbols $x(t_0), x(t_0 + 1), x(t_0 + 2), \ldots$ are applied to its input channel. Then, in accordance with its program, the automaton generates a sequence of output signals $y(t_0), y(t_0 + 1), y(t_0 + 2), \ldots$ and the control block goes through a sequence of internal states $q(t_0 + 1)$, $q(t_0 + 2), \ldots$ This completely describes the functioning of the automaton. It is clear that the output signal generated by the automaton at some sampling time t depends not only on the input symbol received then but also on previous input symbols; the latter are recorded in the automaton by changes in its internal state. In this sense, the set of internal states of an automaton constitutes its (internal) memory. The external medium from which the automaton draws the input information is conveniently represented as a finite or infinite one-dimensional tape, divided into squares, each containing an input symbol. At the beginning of the operation, the control block is set to some initial state, while the input channel (reading head) scans the square chosen as the initial square and reads the symbol recorded there. The tape then moves from square to square in one direction (say from right to left), and so the automaton can read the input symbols recorded in the successively scanned squares of the tape. If the tape is bounded on its right, the reading head falls off the tape after a finite number of sampling times, and the automaton then stops functioning. If the tape is not bounded on its right, the process continues indefinitely. We can also assume that the automaton has another (output) tape, moving to the left in synchronism with the input tape; all squares of the output tape are empty at the beginning of the operation and the output channel (writing head) records the successive output symbols in them.

One can (and sometimes must) specify additional details in this description of the structure and components of the automaton. However, since in this book we are interested not so much in how automata are constructed as in how they function and what they can do, we do not need this specification; we shall concentrate our attention on situations directly related to the program (instruction system) of the automaton. This justifies the following definitions.

An *automaton** is a quintuple $\langle Q, X, Y, \Psi, \Phi \rangle$, where Q, X, Y are alpha-

* *Translator's note*: In the Western literature it is now customary to reserve the term "automaton" for mathematical machines with no output; when there is an output the term "machine" is employed.

bets (internal, input and output alphabets, respectively), Ψ (the next-state function) is a mapping of $Q \times X$ into Q, Φ (the output function) is a mapping of $Q \times X$ into Y. The symbols of Q are called the (internal) states of the automaton. The quadruples $\langle q, x, \Psi(q,x), \Phi(q,x)\rangle$ are called instructions of the automaton; an alternative notation for the instructions is $qx \to \Psi(q,x)\,\Phi(q,x)$.

Let q_0 be some fixed state of an automaton $\mathfrak{M} = \langle Q, X, Y, \Psi, \Phi\rangle$. Then the recurrence relations

$$\begin{aligned} q(t+1) &= \Psi[q(t),x(t)], \\ y(t) &= \Phi[q(t),x(t)], \end{aligned} \tag{1}$$

where $q(t), q(t+1) \in Q$, $x(t) \in X$, $y(t) \in Y$, with the initial condition

$$q(1) = q_0,$$

define an *operator* (which we denote by $T(\mathfrak{M}, q_0)$), which transforms every finite sequence of input symbols

$$x = x(1)x(2)x(3)\ldots x(r)$$

into a sequence, of the same length, of output symbols:

$$y = Tx = y(1)y(2)\ldots y(r).$$

The pair $\langle \mathfrak{M}, q_0\rangle$ is called an *initialized automaton*, and we shall say that the initialized automaton $\langle \mathfrak{M}, q_0\rangle$ *realizes* the operator $T(\mathfrak{M}, q_0)$, or, equivalently, that the operator $T(\mathfrak{M}, q_0)$ is the *behavior* of the initialized automaton $\langle \mathfrak{M}, q_0\rangle$. An automaton $\mathfrak{M}$ is said to realize an operator I if, for some suitably chosen initial state q_0,

$$T = T(\mathfrak{M}, q_0).$$

Any finite nonempty sequence of symbols from some alphabet A is called a *word over the alphabet* A,* and any set of words in A is a *language* over A.** Words over Q, X, Y are called *internal*, *input* and *output* words, respectively.

Analogously, the term *ω-word over the alphabet A*, *input ω-word*, *output ω-word*, *internal ω-word*, *ω-language* will be used instead of the word combinations *infinite sequence of symbols in A*, *set of infinite sequences of*

* Other, synonymous terms appear in the literature: string over A, tape over A.

** Synonymous terms are: event over A, set of tapes over A.

symbols (i.e., ω-words) *in* A, and so on. Thus, the operator $T(\mathfrak{M}, q_0)$ transforms (input) words $x = x(1)x(2)\ldots x(r)$ into (output) words $y = y(1)y(2)\ldots y(r)$; operators of this type may be called *word operators.* It is clear, however, that one can also describe the behavior of an initialized automaton in terms of an *ω-word operator*—an operator which transforms (input) ω-words into (output) ω-words. Indeed, for any $x = x(1)x(2)x(3)\ldots$ the recurrence relations (1), together with the initial condition $q(1) = q_0$, define a unique ω-word $Tx = y = y(1)y(2)y(3)\ldots$ The word behavior of an initialized automaton $\langle \mathfrak{M}, q_0 \rangle$ and its ω-word behavior are obviously closely related, and each determines the other uniquely. Consequently, in our subsequent, more detailed discussion of operators it will not be too important whether $T(\mathfrak{M}, q_0)$ stands for a word operator or an ω-word operator.

An automaton is said to be *finite* if its internal alphabet is finite.

It is obvious that a finite automaton is a constructive entity, since the finiteness of the alphabets means that one can define the mappings Ψ and Φ by means of finite tables, that is to say, one can list all the instructions. It is also clear that the operators induced thereby are effective, in the sense that the recurrence relations (1) may be used to compute successively $q(1)q(2)\ldots$ and $y(1)y(2)\ldots$, provided that the initial states and $x(1)x(2)\ldots$ are known.

If Q is infinite our definition imposes no formal restrictions on the mappings Ψ and Φ, which may therefore turn out to be noneffective. As a matter of fact, in all interesting problems only effective next-state and output functions are considered, and their effectiveness is moreover of a rather special type. These questions are clarified and formulated in the structural theory of automata (see, e.g., [6, 7]), and we shall mostly ignore them.

Every finite automaton $\langle Q, X, Y, \Psi, \Phi \rangle = \mathfrak{M}$ may be defined by two finite tables with binary input, corresponding to the functions Ψ and Φ. In these tables, known as the *transition matrix* and the *output matrix*, re-

TABLE 1a

$\Psi(q, x)$

x \ q	1	2	3
a	3	3	1
b	2	3	3

TABLE 1b

$\Phi(q, x)$

x \ q	1	2	3
a	b	a	b
b	c	c	c

spectively, the rows are labeled by the input letters and the columns by the states (see, for example, Tables 1a and 1b, where $Q = \{1, 2, 3\}$; $X = \{a, b\}$; $Y = \{a, b, c\}$).

Given fixed alphabets $Q = \{q_1, q_2, \ldots, q_k\}$, $X = \{x_1, \ldots, x_m\}$, $Y = \{y_1, \ldots, y_n\}$, the transition matrix may be filled out in k^{mk} ways, the output matrix in n^{mk} ways. Thus the total number of automata with fixed alphabets consisting of k, m, n symbols, respectively, is exactly $(kn)^{mk}$.

REMARK. The two tables corresponding to the functions Ψ and Φ may be replaced by one, whose rows are labeled by the input letters and columns by the states, while the entries specify the values of the functions Ψ and Φ in pairs.

EXAMPLE. Consider the finite automaton $\mathfrak{M} = \langle Q, X, Y, \Psi, \Phi \rangle$, where $Q = \{1, 2, 3\}$, $X = \{a, b\}$, $Y = \{a, b, c\}$, and Ψ and Φ are given by Tables 1a and 1b. The instructions of this automaton are:

1) $1a \to 3b$, 3) $2a \to 3a$, 5) $3a \to 1b$,

2) $1b \to 2c$, 4) $2b \to 3c$, 6) $3b \to 3c$.

Suppose that at some time t the automaton is in state 1, and the sequence abb is applied at its input at times t, $t + 1$, $t + 2$; the automaton then generates the output sequence bcc and will be in state 3 at time $t + 3$.

0.2. Types of automata

The above concept of an automaton is fairly general. In the literature one can find various particular cases (often differently named), obtained by imposing various restrictions on the components Q, X, Y, Ψ, Φ of the automaton. Most important are restrictions on the cardinalities of the alphabets. In particular, the requirement that the internal alphabet be finite defines the class of finite automata; this is the mathematical explicatum of the condition that the internal memory of the automaton be finite.

Special mention should be made of "degenerate" cases, in which one of the alphabets Q, X, Y is a singleton (= one-element set). In such cases it is convenient to modify the definition of the automaton by dropping the degenerate component(s) from the quintuple $\langle Q, X, Y, \Psi, \Phi \rangle$ and modifying the other components accordingly. We proceed to these definitions.

a) A *memoryless automaton** is a triple $\langle X, Y, \Phi \rangle$, where Φ is a mapping

* *Translator's note*: This resembles the "feedback-free" automaton of Hartmanis and Stearns.

of the input alphabet X into the output alphabet Y. In other words, the instructions of a memoryless automaton have the form

$$x_r \to y_s,$$

where $x_r \in X$, $y_s \in Y$. The recurrence relations (1) become

$$y(t) = \Phi[x(t)], \tag{1a}$$

that is to say, the output symbol at a given time depends only on the input symbol at that time and is absolutely independent of previously received symbols. Thus every memoryless automaton realizes a unique operator which performs "literal translation" of the input symbols into the output symbols; operators of this type are known as *truth-table operators* or *memoryless operators.*

The number of all memoryless automata with given alphabets $X = \{x_1, x_2, \ldots, x_m\}$ and $Y = \{y_1, y_2, \ldots, y_n\}$ is n^m.

b) An *autonomous automaton** is a quadruple $\langle Q, Y, \Psi, \Phi \rangle$, where Ψ and Φ map Q into Q and Y, respectively. The instructions are of the form $q_i \to q_j y_s$, and the recurrence relations (1) are

$$\begin{aligned} q(t+1) &= \Psi[q(t)], \\ y(t) &= \Phi[q(t)]. \end{aligned} \tag{1b}$$

Given an initial state $q(1) = q_0$ of an autonomous automaton, these relations (1b) uniquely define an output ω-word $y(1)y(2)\ldots$ and an internal ω-word $q(1)q(2)\ldots$. Obviously, if an autonomous automaton is finite, with k states, the sequence $q(1)q(2)\ldots q(k+1)$ must contain repeated elements; thus, the ω-words $y(1)y(2)y(3)\ldots$ and $q(1)q(2)q(3)\ldots$ are both periodic (not necessarily purely periodic, i.e., there may be some phase), with period no greater than the number of states of the automaton. The number of all autonomous automata with given alphabets $Q = \{q_1, q_2, \ldots, q_k\}$ and $Y = \{y_1, \ldots, y_n\}$ is obviously $(nk)^k$.

c) An *outputless automaton*** is a triple $\langle Q, X, \Psi \rangle$, where Ψ maps $Q \times X$ into Q. The instructions of an outputless automaton have the form $q_i x_r \to q_j$, and only the first recurrence relation of (1) is retained:

$$q(t+1) = \Psi[q(t), x(t)]. \tag{1c}$$

The total number of outputless automata with alphabets $Q = \{q_1, \ldots, q_k\}$

* *Translator's note:* Called a "clock" by some Western authors.

** *Translator's note*: Hartmanis and Stearns call this a *state machine.*

and $X = \{x_1, \ldots, x_m\}$ is k^{mk}. The behavior of an outputless automaton cannot be described in terms of word (or ω-word) operators mapping input words (ω-words) into output words (ω-words). Of course, one could consider operators mapping sequences of input symbols into sequences of internal states, as defined by the recurrence relation (1c) for some fixed initial state $q(1)$. However, it is more convenient to define the behavior of outputless automata in terms of the languages (ω-languages) that they represent. The necessary definitions and notation follow.

Given an automaton $\mathfrak{M} = \langle Q, X, \Psi \rangle$, fix a state q_0 and an input word $x = x(1)x(2)\ldots x(r)$. Then the recurrence relations

$$q(1) = q_0, \quad q(t+1) = \Psi[q(t), x(t)]$$

uniquely define an internal word $q(1)\ldots q(r)q(r+1)$. If $q(r+1) = q' \in Q$, we shall say that the word x takes state q_0 into state q'.

An *anchored* (outputless) *automaton** is a triple $\langle \mathfrak{M}, q_0, Q' \rangle$, where $\mathfrak{M}$ is an outputless automaton, q_0 some distinguished state (the *initial state*) and Q' a subset of the state set Q (Q' is the set of *final states*). The (possibly empty) set of input words which take q_0 into some final state $q \in Q'$ is called the *language represented*** *by the anchored automaton* $\langle \mathfrak{M}, q_0, Q' \rangle$, or its *behavior*, and is denoted by $\omega(\mathfrak{M}, q_0, Q')$.

In this treatment, an outputless automaton is regarded as a device which receives questions (after suitable "anchoring," i.e., selection of its initial and final states) and answers "yes" or "no." The application of an input word $x = x(1)\ldots x(r)$ is interpreted as the question: does this word belong to our language? If the automaton ends up in a final state, the answer is affirmative; otherwise it is negative.†

EXAMPLE. Consider the automaton $\mathfrak{M}$ defined by Table 2, anchored in the following way: the initial state is 1, which is also the only final state. This automaton represents the language consisting of all words with an even number of ones.

* *Translator's note:* This is the usual definition of "automaton" (or Rabin-Scott automaton) in the West. The term "anchored" is apparently due to Rabin (private communication from E. Shamir).

** *Translator's note*: The more common term in the Western literature is *accepted.*

† In the linguistic interpretation, each symbol is interpreted as a word-token (not a letter!) and what we have called a word is a sequence of word-tokens—a sentence. Thus mathematical linguistics employs the terms "symbol," "vocabulary," "string" rather than "letter," "alphabet," "word." The questions that the automaton is asked have the following meaning: is this string of symbols a grammatically regular sentence in the language under consideration?

TABLE 2

x \ q	1	2
0	1	2
1	2	1

Another definition of behavior for an outputless automaton is based on consideration of the input ω-words that the automaton receives.

Consider an ω-word $a = a(1)a(2)\ldots$ over the alphabet A. It is natural to call a symbol a *limit symbol* of a if it appears in a infinitely many times. Denote the set of all limit symbols of the ω-word x by $\lim x$. Fix a state q_0 and an input ω-word $x = x(1)x(2)\ldots$ in the automaton $\mathfrak{M} = \langle Q, X, \Psi\rangle$. Then, as mentioned above, the recurrence relations (1c) define an internal ω-word $q = q(1)q(2)\ldots$ Let $Q' = \lim q$. Then we shall say that the input ω-word x takes the state q_0 of the automaton $\mathfrak{M}$ into the limit set of states $Q' \subseteq Q$. Now, by analogy with "anchoring" of $\mathfrak{M}$, we consider "macro-anchoring" of the automaton, i.e., selection of an initial state and a system of limit sets of states. This motivates the following definitions.

A *macrostate* of an automaton is any subset of its state set.

A *macroanchored automaton* is a triple $\langle \mathfrak{M}, q_0, \mathfrak{C}\rangle$, where $\mathfrak{M}$ is an outputless automaton, q_0 a distinguished (initial) state, $\mathfrak{C}$ a set of macrostates (the *limit macrostates*). The set of all ω-words which take q_0 into some macrostate $Q' \in \mathfrak{C}$ is called the *ω-language represented* by the macroanchored automaton $\langle \mathfrak{M}, q_0, \mathfrak{C}\rangle$ and denoted by $\Omega(\mathfrak{M}, q_0, \mathfrak{C})$.

EXAMPLE. Consider the automaton $\mathfrak{M}$ defined by Table 2. Macroanchor $\mathfrak{M}$ as follows: the initial state is 1, and $\mathfrak{C}$ contains the macrostates $\{1, 2\}$ and $\{2\}$. This macroanchored automaton represents the ω-language consisting of all ω-words containing infinitely many ones and all ω-words containing a finite, but odd number of ones.

We have thus considered the behavior of three special types of "degenerate" automata, in which one of the alphabets Q, X, Y is a singleton. The behavior of a memoryless automaton is an extremely simple and clear-cut object—a truth-table operator. The behavior of an autonomous automaton is characterized by one output ω-word and one internal ω-word; if the autonomous automaton is finite (and this is the case of interest), each of these ω-words is periodic and we are again dealing with very simple objects.

This is no longer true for outputless automata, even finite ones. The classes of languages and ω-languages representable by finite outputless automata are quite extensive. As we shall see later, each of these classes is, in a certain sense, as rich as the class of all operators definable by finite automata. In other words, the restriction that led us to the concept of outputless automata is in fact inessential, in contrast to the restrictions leading to memoryless and autonomous automata. To some extent, this might have been expected; of the two relations (1), only the first is a real recurrence relation and thus it is the more "informative" of the two. For this reason, we shall not devote special attention to memoryless or autonomous automata, whereas outputless automata will be studied in considerable detail.

To conclude this section, we shall briefly dwell on two types of automaton which arise when restrictions are imposed on the output function.

d) *Automaton with delay.* The output function Φ is a function $\lambda(q)$ independent of x. The output symbol of an automaton with delay at a sampling time t is independent of the current input symbol; it depends only on previously received symbols.

e) In a *Moore automaton* the functions Φ and Ψ must satisfy the condition $\Phi[q,x] = \lambda[\Psi(q,x)]$, where λ, the so-called *shifted output function*, is a mapping of Q into Y. Thus, the only difference between automata with delay and Moore automata lies in the form of the second recurrence relation of (1), the relation $q(t+1) = \Psi[q(t),x(t)]$ being common to both types. For the former,

$$y(t) = \lambda[q(t)],$$

while for the latter

$$y(t) = \lambda[q(t+1)].$$

Automata with delay are especially useful in the structural theory of automata; in this book there is no justification for a special discussion. Moore automata, on the other hand, are often convenient in the behavioral theory (see Sections II.3 and II.5).

Given the alphabets $Q = \{q_1, \ldots, q_k\}$, $X = \{x_1, \ldots, x_m\}$ and $Y = \{y_1, \ldots, y_n\}$, the number of automata of either of the above types is $k^{mk} \cdot n^k$.

0.3. Automata and graphs

In automata theory it is convenient to employ the language of graph theory, whose visual clarity makes it easy to apply many of its concepts and methods

to automata. By a *graph* we mean a directed multigraph—a graph in which vertices α and β may be connected by more than one directed edge. An automaton $\mathfrak{M} = \langle Q, X, Y, \Psi, \Phi \rangle$ may be represented by a graph whose vertices are labeled by the symbols of Q (the states of the automaton); each instruction $q_i x_r \to q_j x_s$ corresponds to an edge going from the vertex q_i to the vertex q_j, labeled by the pair $x_r y_s$ (x_r is the input label, y_s the output label of the edge). Figure 1 illustrates the graph of the automaton defined above by Table 1. Here the input and output alphabets have common letters; the output labels are distinguished from the input labels by enclosing the former in parentheses.

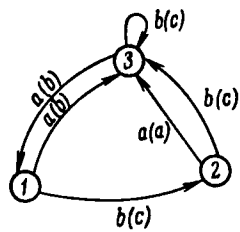


Figure 1

Let us use the term *diagram* over the alphabets Q, A for any graph whose vertices are labeled in one-to-one fashion by the letters of Q (i.e., every symbol in Q is used exactly once as the label of a vertex), while the edges are labeled arbitrarily by symbols of A (here not every symbol need be used as a label, and different edges may be identically labeled). We shall apply standard graph-theoretic terminology to diagrams, speaking of connected diagrams, subdiagrams, etc.

Thus, any automaton $\langle Q, X, Y, \Psi, \Phi \rangle$ is defined by a diagram over the alphabets $Q, X \times Y$, but not every diagram over $Q, X \times Y$ defines an automaton. The conditions for a diagram to represent an automaton, which we call the *automaton conditions*, are obvious:

1. No two edges with identical input labels issue from the same vertex (uniqueness condition).

2. For any vertex q and any input symbol x, there is an edge labeled x issuing from q (complete-specification condition).

It should already be clear how one can define the special types of automata considered in Section 0.2 via diagrams, and what properties their diagrams possess. Thus, for example, an outputless automaton is defined by a diagram whose edges are labeled by letters of the input alphabet, provided the above conditions are satisfied. Such a diagram will be called an *automaton graph.*

The diagram of Figure 2 does not define an automaton, since the automaton conditions are not satisfied. But if we add an edge going from the vertex 1 to some vertex, labeling it b, and remove one of the edges issuing from 2 and labeled b, the result is an automaton graph. The diagram of an autonomous automaton contains exactly one edge issuing from each vertex, and this edge is labeled by an output letter. Figure 3 is a diagram of the automaton defined by Table 3; it consists of two connected components, one a simple loop, the other a cyclic tree.

The diagram of an automaton with delay has the characteristic property that all edges issuing from a vertex have the same output label.

The characteristic property of the diagram of a Moore automaton is that all edges converging on a vertex have the same output label.

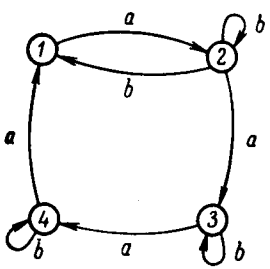


Figure 2

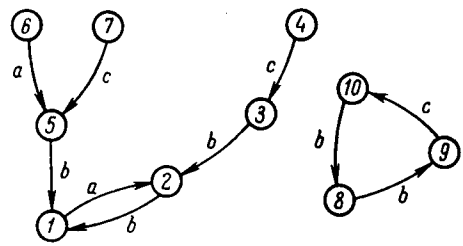


Figure 3

TABLE 3a

q	1	2	3	4	5	6	7	8	9	10
$\Psi(q)$	2	1	2	3	1	5	5	9	10	8

TABLE 3b

q	1	2	3	4	5	6	7	8	9	10
$\Phi(q)$	a	b	b	c	b	a	c	b	c	b

We shall sometimes use the correspondence between automata and diagrams rather freely, referring to the vertices of a diagram as its states and to sets of vertices as macrostates. Finite automata of course define finite graphs (diagrams), infinite automata—infinite graphs. An example

of an infinite diagram is a tree with m edges (branches) issuing from each vertex, the edges labeled in turn by the letters of the input alphabet $X = \{x_1, x_2, \ldots, x_m\}$ (see Figure 4 for the two-letter alphabet $X = \{a, b\}$). This tree defines an infinite outputless automaton, whose states may be denoted $1, 2, 3, \ldots$. If the edges are also labeled with letters from an output alphabet Y, we get the tree-diagram of an automaton with output. Diagrams of this type will frequently be considered in the sequel, and, wherever the meaning is clear from the context, we shall use the unmodified terms *infinite tree* or *tree*.

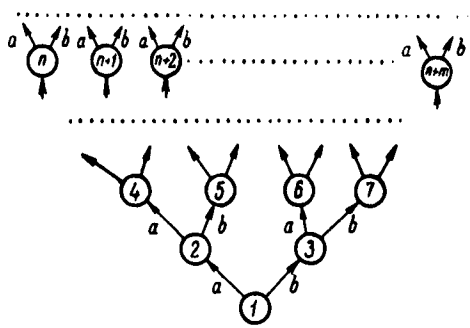


Figure 4

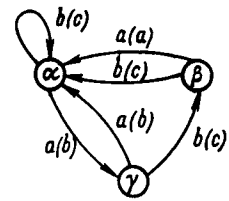


Figure 4′

We now recall some additional graph-theoretic terminology and notation which we shall often use. As usual, a path in a graph G (in particular, in a diagram) from a vertex α to a vertex β is a finite sequence $\alpha = \alpha_1, A_1, \alpha_2, A_2, \alpha_3, A_3, \ldots, A_n, \alpha_{n+1} = \beta$, where $A_i (i \leqq n)$ is an edge going from α_i to α_{i+1}. If $\alpha = \beta$ the path is a loop; if the vertices $\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_n$ (but not necessarily α_{n+1}) are pairwise distinct, the path (or loop) is said to be simple. Similarly, an ω-path in a graph G, beginning at a vertex α, is an infinite sequence $\alpha = \alpha_1, A_1, \alpha_2, A_2, \alpha_3, \ldots, A_j, \alpha_{j+1}, \ldots$, where, as before, A_i is an edge from α_i to α_{i+1} $(i = 1, 2, \ldots)$. A *flow* (*ω-flow*) in a graph is any set of paths (ω-paths). Every path (ω-path) ξ in a diagram uniquely defines a word (ω-word) $\omega(\xi)$ over the alphabet X, obtained by writing the labels of the edges in ξ in order. We shall say that ξ *carries* the word (ω-word) $\omega(\xi)$. Accordingly, every flow (ω-flow) Π in a diagram may be associated with a language $\omega(\Pi)$ (ω-language $\Omega(\Pi)$), consisting of all words (ω-words) carried by paths (ω-paths) in Π. We shall say that Π *carries* the language $\omega(\Pi)$ (ω-language $\Omega(\Pi)$).

In these terms, the language $\omega(\mathfrak{M}, q_0, Q')$ is precisely the language carried

in the corresponding diagram by the flow consisting of all paths from q_0 to Q'. Similarly, the ω-language $\Omega(\mathfrak{M}, q_0, \mathfrak{C})$ is the ω-language carried by the ω-flow consisting of all ω-paths beginning at q_0 with limit sets in $\mathfrak{C}$. The operator T realized by an initialized automaton $\langle \mathfrak{M}, q_0 \rangle$ with output may be characterized as follows: for any input word $x(1) \dots x(r)$, consider the (unique) path beginning at q_0 and carrying input labels $x(1), \dots, x(r)$, respectively; the corresponding output word then consists of the output labels read along this path.

Throughout this book we shall make constant use of all graph-theoretic tools induced in a natural manner by the above automaton–diagram correspondence. For example, an automaton $\mathfrak{M}$ will be called a *subautomaton* of $\mathfrak{M}'$ if its diagram is a subdiagram of $\mathfrak{M}'$. Two diagrams are said to be *isomorphic* if each can be converted into the other by a suitable relabeling (renumbering) of its vertices; two automata are said to be isomorphic if their diagrams are isomorphic. This definition is applicable not only to the general automaton $\langle Q, X, Y, \Psi, \Phi \rangle$ but also to the special types in Section 0.2. A detailed definition runs as follows. Two automata $\mathfrak{M}_1 = \langle Q_1, X, Y, \Psi_1, \Phi_1 \rangle$ and $\mathfrak{M}_2 = \langle Q_2, X, Y, \Psi_2, \Phi_2 \rangle$ are isomorphic if there exists a one-to-one mapping λ of Q_1 onto Q_2 such that $\Psi_1(q, x) = \lambda^{-1}[\Psi_2[\lambda(q), x]]$ and $\Phi_1 = \Phi_2(\lambda(q), x)$. Henceforth we shall not formulate the definitions in detail, but simply refer to the corresponding graph-theoretic concepts.

EXAMPLE. The diagrams of Figures 1 and 4′ are isomorphic; therefore, the same is true of the automata that they define. The isomorphism is obtained by the following relabeling of the states: $1 \to \gamma, 2 \to \beta, 3 \to \alpha$.

It is a trivial task to compute the number of all automata with given alphabets

$$Q = \{q_1, \dots, q_k\}, \qquad X = \{x_1, \dots, x_m\}, \qquad Y = \{y_1, \dots, y_n\}$$

which, as we have seen, is $(nk)^{mk}$. Now it should be clear that in most automata problems (especially in the behavioral theory) there is no need to differentiate between isomorphic automata, and this raises the question as to the number $A(m, n, k)$ of all pairwise nonisomorphic automata with fixed alphabets

$$Q = \{q_1, \dots, q_k\}, \qquad X = \{x_1, \dots, x_m\}, \qquad Y = \{y_1, \dots, y_n\}.$$

There are $k!$ possible permutations of the set Q, and all automata (with the above alphabets) which are isomorphic to an automaton $\mathfrak{M}$ may be ob-

tained from $\mathfrak{M}$ by suitable permutations of Q. Thus,

$$A(m, n, k) \cdot k! \geqq (nk)^{mk}.$$

This is a strict inequality since in some cases different permutations of Q transform $\mathfrak{M}$ into the same automaton. However, $A(m, n, k)$ is much smaller than $(nk)^{mk}$ (the number of all automata with these alphabets), since there certainly exist different automata which are isomorphic. Thus,

$$\frac{(nk)^{mk}}{k!} \leqq A(m, n, k) \leqq (nk)^{mk}.$$

There is an exact formula for $A(m, n, k)$ but it is very unwieldy [95]. It is therefore most interesting to try to establish fairly simple asymptotic formulas [35, 36].

0.4. Terminological clarifications

This section contains a few remarks and clarifications about our use of the concepts *symbol*, *alphabet*, *word*, *ω-word*, *language*, *ω-language*. We first note that some or all of the alphabets of an automaton may be (cartesian) products of other alphabets. For example, let $X = X_1 \times X_2 \times \ldots \times X_m$. This means that the symbols of X are in effect all m-tuples $\langle x_1 x_2 \ldots x_m \rangle$ with $x_i \in X_i (i \leqq m)$. If x is a letter from the alphabet X, its *projection* on the alphabet $X_{s_1} \times X_{s_2} \times \ldots \times X_{s_\mu} (s_1 < s_2 < \ldots < s_\mu \leqq m)$ is defined as the symbol $\langle x_{s_1} x_{s_2} \ldots x_{s_\mu} \rangle$; the *projection of a word* (*ω-word*) is defined as the word (ω-word) formed by the corresponding projections of its letters. Products of alphabets arise frequently in the structural theory of automata. For example, an input alphabet $X = X_1 \times \ldots \times X_m$ may be interpreted as m input channels, each of which receives the corresponding projection of the input information. As an illustration, we confine ourselves to two simple operations on automata, which we shall use later. Given two automata

$$\mathfrak{M}' = \langle Q', X', Y', \Psi', \Phi' \rangle \quad \text{and} \quad \mathfrak{M}'' = \langle Q'', X'', Y'', \Psi'', \Phi'' \rangle,$$

we define their (direct) *product*

$$\mathfrak{M} = \langle Q, X, Y, \Psi, \Phi \rangle$$

by $Q = Q' \times Q''$, $X = X' \times X''$, $Y = Y' \times Y''$ (Figure 5*a*); for every pair of instructions $q'_i x'_r \to q'_v y'_s$ and $q''_j x''_\rho \to q''_\mu y''_\sigma$ of $\mathfrak{M}'$ and $\mathfrak{M}''$, respectively,

$\mathfrak{M}$ contains an instruction $\langle q'_i q''_j \rangle \langle x'_r x''_\rho \rangle \to \langle q'_\nu q''_\mu \rangle \langle y'_s y''_\sigma \rangle$. If $\mathfrak{M}'$ and $\mathfrak{M}''$ have a common input alphabet ($X' = X''$), one can define a related operation—*product with identified inputs.* To simplify matters, we consider the case in which $\mathfrak{M}'$ and $\mathfrak{M}''$ (and therefore also the resultant automaton $\mathfrak{M}$) are outputless (Figure 5*b*): $\mathfrak{M} = \langle Q, X, \Psi \rangle$, where $X = X' = X''$, $Q = Q' \times Q''$ and for every pair of instructions $q'_i x_\lambda \to q'_\nu$, $q''_j x_\lambda \to q''_\mu$ of $\mathfrak{M}'$ and $\mathfrak{M}''$ respectively, $\mathfrak{M}$ has an instruction $\langle q'_i q''_j \rangle x_\lambda \to \langle q'_\nu q''_\mu \rangle$.

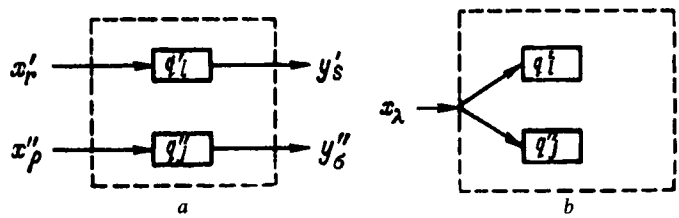


Figure 5

Using products of alphabets, we can regard the operator defined by an automaton as having several arguments, i.e., it transforms an ordered sequence of input words (ω-words) of the same length into an ordered sequence of words (ω-words), again of the same length. In exactly the same way, the language (ω-language) represented by an automaton can be regarded as a set of ordered sequences of words (ω-words).

TABLE 4

x_1x_2 \ q	q_0	q_1
00	$0q_0$	$1q_0$
01	$1q_0$	$0q_1$
10	$1q_0$	$0q_1$
11	$0q_1$	$1q_1$

For example, consider the automaton defined by Table 4, which is known as a serial binary adder. Its input symbols are the pairs 00, 01, 10, 11. If the initial state is q_0, the operator of the automaton transforms any pair of words $x_1(1) \ldots x_1(r)$, $x_2(1) \ldots x_2(r)$ into a word $y(1) \ldots y(r)$, where $x_1(i)$, $x_2(i)$, $y(i)$ ($i \leqq r$) are the i-th digits from the right in the binary expansions of the first term, second term and sum, respectively.

Very commonly used alphabets in the theory of automata are cartesian powers of the binary alphabet $\{0, 1\}$. Words (ω-words) in the alphabet $\{0, 1\}$ may then be regarded as predicates defined over initial sections of the natural number sequence (over the entire natural number sequence), by identifying the symbols 1 and 0 with the truth values "true" and "false." With this in mind, one can use the operations of logic to set up formulas describing the next-state function Ψ and the output function Φ. For a serial binary adder,

$$\Psi(q,x_1,x_2) = x_1 \cdot x_2 \vee x_1 \cdot q \vee x_2 \cdot q,$$

$$\Phi(q, x_1, x_2) = x_1 \oplus x_2 \oplus x_3,$$

where $\cdot$, $\vee$, $\oplus$ denote conjunction, disjunction and addition mod 2, respectively, and q assumes the values 0 and 1.

The set of all words (the *universal language*) over a given alphabet A is in effect the free semigroup with respect to *concatenation*, the binary operation mapping an ordered pair of words a,b onto the word ab obtained by juxtaposition of b to the right of a. It is clear that this operation is associative: $(ab)c = a(bc)$, and so one can omit the parentheses in concatenations of more than two words.

Analogous to the concatenation of two words is the concatenation of a word a and an ω-word b, defined as the ω-word ab obtained by juxtaposing the ω-word b to the right of a. One can also define the concatenation of an infinite sequence of words. For example, given the words p, r, $ppp\ldots$ is a periodic ω-word, $rppp\ldots$ an ultimately periodic ω-word with phase r and period p. We shall be rather liberal with our terminology and notation, not distinguishing between a word containing a single letter and the letter itself, or between a language containing a single word and the word itself (wherever the meaning is self-evident).

Formal considerations sometimes make it convenient to introduce the *empty word* (to be denoted by $\wedge$), stipulating that $\wedge p = p \wedge = p$ for any word p and $\wedge p = p$ for any ω-word p. The universal language then becomes a semigroup with identity [also called a *monoid* in Western literature] (the role of the identity being played by the empty word); this is sometimes convenient in formulating algebraic characterizations of languages, especially finite-state languages.* For this reason, the term "language" (in par-

* We use the term *finite-state* instead of the word combinations "representable in a finite automaton," "realizable by a finite automaton."

ticular, language representable by an automaton) is often used in the literature for a set of words including (possibly) the empty word. To be precise, the empty word is said to belong to the language $\omega(\mathfrak{M}, q_0, Q')$ if one of the final states coincides with the initial state q_0. In terms of diagrams, the ordinary paths from q_0 to Q' are supplemented by an "empty path" carrying the empty word. The reader should take care to distinguish sharply between the empty language $\varnothing$ and the language $\{ \wedge \}$ which contains the empty word alone. It should be clear from the definitions of the preceding sections that we are considering only nonempty words, and so our languages (such as languages representable by automata) do not contain the empty word.*

Since our approach is by no means generally accepted, we shall make two almost obvious observations which should convince the reader, as he reads further into the book, that all our theorems and proofs carry over trivially to systems admitting the empty word.

Let $\langle \mathfrak{M}, q_0, Q' \rangle$ be a finite anchored automaton. Then:

a) There is an effective procedure for deciding whether the empty word belongs to the language** that it represents (i.e., $q_0 \in Q'$).

b) There is an effective construction of a finite anchored automaton $\langle \mathfrak{M}', \pi_0, \Pi' \rangle$ such that the only difference between $\omega(\mathfrak{M}', \pi_0, \Pi')$ and $\omega(\mathfrak{M}, q_0, Q')$ is that one of these languages contains the empty word while the other does not (they contain exactly the same nonempty words). This construction, which we call *adjunction of an initial state*, proceeds as follows. Enlarge the state set of $\mathfrak{M}$ by one additional state π_0; for every instruction of $\mathfrak{M}$ of the form $q_0 x \to q'$, add an instruction $\pi_0 x \to q'$. (In other words, add a new vertex to the diagram of $\mathfrak{M}$ and draw edges from this vertex to all vertices which are ends of edges from q_0, retaining the same labels) The state π_0 replaces q_0 as the initial state of the automaton. The new final states are all states of Q', together with π_0 if and only if $q_0 \notin Q'$. Thus, the empty word belongs to $\omega(\mathfrak{M}, \pi_0, \Pi')$ if and only if it does not belong to $\omega(\mathfrak{M}, q_0, Q')$. It is easy to see that $\omega(\mathfrak{M}, q_0, Q')$ and $\omega(\mathfrak{M}', \pi_0, \Pi')$ contain exactly the same nonempty words.

* Had we utilized the empty word in describing the behavior of automata, we should also have had to define the empty ω-word, which (apart from a certain dissonance in the term itself) in no way simplifies the exposition. Moreover, recourse to "empty words" requires great care; an instructive example is described briefly in Problem XV of Chapter III.

** From here to the end of this section we shall use representations admitting the empty word.

0.5. Survey of the contents of Chapters I to V

Chapter I is devoted to the behavior of finite outputless automata, or, in the terminology of Section 0.2, finite-state languages and ω-languages. The main problem here is to delineate the class of finite-state languages (ω-languages). This is done in two ways. First, we introduce several concepts which are useful in formulating criteria for a language to be finite-state (Sections I.2, I.3, I.11, I.12). We then describe various operations under which the class of finite-state languages is closed (Sections I.1, I.5–I.10), and study each in some detail. The concepts of distinguishability (Section I.3) and interchangeability (Section I.2) enable us to characterize the memory capacity required to represent a given language (ω-language). The concepts of probabilistic automaton (Section I.11) and grammar (Section I.12) are discussed with an eye to the concepts of language representation that they define. It is instructive to compare each of these concepts with the usual finite-automaton concept. One result is that grammars with finitely many states give nothing new in comparison with finite automata, but probabilistic automata lead to a considerable extension of the class of representable languages (ω-languages).

The main content of this chapter involves operations under which the class of finite-state languages is closed. This class is closed under many operations (though one can easily cite simple examples of languages or ω-languages which are not representable in finite automata). Roughly speaking, the operations we shall study fall into two groups: a) set-theoretic and logical; b) operations defined in terms of word concatenation (see Section I.4). In all cases our proofs of closure properties yield effective closure, i.e., they prove the existence of an algorithm which, given anchored (macro-anchored) finite automata, construct the resultant anchored (macro-anchored) automaton.

Languages and ω-languages will be discussed in parallel. The principal difficulty relates to the operation of projection, whose logical counterpart is the existential quantifier. To overcome this difficulty we introduce a more general concept than the anchored (macroanchored) automaton—the source (macrosource) or nondeterministic automaton [which is the accepted term in the Western literature] (dropping the uniqueness condition; see Section 0.1). Of basic importance here are the determinization theorems, which show that this generalization of automata does not extend the class of representable languages (ω-languages). The determinization theorem for sources has been known for some time and is quite simple.

McNaughton's determinization theorem for macrosources, which gives similar information about ω-languages, is based on more subtle constructions; with its aid the analogy between representable languages and ω-languages may be carried to its logical completion.

Much of the material in Section I.4, though of independent interest, is used in the proofs of subsequent theorems. We refer to existence theorems for algorithms which can be used to check whether a language (ω-language) representable by a given finite automaton has various properties. While fairly easy to prove, these theorems are of essential importance. Later (in Chapter II) we shall cite examples of properties for which no algorithms exist.

Chapter II is devoted to the behavior of automata with output, i.e., finite-state operators. We begin with a classification of word (ω-word) operators according to the criteria of anticipation (Section II.1) and memory (Section II.2); finite-state operators turn out to be nonanticipatory [or causal] operators with finite memory. We then distinguish clearly between the properties of these operators due to their lack of anticipation and the properties due to the finiteness of their memory (weight). This enables us to phrase the arguments in such a way as to reveal, in passing, certain laws applying to operators definable by infinite automata and sometimes even to operators not definable by any (even infinite) automaton. In a certain sense, the material of Sections II.3 and II.4 is analogous to that of Sections I.2 and I.3; here we determine the memory capacity needed for the definition of any given finite-state operator. Though an outputless automaton is only a particular case of a general automaton (with output), many essential propositions concerning the behavior of automata with output are actually established in Chapter I. Section II.5 indicates the relation between the two concepts of automaton behavior, so that we are able to single out those facts specific to automata with output whose verification requires a special study. These facts relate to the concept of uniformization of a finite-state ω-language $\mathfrak{A}$ over the product of alphabets X, Y. i.e., the existence of a finite-state operator $y = Tx$ whose graph lies in $\mathfrak{A}$. The problems considered here resemble those usually arising when one is interested in explicitly defining a function given by an implicit definition. Using McNaughton's suggested game-theoretic interpretation, we shall present Büchi and Landweber's solution of the uniformization problem in Sections II.9 and II.10. It is of paramount importance for solution of the synthesis problem.

Sections II.11 through II.13 constitute a digression; they introduce

various parameters (degree of distinguishability, accessibility, reproducibility) and spectra (= sequences of numerical parameters) which serve for a finer classification of operators and automata. Various types of lower and upper bounds for spectra and parameters are established. Though of independent value, these facts are here mainly in the nature of preparatory material for Chapters IV and V. (Chapter V deals with statistical bounds for these parameters and spectra.) Absolute and statistical bounds for behavioral parameters are particularly important in the so-called theory of experiments on automata, a basic component of which is the theory of synthesis of automata in identification problems (this theory is set forth in Chapter IV).

Chapters III and IV deal directly with the synthesis problem, though the problem is formulated differently in each chapter.

In the most general terms, the synthesis problem is to construct an automaton whose behavior satisfies certain explicitly specified or implicit properties. The problem is central in both behavioral and structural automata theory. Of course, here we must restrict ourselves to those aspects of the problem which concern the behavioral theory, and in this sense the term "functional synthesis" is used in contradistinction to structural synthesis. Moreover, we restrict ourselves to the functional synthesis of finite automata. In this formulation, an automaton is deemed constructed once we have finite tables defining its next-state and output functions, or once we have drawn its (finite) diagram. Within the bounds of the general synthesis problem, functional synthesis constitutes only the first step of the construction of an automaton, serving as raw material for the next step, in which the actual structure (circuit) of the automaton is designed.

Our formulation and investigation of the synthesis problem distinguish between the following two situations which reflect the dialog between the prospective user of the automaton and the designer.

1. *Metalanguage.* The user presents the designer with the conditions to be satisfied by the behavior of the projected automaton (i.e., language or ω-language, operator). These conditions are assumed to be stated in a sufficiently unambiguous and formalized language (the so-called metalanguage), understood by the designer. This situation is examined in detail in Chapter III, with extensive use of the basic theorems on the behavior of automata (Chapters I and II), as well as predicate logic.

2. *Identification.* The user has fully planned the required behavior (language, ω-language, operator), but is not in a position (or is unwilling) to formulate the conditions in a language accessible to the designer. It is also

assumed that the user can (and must) provide the designer with answers to questions of the type "What does the operator do with the word $x(1)\dots x(t)$?" or "Does the word $x(1)\dots x(t)$ belong to the language?" Thus, by dint of suitable inquiries, the designer tries to "guess" the operator (language, ω-language) and (if possible) to construct a suitable automaton. It is known that this identification procedure is not always feasible, i.e., there is no algorithm for "guessing" all projected operators. This being so, can one guess *some* of these operators (more precisely, a sufficiently large proportion of them), and what algorithms can be used to that end? These questions are studied in Chapter IV.

We now examine the detailed contents of each of the last-mentioned chapters.

Chapter III discusses various particular problems relating to the general synthesis problem in the metalanguage situation. These include the existence, uniqueness and construction of an automaton conforming to conditions expressed in the metalanguage. The situation is similar to that obtaining, for example, in the theory of functional equations, when analogous questions arise for formulas of a certain metalanguage (such as differential equations). The investigation proceeds in regard to several metalanguages (sources, finite trees, regular formulas, ω-regular formulas) and, finally, a special metalanguage (the language **I**) based on monadic second-order predicate logic. The aim of our argument is to show that this metalanguage is considerably more expressive than those listed above, and we prove that there exist algorithms solving the general synthesis problem for **I**. These algorithms are far more complicated than the synthesis algorithms for the other languages—understandably so, since they solve the synthesis problem for them too. The comparatively brief exposition of the synthesis problem in this chapter is possible thanks to the extensive preparation of all auxiliary material in the preceding chapters. In effect, the algorithms described in the proofs that the class of finite-state languages (ω-languages, operators) is closed under various types of operations are constituent parts of the synthesis procedure. In Chapter III these algorithms are merely assembled and combined into a single synthesis algorithm.

The metalanguage **I** reveals a remarkable connection between the problems of automata theory and certain traditional problems of mathematical logic. In particular, one can prove the algorithmic solvability of the decision problem of **I**. It is also demonstrated that further attempts (sometimes harmless at first sight) to extend the logical metalanguage, with a view to enlarging the metalanguage itself, lead to a situation in which there is no

synthesis algorithm. This is, in a sense, another argument for the statement that the metalanguage **I** is so to speak the "most expressive" language suitable for the synthesis theory.

We reiterate that McNaughton's determinization theorem and the Büchi-Landweber uniformization theorem (Chapters I, II) are essential components in the solution of the synthesis problem for the metalanguage **I**.

The beginning of Chapter IV is concerned with various formulations of the identification problem for automata. We then proceed (Section IV.2 and, in part, Section IV.3) to the identification problem in its traditional formulation, in which one assumes a given (i.e., usable) upper bound on the number of states. There exists an algorithm which identifies all automata under these conditions. However, the main part of the chapter (Sections IV.4 through IV.11) is devoted to the identification problem when there is no known bound on the number of states. The main result we prove is that, as before, there exist algorithms which, with any preassigned frequency, reconstruct the projected operators ("black box" identification). At the same time we establish bounds for the complexity of these algorithms (for both simple and multiple experiments). The precise formulation and solution of these problems involves the idea of frequency algorithms (i.e., algorithms that produce the correct answer not for all initial data but only for some —presumably a sizable proportion). We shall not consider the general concept of a frequency algorithm in this book, confining ourselves to a few illustrations. An exact definition of this concept presupposes that some frequency concept has been adopted. In our case, this might be, say, the ratio of the number of automata with k states (where k is a fixed number) for which the algorithm gives a correct answer to the number of all automata with k states. We describe special classes of frequency algorithms—the so-called iterative algorithms. In a certain sense, these algorithms involve incomplete induction. We shall show that for any $\varepsilon > 0$ there exists an iterative algirithm that performs correct identification with frequency $1 - \varepsilon$, and the choice of this algorithm is in a certain sense uniform.

It should be clear from this survey that the principal subject matter of the book is, in one way or another, connected with the construction of algorithms. In the problems in question, the very existence of an algorithm is far from trivial. Of course, we do not want an algorithm merely to exist —we would like it to be a "good" algorithm. There are various criteria for the quality of an algorithm; they may be formulated and investigated in the theory of complexity of algorithms. We shall not discuss bounds on the complexity of synthesis algorithms in the metalanguage situation. For

identification algorithms we do present bounds, even quite precise ones. We estimate the amount of information that the designer receives from the user; more precisely, we estimate the maximal length of the input words according to which questions are answered, and give fairly precise bounds for almost all automata. In so doing we make essential use of the statistical bounds for parameters and spectra for the behavior of automata established in Chapter V. It is worth mentioning here that the material of Chapter V should logically precede that of Chapter IV; together with Sections II.11 to II.13 it forms, in effect, an independent part of the book, dealing with parameters and spectra for behavior. However, we believe this more logical order unsuitable, inasmuch as the proofs collected in Chapter V involve long and sometimes complicated combinatorial computations. The order adopted is more convenient for the reader and, in effect, Chapter V is in the nature of an appendix which presents the proofs of various bounds and inequalities already used in Chapter IV.

In actual fact, these proofs yield more information than that directly derived from the statements of the theorems. Thus, for example, in deriving a statistical bound for the degree of accessibility, we in fact establish certain statistical regularities in stochastic graphs, which have applications outside the theory of automata (e.g., in the theory of epidemics).

Notes

The concept of a finite automaton is already implicit in Turing [126], in his description of the computing machines now known as Turing machines. The Turing machine is a more general object than the finite automaton, because the reading and writing heads moving along the tape can both change direction and erase the symbol recorded in a square of the tape.

Though finite automata are much simpler than Turing machines, they were not systematically studied until the fifties (not counting the early paper of McCulloch and Pitts [101]). A considerable part of the collection *Automata Studies* [1], translated into Russian in 1956, is already devoted to finite automata (the Russian translation includes the paper [101] and an appendix by Medvedev [48]). The papers present several similar modifications of the concept "finite automaton," but there is no unified terminology.

Later monographs and survey articles, in both Russian and other languages, maintain and even intensify the lack of a coordinated terminology

(unfortunately, this applies to the present book as well). The following terminological clarifications are thus in order.

Several papers consider objects which, apart from the properties we have included in the concept "finite automaton," possess more specific structure (with indications of their construction as combinations of elementary components); various terms are employed—nerve net [101], finite automaton [100], logical net [79]. In particular, the book [7] defines a "finite automaton" as a combination of two objects, one of which defines the functioning of the automaton and is identical to the concept studied in this book, while the other (logical net) specifies the structure of the automaton.

As authors gradually came to realize the fruitfulness of studying the behavior of automata without regard for their structure, they adopted concepts of the type presented above in Sections 0.1 and 0.2. A frequently used synonym for "finite automaton" is "sequential machine" (e.g., in [2]). However, Moore himself, who originated this term, uses it [108] in the sense of our "automaton with delay." On the other hand, in [48] and [2] "finite automaton" is used for our "finite outputless automaton."

Our definitions are slightly different from those by Glushkov in his monograph [6]. Glushkov measures time for states from zero, and for input and output symbols from unity, so that the recurrence relations for an operator become

$$\begin{aligned} q(0) &= q_0, \\ q(t) &= \Psi[q(t-1), x(t)], \\ y(t) &= \Phi[q(t-1), x(t)], \qquad t = 1, 2, \ldots \end{aligned} \tag{1}$$

In particular, the recurrence relations for a Moore automaton (without delay) become

$$\begin{aligned} q(0) &= q_0 \\ q(t) &= \Psi[q(t-1), x(t)], \\ y(t) &= \lambda[q(t)], \qquad t = 1, 2, \ldots \end{aligned} \tag{2}$$

After publication of Glushkov's monograph [6], the terms "Mealy automaton" and "Moore automaton" came into extensive use for cases (1) and (2), respectively.

The earliest meaning of the expression "behavior of an automaton" was what we have called "finite behavior," manifested in the transformation the automaton applies to words or in the set of words that it represents.

Burks and Wright [79] were apparently the first to study the infinite behavior of an automaton with output, i.e., operators transforming infinite input sequences (which we call ω-words) into infinite output sequences.

This motivated further development of the approach by Trakhtenbrot [56, 57]. This infinite behavior of outputless automata was apparently first considered in a systematic manner by Muller [109]; our definition (Section 0.2) of the representability of an ω-language is due to McNaughton [102].

CHAPTER I

BEHAVIOR OF OUTPUTLESS AUTOMATA

I.1. Representation of languages and ω-languages in automata

If no restrictions are imposed on the automaton and the anchoring procedure, it is easily seen that any language may be represented in a suitable anchored automaton. Indeed, let $\mathfrak{A}$ be a language over an alphabet X containing m letters. We construct a representing automaton $\langle \mathfrak{M}, q_0, Q' \rangle$ as follows. The diagram of the automaton is an infinite tree, with exactly m edges, labeled by the input letters, issuing from each vertex. The root of the tree represents the initial state q_0. The set Q' is assumed to contain all states q (and only such states) such that $\mathfrak{A}$ contains a word p taking the state q_0 into q. The class of ω-languages representable in automata is also too large (again we mean arbitrary automata and arbitrary macroanchoring).

There are various types of restriction on the automata and the anchoring procedure which yield less trivial classes of languages and ω-languages. Examples of such classes are the class ω of all languages representable in finite automata (*finite-state languages*) and the class Ω of all ω-languages representable in finite automata (*finite-state ω-languages*). Examples of finite-state languages and ω-languages were given in Chapter 0. A better idea of the classes ω and Ω may be derived from the examples and theorems considered below.

EXAMPLE 1. Any language consisting of a single word $x = x(1)x(2)\ldots x(r)$ is finite-state. A representing automaton may be constructed with $r + 2$ states $q_1, q_2, \ldots, q_r, q_{r+1}, q_{r+2}$, where q_1 is the initial state and q_{r+1} the only final state. The instructions are $q_1x(1) \to q_2$, $q_2x(2) \to q_3, \ldots, q_rx(r) \to \to q_{r+1}$, and, for any other pair $q_\alpha x_\beta$ not in the left-hand side of any of these instructions, an additional instruction $q_\alpha x_\beta \to q_{r+2}$.

REMARK. The state q_{r+2} has the property that any input letter (and so any input word) takes it back to q_{r+2} (i.e., to the same state). We shall call states possessing this property *absorbing* states. They will be used again in various constructions.

EXAMPLE 2. Any ω-language consisting of a single periodic ω-word $x = x(1)x(2)\dots x(r)x(r+1)x(r+2)\dots x(r+s)\dots$ with phase $x(1)\dots x(r)$ and period $x(r+1)\dots x(r+s)$ is finite-state. A representing automaton may be constructed with $r+s+1$ states $q_1, \dots, q_r, q_{r+1}, \dots, q_{r+s}, q_{r+s+1}$, where q_1 is the initial state. There is one limit macrostate—the set $\{q_{r+1}, q_{r+2}, \dots, q_{r+s}\}$, and the instructions are

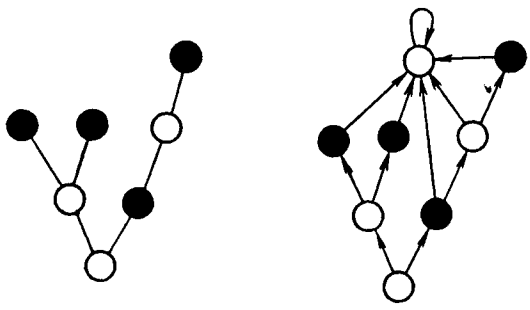
$$q_i x(i) \to q_{i+1} (i < s + r), \quad q_{r+s} x(r+s) \to q_{r+1},$$

supplemented by instructions $q_\alpha x_\beta \to q_{r+s+1}$ for all pairs $q_\alpha x_\beta$ which are not left-hand sides of the above instructions (i.e., q_{r+s+1} is an absorbing state).

EXAMPLE 3. Every finite language is finite-state. The representing automaton is more conveniently described in terms of its diagram; we illustrate the procedure for the language $\mathfrak{A} = \{00, 01, 1, 111\}$. Construct a finite tree for this set of words, as illustrated in Figure 6*a*, where each filled circle (vertex) corresponds to a word in $\mathfrak{A}$. Complete this tree to a diagram by adding a new vertex, to which all missing edges are directed (see Figure 6*b*, where pairs of "parallel" edges have been replaced by a single edge). The root of the tree represents the initial state, the filled circles the final states.

Analogously, one can show that any finite ω-language consisting of periodic ω-words alone may be represented by a macroanchored finite automaton. That a suitable diagram can be constructed follows from Theorem 1.1 below.

To obtain more precise information on the size of the classes ω and Ω, we shall try to find operations over languages (ω-languages) under which these classes are closed. We first consider the set-theoretic operations:



a *b*

Figure 6

union $\cup$, intersection $\cap$, complementation $\neg$, projection and cylindrification. The first three operations are defined as usual; their relation to the logical operations (disjunction, conjunction and negation) is clear from the definitions:

$$x \in \mathfrak{A} \cap \mathfrak{B} \underset{df}{=} x \in \mathfrak{A} \,\&\, x \in \mathfrak{B},$$

$$x \in \mathfrak{A} \cup \mathfrak{B} \underset{df}{=} x \in \mathfrak{A} \vee x \in \mathfrak{B},$$

$$x \in \neg\, \mathfrak{A} \underset{df}{=} \neg (x \in \mathfrak{A}).$$

We recall the definitions of the last two operations. For each word over the alphabet $X \times Y$

$$\langle x(1)y(1)\rangle \langle x(2)y(2)\rangle \ldots \langle x(t)y(t)\rangle$$

its *projections* onto X and Y are defined as the words

$$x(1)\ldots x(t), \quad y(1)\ldots y(t),$$

respectively. The projections of an ω-word are defined similarly. Given a language (ω-language) $\mathfrak{A}$ over an alphabet $X \times Y$, the *projection* of $\mathfrak{A}$ onto X is the language (ω-language) consisting of the projections of all words (ω-words) of $\mathfrak{A}$ onto X. Given an alphabet Y and a language (ω-language) $\mathfrak{A}$ over an alphabet X, the *Y-cylinder* of the language (ω-language) $\mathfrak{A}$ is defined as the language (ω-language) consisting of all words (ω-words) over $X \times Y$ whose X-projections are in $\mathfrak{A}$. The relation between projection and existential quantification should be quite clear. To make the description more precise we introduce the concept of the convolution of words (ω-words). The *convolution* of two words of the same length, $x = x(1)x(2)\ldots x(t)$, $y = y(1)y(2)\ldots y(t)$, is the word $\langle x(1)y(1)\rangle \cdot \cdot \langle x(2)y(2)\rangle \ldots \langle x(t)y(t)\rangle$ over the direct product of the corresponding alphabets. The convolution is denoted by $x * y$. It is now clear that a word (ω-word) belongs to the projection $\mathfrak{B}$ of a language (ω-language) $\mathfrak{A}$ if and only if

$$x \in \mathfrak{B} \underset{df}{=} \exists y \{x * y \in \mathfrak{A}\}.$$

It is useful to note the relation between cylindrification operation and the operation, frequently used in logic, of introducing dummy variables.

For the above operations we have the following theorem, whose statement and proof apply equally well to languages and ω-languages.

THEOREM 1.1. *There exist algorithms which solve the following problems:* (I) *Given a finite automaton* $\langle \mathfrak{M}, q_0, Q\rangle$ ($\langle \mathfrak{M}, q_0, \mathfrak{C}\rangle$), *construct a finite*

automaton representing the language $\neg\,\omega(\mathfrak{M}, q_0, Q)$ *(ω-language* $\neg\,\Omega(\mathfrak{M}, q_0, \mathfrak{C})$*).*

(II) *Given* $\langle \mathfrak{M}', q_0', Q' \rangle$ *and* $\langle \mathfrak{M}'', q_0'', Q'' \rangle$ ($\langle \mathfrak{M}', q_0', \mathfrak{C}' \rangle$ *and* $\langle \mathfrak{M}'', q_0'', \mathfrak{C}'' \rangle$) *over the same alphabet X, construct a finite automaton* $\langle \mathfrak{M}, q_0, Q \rangle$ ($\langle \mathfrak{M}, q_0, \mathfrak{C} \rangle$) *representing the language* $\omega(\mathfrak{M}', q_0', Q') \cup \omega(\mathfrak{M}'', q_0'', Q'')$ *(ω-language* $\Omega(\mathfrak{M}', q_0', \mathfrak{C}') \cup \Omega(\mathfrak{M}'', q_0'', \mathfrak{C}'')$*).*

(III) *Given a finite automaton* $\langle \mathfrak{M}, q_0, Q' \rangle$ ($\langle \mathfrak{M}, q_0, \mathfrak{C} \rangle$) *and an alphabet Y, construct a finite automaton* $\langle \mathfrak{N}, q_0, Q' \rangle$ ($\langle \mathfrak{N}, q_0, \mathfrak{C} \rangle$) *representing the Y-cylinder of the language* $\omega(\mathfrak{M}, q_0, Q')$ *(ω-language* $\Omega(\mathfrak{M}, q_0, \mathfrak{C})$*).*

Proof. We carry out the proof for ω-languages. Part (I) is obvious, for it suffices to replace $\mathfrak{C}$ in the given automaton by the set of all nonempty macrostates not in $\mathfrak{C}$. The automaton $\langle \mathfrak{M}, q_0, \mathfrak{C} \rangle$ of part (II) is constructed as follows. $\mathfrak{M}$ is the direct product of the automata $\mathfrak{M}', \mathfrak{M}''$ with identified inputs (see Section 0.4 of the preceding chapter). The initial state q_0 is $\langle q_0' q_0'' \rangle$. Now let $\widetilde{Q}$ be any macrostate of the automaton $\mathfrak{M}$. The set of all states of $\mathfrak{M}'$ which are first elements of pairs in $\widetilde{Q}$ is the projection of $\widetilde{Q}$ onto the state set of $\mathfrak{M}'$; denote this set by Q'. Similarly, Q'' is the projection of $\widetilde{Q}$ onto the state set of the automaton $\mathfrak{M}''$. Stipulate that the macrostate $\widetilde{Q}$ belongs to $\mathfrak{C}$ if and only if the disjunction $Q' \in \mathfrak{C}' \vee Q'' \in \mathfrak{C}''$ is true. This gives a macroanchored automaton $\langle \mathfrak{M}, q_0, \mathfrak{C} \rangle$, and it can be readily seen that $\Omega(\mathfrak{M}, q_0, \mathfrak{C})$ is precisely

$$\Omega(\mathfrak{M}', q_0', \mathfrak{C}') \cup \Omega(\mathfrak{M}'', q_0'', \mathfrak{C}'').$$

(III) The automaton $\langle \mathfrak{M}, q_0, \mathfrak{C} \rangle$ representing an ω-language $\mathfrak{A}$ over an alphabet X can be converted into an automaton $\langle \mathfrak{N}, q_0, \mathfrak{C} \rangle$ representing the corresponding Y-cylinder by the following procedure—*introduction of multiple edges.* The vertices in the diagram of $\mathfrak{N}$ are the same as those of the diagram of $\mathfrak{M}$. If there is an edge from q_i to q_j in $\mathfrak{M}$, labeled by the letter x_k, then, for every letter $y_i \in Y$, put an edge in $\mathfrak{N}$ from q_i to q_j and label it by the pair $\langle x_k y_i \rangle$. The limit macrostates are as before. This proves the theorem.

REMARKS. I. The proof implies the following bounds on the number of states of the resultant automaton (in terms of the number of states of the original automata)*: complementation and cylindrification do not increase the number of states; union requires at most the product of the numbers of states in the given automata.

II. Since the intersection $\mathfrak{A}_1 \cap \mathfrak{A}_2$ may be expressed in terms of union and complement, $\mathfrak{A}_1 \cap \mathfrak{A}_2 = \neg(\neg\,\mathfrak{A}_1 \cup \neg\,\mathfrak{A}_2)$, the class of finite-state

* We are referring to finite automata.

languages (ω-languages) is also closed under intersection, and the bound on the number of states is the same as for union.

III. Our theorem does not mention closure under projection, which is much more difficult to treat and requires special consideration; we shall return to it later.

I.2. Interchangeability

Finite-state languages may be characterized in terms of interchangeability of words. We first introduce the required concepts. Assume given an arbitrary language $\mathfrak{A}$ over an alphabet X and two (possibly empty) words p, r over the same alphabet.

The words p and r are said to be *interchangeable* in the language $\mathfrak{A}$ (notation: $p \approx r(\mathfrak{A})$) if, for any (possibly empty) words x and y, $xpy \in \mathfrak{A} \equiv \equiv xry \in \mathfrak{A}$. The words p and r are said to be *left-interchangeable* (*right-interchangeable*) in $\mathfrak{A}$ (notation: $p \approx_L r$, $p \approx_R r$) if, for any word x,

$$px \in \mathfrak{A} \equiv rx \in \mathfrak{A} \qquad (xp \in \mathfrak{A} \equiv xr \in \mathfrak{A}).$$

Interchangeability is an important concept in mathematical linguistics, where it is interpreted as syntactic equivalence of words (or word combinations).

The following propositions follow directly from the definition.

I. *Interchangeability* (*left interchangeability*, *right interchangeability*) *is an equivalence relation.* Consequently, it partitions the universal language over X^* into interchangeability classes (left, right interchangeability classes). Its index** may be infinite, as evidenced by the following example. Let the language $\mathfrak{B}$ consist of all words whose lengths are perfect squares. It is then easily seen that only words of the same length are interchangeable. Since interchangeability implies right (left) interchangeability, the partition into interchangeability classes is finer than the partition into right (left) interchangeability classes. In other words, every right (left) interchangeability class is either an interchangeability class or the union of several such classes.

II. *Let* $p \approx_L r$; *then* $px \approx_L rx$ *for any* x. In particular, if the same letter is added to left-interchangeable words on the right, the resulting words are again left-interchangeable. We can therefore define multiplication of a left

* That is, the set of all words over X.

** The index of an equivalence relation is the number of its equivalence classes.

interchangeability class $\mathfrak{A}_i$ by any letter a: the product $\mathfrak{A}_i a$ is the left interchangeability class that contains all words of the form pa for $p \in \mathfrak{A}_i$.

III. *The language $\mathfrak{A}$ is either the union of several interchangeability (right interchangeability, left interchangeability) classes, which generate it, or a single such class.*

The following theorem reveals the relation between the index of left interchangeability in a language and the number of states in an automaton representing it.

THEOREM 1.2. (I) *If $\mathfrak{M}$ is an automaton representing a language $\mathfrak{A}$ with left interchangeability of index μ, the number of states in $\mathfrak{M}$ is at least μ.*

(II) *Any language $\mathfrak{A}$ with left interchangeability of index μ is representable by an automaton with μ states.*

Proof. (I) Let $\mathfrak{A} = \omega(\mathfrak{M}, q_0, Q')$. Fix some $q_i \in Q'$, and note that if $p \in \omega(\mathfrak{M}, q_0, q_i), r \in \omega(\mathfrak{M}, q_0, q_i)$, then $p \approx_L r(\mathfrak{A})$, since for any word x the words px and rx take q_0 to the same state. Thus, the partition of the universal language into languages $\omega(\mathfrak{M}, q_0, q_i)$* is finer than the partition into left interchangeability classes. It follows that the number of these classes is at most the number of states q_i in Q'.

(II) Let $\mathfrak{A}$ generate a finite system of left interchangeability classes $\mathfrak{A}_0, \ldots, \mathfrak{A}_{\mu-1}$, where $\mathfrak{A}_0$ denotes the class of all words interchangeable with the empty word.

Now define an automaton $\mathfrak{M}$ as follows: The states $q_0, q_1, \ldots q_{\mu-1}$ correspond to the classes $\mathfrak{A}_0, \mathfrak{A}_1, \ldots, \mathfrak{A}_{\mu-1}$. Define $q_i a \to q_j$ if $\mathfrak{A}_i a = \mathfrak{A}_j$ (see Proposition II before the theorem). It is easily seen that the language $\omega(\mathfrak{M}, q_0, q_i)$ coincides with the class $\mathfrak{A}_i$. Let $\mathfrak{A}$ be the union of classes $\mathfrak{A}_{i_1}, \ldots, \mathfrak{A}_{i_s}$ (see Proposition III). Denoting $\{q_{i_1}, \ldots, q_{i_s}\}$ by Q', we finally get $\mathfrak{A} = \omega(\mathfrak{M}, q_0, Q')$. Q.E.D.

COROLLARY. *$\mathfrak{A}$ is a finite-state language if and only if it generates a finite system of left interchangeability classes.*

REMARK. The *reflection* $\mathfrak{A}^{-1}$ of a language $\mathfrak{A}$ is the language consisting of all words obtained by writing the words of $\mathfrak{A}$ in the reverse order. For example, if $a(1)a(2)\ldots a(n) \in \mathfrak{A}$, then $a(n)\ldots a(2)a(1) \in \mathfrak{A}^{-1}$. Let the index of the right interchangeability generated by $\mathfrak{A}$ be μ; then this is also the index of the left interchangeability of the reflection $\mathfrak{A}^{-1}$. If μ is finite, then

* The simplified notation $\omega(\mathfrak{M}, q_0, q_i)$, $\Omega(\mathfrak{M}, q_0, \Gamma)$, etc., is used for $\omega(\mathfrak{M}, q_0, \{q_i\})$, $\Omega(\mathfrak{M}, q_0, \{\Gamma\})$, etc., where $\{q_i\}$ and $\{\Gamma\}$ are singletons.

$\mathfrak{A}^{-1}$ is a finite-state language. The proof of Theorem 1.2 does not carry over directly to right interchangeability. Later (see Theorem 1.9) we shall see from other arguments that if μ is finite then not only $\mathfrak{A}^{-1}$ but also $\mathfrak{A}$ is a finite-state language, though the representing automaton generally has more than μ states.

The set of all languages over a finite alphabet X is nondenumerable. The set of recursive languages over X, however, is denumerable, and so a fortiori is the set of finite-state languages. Thus, the "overwhelming majority" of languages are not representable by finite automata. Moreover, the class of finite-state languages is a very small subclass of the class of all recursive languages.

Theorem 1.2 and its corollary provide simple examples of recursive languages which are not finite-state. One such language is that ($\mathfrak{B}$) described in Proposition I before Theorem 1.2. Another example is the language $\mathfrak{A}$ consisting of all words 0^n10^n, where 0^n denotes the word consisting of n zeros ($n = 1, 2, \ldots$). For suppose that $\mathfrak{A}$ is a finite-state language. Then the sequence of words $0, 00, 000, \ldots$ must contain a pair of left-interchangeable words (since by the corollary to Theorem 1.2 the number of left-interchangeability classes must be finite). Let $0^k \approx_L 0^s(\mathfrak{A})$. Then, since $\mathfrak{A}$ contains the word 0^k10^k, it must also contain 0^s10^k, which contradicts the definition of the language $\mathfrak{A}$. Thus $\mathfrak{A}$ is not representable in a finite automaton. This reasoning in fact establishes the following general proposition, which can be formulated in terms of separability of languages. We say that a language $\mathfrak{B}$ *separates* a language $\mathfrak{A}_1$ from a language $\mathfrak{A}_2$ (disjoint from $\mathfrak{A}_1$) if $\mathfrak{B} \supseteq \mathfrak{A}_1$ and $\mathfrak{B} \cap \mathfrak{A}_2 = \varnothing$. A language $\mathfrak{A}_1$ is *finite-state separable* from a language $\mathfrak{A}_2$ if there exists a finite-state language $\mathfrak{B}$ separating $\mathfrak{A}_1$ from $\mathfrak{A}_2$ (and then, as is easily seen, $\mathfrak{A}_2$ is also finite-state separable from $\mathfrak{A}_1$ by the language $\neg\mathfrak{B}$, and therefore finite-state separability of languages is a symmetric relation). In our example, the language $\{0^n10^n\}$ is not finite-state separable from the language $\{0^k10^s\}$ ($k \neq s$).

I.3. Distinguishability of words and ω-words

We adopt the following notation. If Q is the internal alphabet of an automaton, $\widetilde{Q}$ is the set of all mappings of Q into itself. The state to which an input word a takes a state q will be denoted by qa. For example, consider the finite automaton $\mathfrak{M}$ defined by Table 5; Figure 7 is its diagram (to avoid unnecessary complications the three edges going from 1 to 2 and labeled

TABLE 5

x \ q	1	2	3	4
a	2	3	4	1
b	2	1	3	4
c	2	2	3	4

a, b, c, respectively, are represented by a single edge labeled by all three letters; we shall use this device in the sequel).

Each input word a may be associated in a natural way with an element of $\widetilde{Q}$—the mapping of Q into itself defined by the product qa, with a fixed and q running through Q. In particular, each row of the transition matrix, corresponding to some input letter x, defines the mapping induced by the one-letter word x. For example, Table 5 shows that a and b induce a one-to-one mapping of the set $\{1, 2, 3, 4\}$ onto itself (a induces a cyclic permutation, b the transposition of $(1, 2)$); the mapping induced by c merges the states 1, 2, i.e., maps them onto the same state.

If x, y are *words* that induce the same mapping, we shall call them *indistinguishable* (by the given automaton $\mathfrak{M}$), denoting this relation by $x \approx y(\mathfrak{M})$; $\mathfrak{M}$ will be omitted when there is no danger of confusion. It is clear that $\approx$ is an equivalence relation, and so it partitions the set of all input words (i.e., the universal language) over the alphabet X into equivalence classes (called indistinguishability classes). If $\mathfrak{M}$ is a finite automaton with n states, the index of this partition if finite. For since the number of all possible mappings of an n-element set Q into itself is n^n, there cannot be more than n^n indistinguishability classes. For an infinite automaton the number of indistinguishability classes may be either finite or infinite.

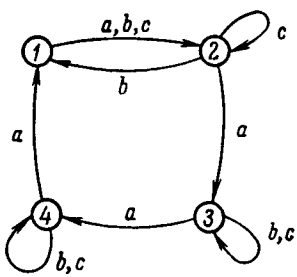


Figure 7

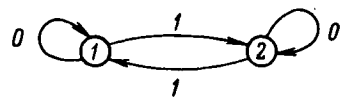


Figure 8

If a is an input ω-word, we retain the notation qa for the macrostate into which a takes the state q. Each input ω-word induces a mapping of the state set Q into the set of all macrostates. If two ω-*words* x and y induce the same mapping, they are said to be *indistinguishable*, and the relation is again denoted by $x \approx y(\mathfrak{M})$. Clearly, $\approx$ is again an equivalence relation; if $\mathfrak{M}$ is a finite automaton with k states, the number of equivalence classes (classes of indistinguishable ω-words) is at most $(2^k - 1)^k$.

EXAMPLE. Consider the finite automaton whose diagram is illustrated in Figure 8. There are two classes of indistinguishable words: 1) the language $\mathfrak{A}_1$ consisting of all words with an odd number of ones, and 2) the language $\mathfrak{A}_2$ consisting of all words with an even number of ones. The corresponding mappings of Q into itself are $\begin{pmatrix} 1 & 2 \\ 1 & 2 \end{pmatrix}$ and $\begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}$. There are three classes of indistinguishable ω-words: the ω-language $\mathfrak{B}_\infty$ of all ω-words with infinitely many ones, the ω-language $\mathfrak{B}_1$ of ω-words with a finite and odd number of ones, and the ω-language $\mathfrak{B}_2$ of all ω-words with a finite and even number of ones. For

$$\begin{aligned} a &\in \mathfrak{B}_\infty, \quad 1a = 2a = \{1, 2\}, \\ a &\in \mathfrak{B}_1, \quad 1a = 2;\ 2a = 1, \\ a &\in \mathfrak{B}_2, \quad 1a = 1;\ 2a = 2. \end{aligned}$$

It follows directly from the definition that generally $\omega(\mathfrak{M}, q_0, Q')$ is the union of certain indistinguishability classes of $\mathfrak{M}$ (in particular, it may be a single class). Similarly, it is clear that the ω-language represented by a macroanchored automaton is the union of certain indistinguishability classes of ω-words. The relation between the input words and the mappings of Q into itself that they induce may be characterized more precisely in algebraic terms.

Recall that the set of all input words is a finitely generated semigroup $\tilde{A}$ with respect to concatenation. Now the set $\tilde{Q}$ of all mappings of Q into itself is also a semigroup with identity, under the binary operation defined by composition of mappings; the identity element is the identity mapping of Q into itself. Since $(qa)b = q(ab)$, it is quite clear that the above correspondence between input words and mappings of Q into itself is a homomorphism of the semigroup $\tilde{A}$ into the semigroup $\tilde{Q}$. The image $\tilde{Q}' \subseteq \tilde{Q}$ of $\tilde{A}$ under this homomorphism is a subsemigroup of $\tilde{Q}$. If A is a finite alphabet, the semigroup $\tilde{Q}'$ is of course finitely generated: it is generated by all mappings by symbols of A. Hence the study of an outputless automaton amounts to the study of a subsemigroup of $\tilde{Q}$ defined by finitely many

generators of $\tilde{Q}$. These remarks provide the basis for a novel approach to automata theory in which algebraic concepts and methods assume a major role. The algebraic theory of automata is particularly useful in formulating and solving problems in the structural theory, and it lies beyond the scope of this book. Here we confine ourselves to the following observation, which is directly applicable to the type of problem considered here and will be used later (Section I.7). We have already mentioned that the number of indistinguishability classes of a finite automaton with n states is finite, at most n^n. The following theorem shows that this bound is the best possible.

THEOREM 1.3. *For any n, there exists an automaton $\mathfrak{M}_n$ with n internal states and three input letters a, b, c for which the number of indistinguishability classes is exactly n^n.*

Proof. The proof is based on the following well-known (and easily verified) assertions.*

1. Consider the cyclic permutation

$$a = \begin{pmatrix} 123 \ldots n-1 & n \\ 234 \ldots & n \quad 1 \end{pmatrix}$$

and any transposition, e.g.,

$$b = \begin{pmatrix} 123 \ldots n-1 & n \\ 213 \ldots & n-1 & n \end{pmatrix},$$

in the set of all permutations of n elements $J_n = \{1, 2, \ldots, n\}$. Then every permutation may be represented as a product of permutations $\alpha_1 \alpha_2 \ldots \alpha_s$, where $\alpha_i (i \leqq s)$ is either a or b.

2. Let c be a fixed function mapping J_n onto a subset of itself containing $n-1$ elements (i.e., c "merges" exactly two elements). Then any mapping of J_n onto a subset of itself containing m elements ($n > m$) is a product $\beta_1 \beta_2 \ldots \beta_\nu$, where $\beta_i (i \leqq \nu)$ is either a permutation or the mapping c (this product contains exactly $n-m$ factors equal to c, carrying out the $n-m$ necessary "mergings").

Thus any mapping of J_n onto itself or onto a proper subset of itself may be expressed as a product of mappings $\gamma_1 \gamma_2 \ldots \gamma_\mu$, where each γ_i is the cyclic permutation a, the transposition b or a mapping of type c.

Now consider the automaton $\mathfrak{M}_n$ with transition matrix given by Table 6a. It is clear that the input letters a, b, c, induce precisely those mappings of the set of n states that we have just denoted by the same letters. Con-

* See, e.g., A. G. Kurosh, Theory of Groups (in Russian), p. 42, "Nauka", 1968.

sequently, each of the n^n possible mappings of the state set into itself is induced by some input word, and so the number of indistinguishability classes of the automaton $\mathfrak{M}_n$ is n^n. Q.E.D.

The concepts of indistinguishability (with respect to an automaton $\mathfrak{M}$) and interchangeability are related. Let p, r be words indistinguishable by the automaton $\mathfrak{M}$; then they are interchangeable with respect to any indistinguishability class. In other words, if K is any indistinguishability class, then $\forall xy(xpy \in K \equiv xry \in K)$, i.e., the words xpy and xry induce the same mapping of Q into itself. Therefore p and r are interchangeable with respect to every language represented by some anchoring of the automaton $\mathfrak{M}$.

The converse is also true: if p and r are interchangeable with respect to every language representable by $\mathfrak{M}$, then they are indistinguishable by $\mathfrak{M}$.

TABLE 6a

x \ q	1	2	3	...	$n-1$	n
a	2	3	4	...	n	1
b	2	1	3	...	$n-1$	n
c	2	2	3	...	$n-1$	n

TABLE 6b

x \ q	1	2	3	...	$n-1$	n
a	2	3	4	...	n	1
b	2	1	3	...	$n-1$	n
c	—	$\{1,2\}$	3	...	$n-1$	n

For let $qp = q'$, i.e., $p \in \omega(\mathfrak{M}, q, q')$. Then, by assumption, $r \in \omega(\mathfrak{M}, q, q')$ and this means that $qr = q'$.

Thus, the indistinguishability of two words with respect to an automaton $\mathfrak{M}$ is equivalent to their interchangeability with respect to every language representable by $\mathfrak{M}$.

I.4. Decidability of properties of finite automata

We shall have to deal with various properties of finite automata (anchored finite automata) or finite systems of finite automata. We shall say that a prop-

erty Φ is *effectively decidable** (or simply *decidable*) if there exists an algorithm which, given any automaton $\mathfrak{M}$ (finite system of automata $\mathfrak{R}$), determines whether $\mathfrak{M}(\mathfrak{R})$ possesses property Φ or not. The automaton may be defined by its diagram (transition matrix), an anchored automaton by a diagram with designated initial and final states. We shall be able to prove the decidability of many properties. In so doing we shall base the decision algorithms on an essentially simple (though cumbersome) procedure—the so-called procedure of word enumeration—which will now be described.

Assume given any finite diagram Γ over an alphabet X and two designated sets of vertices (states) Q' and Q''. We shall consider problems of the following type:

(I) To determine whether there exist paths from Q' to Q'' (in other words, is any vertex in Q'' accessible from some vertex in Q').

(II) If such paths exist, to find at least one of them and the word it carries.

Obviously, if there exists a path from Q' to Q'', there exists also a simple path, whose length does not exceed the number of vertices in the diagram. It is clear that by checking all simple paths in the diagram (for their number is finite!) one can determine which of them go from Q' to Q''; whenever a path is found with the required property, one can record the word that it carries. Of course, this procedure is extremely cumbersome for large diagrams, though quite feasible in principle. We shall not deal here with devices for simplifying the procedure. The only fact of immediate interest is the very existence of an algorithm solving problems of this type. This procedure (word enumeration) is often used in the theory of algorithms as a constituent part of other algorithms.

The idea will be illustrated by the theorems proved below, which are also of independent interest.

THEOREM 1.4. *The following properties of anchored finite automata are decidable:*

(I) *The language represented by the automaton is nonempty.*

(II) *The language represented by the automaton is infinite.*

Proof. Given an automaton $\langle \mathfrak{M}, q_0, Q' \rangle$. It will obviously suffice to prove the decidability of properties (I), (II) for every $\langle \mathfrak{M}, q_0, q_j \rangle$, where $q_j \in Q'$. Property (I) holds if and only if the flow from q_0 to q_j is nonempty. This is effectively decidable, and if the answer is positive a word from $\omega(\mathfrak{M}, q_0, q_j)$ is effectively constructible.

* *Translator's note*: Russian original "recognition," "recognizable."

Property (II) holds if and only if there exist a word p and a state $q_n \in Q$ such that $p = p_1p_2p_3$, where p_1 takes q_0 into q_n, p_2 takes q_n into q_n, and p_3 takes q_n into q_j. If these p and q exist, the language $\omega(\mathfrak{M}, q_0, q_j)$ always contains the infinite sequence of words

$$p_1p_2p_3, \qquad p_1p_2p_2p_3, \qquad p_1p_2p_2p_2p_3, \cdots \tag{1}$$

The above argument implies the following decision algorithm for property (II). Let $q_0, q_1, q_2, \ldots$ be all the states of Q. As in the decision procedure for property (I), check whether there exist a) a word taking q_0 into q_1, b) a word taking q_1 into q_1, c) a word taking q_1 into q_j. These words, if they exist (and then they are effectively identifiable!), are the required p_1, p_2, p_3, and so the automaton possesses property (II). If one of these words does not exist, proceed to an analogous check for q_2, and so on. The procedure continues until the required q_n has been found or until its nonexistence has been ascertained. Q.E.D.

REMARK. The above algorithms do not merely decide the properties in question—if the language is not empty they also construct a word in $\omega(\mathfrak{M}, q_0, Q')$ or an infinite sequence of words (if the language is infinite). We have also proved that if an infinite language is finite-state it must contain a subset of words of type (1). This gives another proof of the fact that the language $\{0^m10^m\}$ is not representable by a finite automaton.

As an illustration of Theorem 1.4, consider the automaton $\mathfrak{M}$ whose diagram is given in Figure 7, and anchor it as follows: initial state, 1; final state, 3. Then the language represented by this automaton is not empty, since the vertex 1 is joined to 3 by the path ba. Moreover, the language is infinite: for the words p_1, p_2, p_3 of the theorem take a, c, a, respectively.

THEOREM 1.5. *The following properties of macroanchored finite automata are decidable:*

(I) *The ω-language represented by the automaton is nonempty.*

(II) *The ω-language represented by the automaton is infinite.*

Proof. As in Theorem 1.4, it will suffice to prove the theorem for a set $\mathfrak{C}$ containing only one element (i.e., one macrostate Q'). Without loss of generality, assume that $Q = \{q_1, q_2, \ldots, q_k\}$, $Q' = \{q_1, \ldots, q_s\}$ $(s \leqq k)$. We confine ourselves to an algorithm for property (I). Assume that there exists an ω-word a taking the initial state into the macrostate Q'. Then the following conditions are necessarily satisfied. 1) There exists a word p_0 taking the initial state into q_1. 2) For any $q_i \in Q'$, $q_j \in Q'$, there exists a word

$p_{ij} = a(1)\dots a(t)$ taking q_i into q_j such that all intermediate states in the corresponding internal word $q(1)q(2)\dots q(t)q(t+1)$ are also in Q'. On the other hand, if these two conditions hold, then, as is easily seen, the ω-language represented by the automaton always contains the following ω-word:

$$b = p_0 p_1 p_1 p_1 \dots, \tag{2}$$

where $p_1 = p_{12}p_{23}\dots p_{k-1k}p_{k1}$.

It is clear that verification of conditions 1), 2) and (if these conditions hold) identification of the corresponding words are effective (word enumeration!). This proves the first part of the theorem.

EXAMPLE. We again consider the automaton $\mathfrak{M}$ (Figure 7), with the following macroanchoring: initial state, 1; limit macrostate, $Q' = \{1, 2\}$. Then the ω-language $\Omega(\mathfrak{M}, 1, \{1, 2\})$ is nonempty, since the word $p_0 = aaaab$ takes state 1 into state 2 (which is in Q'), and there is a path, carrying the word $p_1 = bc$, from 2 to 2 via 1. Thus $\Omega(\mathfrak{M}, 1, \{1, 2\})$ contains the ω-word

$$\underbrace{aaaab}_{p_0}\underbrace{bc}_{p_1}\dots\underbrace{bc}_{p_1}\dots$$

REMARKS. I. The above proof also shows that every nonempty finite-state ω-language must contain a periodic ω-word. Thus a one-element ω-language consists of a single periodic ω-word, and no ω-language consisting of a single nonperiodic ω-word (such as 1010010001...) can be finite-state.

II. If Theorems 1.4 and 1.5 are combined with the previously proved closure of the class of finite-state languages (ω-languages) under the set-theoretic operations, one can prove the decidability of various properties of systems of finite automata. For example, given a pair $\mathfrak{M}_1, \mathfrak{M}_2$ of finite automata, one can effectively determine whether the intersection of the languages (ω-languages) represented by $\mathfrak{M}_1$ and $\mathfrak{M}_2$ is empty, finite or infinite.

III. Up to now all properties of automata (systems of automata) that we have considered have turned out to be decidable. However, it would be easy to formulate properties (moreover, in terms similar to those used above) which are not effectively decidable. For the moment we postpone the discussion of these examples for the sole reason that it is easier to formulate them for automata with output, as we shall indeed do in Chapter II.

I.5. Projections, sources, macrosources

We now take up the projection operation. Let $\mathfrak{A}$ be a language over an alphabet $X \times Y$, defined by an automaton $\langle \mathfrak{M}, q_0, Q' \rangle$. Every edge in the diagram of this automaton is labeled by a letter pair $\langle xy \rangle \in X \times Y$. Replace each of these labels by its projection onto X (i.e., in simpler terms, erase the "superfluous" component y), retaining unchanged the diagram and the initial and final vertices. Now the resulting object is *not* the diagram of a finite automaton with input alphabet X, since the first automaton condition (uniqueness) is violated: each vertex of the graph is joined to other vertices by several edges bearing the same letter from X. However, we can still consider paths going from the initial vertex to final vertices and the words that they carry. Moreover, note that the language carried by the flow from q_0 to Q' is precisely the projection of the original language $\mathfrak{A}$. Similarly, whenever the diagram of an automaton representing an ω-language $\mathfrak{A}$ over $X \times Y$ is given, erasure of the superfluous labels (from Y) yields a diagram (not an automaton diagram!) describing the projection of $\mathfrak{A}$ in the above sense.

This observation points to the advantage in studying languages (ω-languages) described by objects which are more general than anchored automata. Objects of this type, to whose definition we now proceed, are sources and macrosources.

Let B be an arbitrary diagram over an input alphabet X; call its vertices *states* and sets of vertices *macrostates*. A *source** $\langle B, Q', Q'' \rangle$ is defined as a diagram B with two designated macrostates: initial macrostate Q' and final macrostate Q''. The language carried by the flow from Q' to Q'' is said to be *represented* by the source $\langle B, Q', Q'' \rangle$, and denoted by $\omega(B, Q', Q'')$.

A *macrosource* $\langle B, Q_0, \mathfrak{C} \rangle$ is defined as a diagram B with a designated (initial) macrostate Q_0 and a designated family $\mathfrak{C}$ of limit macrostates. $\Omega(B, Q_0, \mathfrak{C})$ denotes the ω-language consisting of all ω-words x (and only those) satisfying the condition: there exists an ω-path carrying x whose first vertex is in Q_0 and whose limit set is a macrostate in $\mathfrak{C}$. We shall say that the macrosource $\langle B, Q_0, \mathfrak{C} \rangle$ represents the ω-language $\Omega(B, Q_0, \mathfrak{C})$.

* *Translator's note*: This is known in the Western literature as a *transition graph* or a *nondeterministic automaton*. Burks and Wright [80] consider a similar object which they call a *sequence generator with goals*; as far as we can ascertain, this term has not received general acceptance. An alternative term for "source" might be an *anchored diagram*.

In these terms, we can state that the projection of a finite-state language (ω-language) is a language (ω-language) representable by a source (macrosource). As already mentioned, erasure of the "superfluous" labels leads to a violation of the uniqueness condition. Our definition of sources and macrosources also relaxes other automaton conditions for diagrams, for the following situations are now admissible:

1) Several edges with the same input label (letter from X) issue from a vertex α (nonuniqueness).

2) There are vertices from which issue no edges labeled by a given letter $x \in X$ (incomplete specification).

3) There are several initial vertices.

An example of a source is the diagram of Figure 9*a*, with initial macrostate $\{1,4\}$ and final macrostate $\{2,3,4\}$. There are two edges labeled c issuing from the vertex 2, no edges labeled c from 1. Thus both automaton conditions are violated. The same diagram (Figure 9*a*) is a macrosource with the following anchoring: initial macrostate, $\{1,4\}$; family of limit macrostates, $\{1,2\}$, $\{3\}$, $\{3,4\}$.

Together with the concepts of source and macrosource just defined, it is sometimes convenient to employ slightly more general concepts. These are obtained by allowing certain edges in the diagram to have no input labels whatsoever (empty edges); these edges may be labeled by the "empty" letter $\wedge$. Given a finite path of length v in such a diagram, one "computes" the word that it carries as follows: write all nonempty labels of the edges in the path in sequence, omitting all empty labels. If all edges in the path are empty, it does not carry any word. Now consider any fixed ω-path. The corresponding ω-word is obtained by writing all nonempty labels in sequence, provided that there are infinitely many nonempty labels. But if all edges of the ω-path are empty from some position on, the ω-path does not carry any ω-word. Otherwise, the concept of the language (ω-language) represented by a source (macrosource) with empty edges is defined as usual as the language (ω-language) carried by a suitable flow with the given anchoring.

An example of a source with empty edges is given by Figure 9*b*, with the anchoring: initial macrostate, $\{1,4\}$; final macrostate, $\{3,4\}$.

REMARKS. I. Any diagram over an alphabet X, containing no empty edges, can be associated with a system of instructions: If there is an edge labeled x going from a vertex q to a vertex q', this edge corresponds to an instruction $qx \to q'$. One can thus interpret any diagram as a certain object

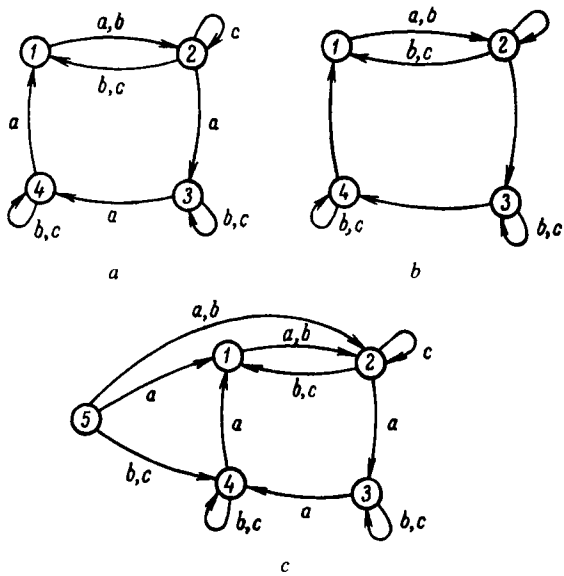


Figure 9

(known as a *nondeterministic automaton*) of more general properties than the automata considered hitherto. Recall that an automaton functions in a strictly deterministic manner: at each sampling time the next state is uniquely determined by the present state and the symbol received. By contrast, a nondeterministic automaton generally admits some freedom in the choice of the next state (nonuniqueness). Suppose the word p is recorded on the tape, and the reading head of a nondeterministic automaton, starting in one of the initial states, begins to read the word. It may happen that some sequence of choices "blocks" the automaton (when the letter a is observed in some state π, but there is no instruction with left-hand side πa). Another possibility is that the word is read in its entirety, but the automaton ultimately reaches a nonfinal state. The word p may nevertheless belong to the language represented by the (suitably anchored) nondeterministic automaton, if there is at least one sequence of admissible choices of states for which the reading ends in a final state.

One can formulate an analogous "automaton" interpretation for the representation of ω-languages in macrosources. Nondeterministic automata should not be confused with probabilistic automata (see Section I.11).

II. As in the case of automata, the definition of languages representable in a source may be modified to include languages with the empty word:

the empty word is considered to belong to the language if at least one of the following conditions holds:

(A) There is a vertex which is both initial and final.

(B) There exists a (nonzero) path from some initial vertex to a final vertex, all of whose edges are empty (condition (B) is meaningful only for sources which admit empty edges).

With this definition, consider the source illustrated in Figure 9*c* with the anchoring: initial macrostate, $\{1, 4\}$; final macrostate, $\{1, 2\}$. This source represents a language containing the empty word.

III. Obviously, the word-enumeration procedure described in Section I.4 and employed there to decide properties of finite automata is also applicable to finite sources (macrosources), for it is based only on the fact that the relevant graphs are finite. Thus Theorems 1.4 and 1.5 remain valid for finite sources (macrosources). A slight modification of the proofs is necessary only for sources (macrosources) with empty words.

Sources (including anchored automata) are said to be *equivalent* if they represent the same language. Similarly, macrosources (including macro-anchored automata) are *equivalent* if they represent the same ω-language.

Our next problem is to establish certain simple rules for converting given sources (macrosources) into equivalent sources (macrosources). These transformations will often prove useful. We begin with a few definitions.

We define an *input* (*output*) *terminal* of a diagram to be any vertex which is not the endpoint (starting point) of any edge. Call a source a *two-terminal source* (or simply *two-terminal*) if it has a single initial vertex and a single final vertex, and these vertices are input and output terminals, respectively.

Assume given an arbitrary source (macrosource) with several initial states $q_1, q_2, \ldots, q_s$. An equivalent source (macrosource) with a single initial state, which is an input terminal, may be obtained by *contracting initial vertices to one input terminal*; this procedure slightly generalizes the adjunction of an initial state in an automaton (see Section 0.4). Add a new vertex γ_1 to the diagram, drawing edges from it to all vertices which are endpoints of edges issuing from the initial vertices of the diagram, with the same respective labels. Define γ_1 to be the sole initial vertex of the new diagram; the final vertices (limit macrostates) are as before. The resulting source (macrosource) is clearly equivalent to the original one (see, e.g., the diagram of Figure 9*c*, which is obtained from that of Figure 9*a* by contracting the initial vertices 1, 4 to a terminal 5). Analogously, several final vertices in a source may be contracted to an output terminal. Applying

both contraction procedures to a source in turn, one can convert it into an equivalent two-terminal. We thus have the following simple theorem, which will be used in the following sections.

THEOREM 1.6. *There exists an algorithm which, given any finite source (macrosource), will construct an equivalent two-terminal (macrosource with single initial state—input terminal).*

This algorithm for construction of an equivalent two-terminal complicates the source by the addition of new vertices and edges. In practice it is often useful to carry out what is, in a certain sense, the inverse procedure: "simplification" of the source (macrosource). By deleting certain vertices, together with all edges issuing from them, the source (macrosource) is converted into an equivalent source (macrosource). Let us consider a few cases in which this is admissible (i.e., the procedure yields an equivalent source). In each case we shall specify where to connect edges previously connected to the deleted vertex. It is left to the reader to verify that these transformations are admissible.

I. *Deletion of inaccessible vertices.* Call a vertex q' *inaccessible* from q if there are no paths from q to q'. If a vertex q is inaccessible from all initial vertices, it may be deleted. The edges previously reaching it may be connected arbitrarily. For example, the vertex 7 in the source $\langle \tilde{\mathfrak{M}}, 1, 3, \rangle$ of Figure 11*b* is inaccessible from the initial vertex 1, and may therefore be deleted together with the edges a and b that issue from it.

II. *Merging of equivalent vertices.* Call two vertices q and q' of a diagram *equivalent** if there exists a one-to-one correspondence between the edges issuing from q and those issuing from q', under which every two corresponding edges have the same label and lead to the same vertex.

If equivalent vertices q and q' of a source are either both final or both nonfinal, one of them may be deleted. Edges reaching the deleted vertex are connected to the remaining vertex of the pair. This procedure is known as "merging of equivalent vertices" q and q'. For example, the vertices q_3 and q_5 in the automaton $\langle \tilde{C}, q_1, q_4 \rangle$ of Figure 10*d* are equivalent. Merging of q_3 and q_5 gives the automaton $\langle C, q_1, q_4 \rangle$ of Figure 10*e*.

III. *Merging of absorbing vertices.* Let q and q' be absorbing vertices of a source, satisfying the following conditions: there exists a one-to-one correspondence between the edges issuing from q and q' such that corresponding edges have the same labels. If these vertices are either both final

* *Translator's note*: in Russian, "twins."

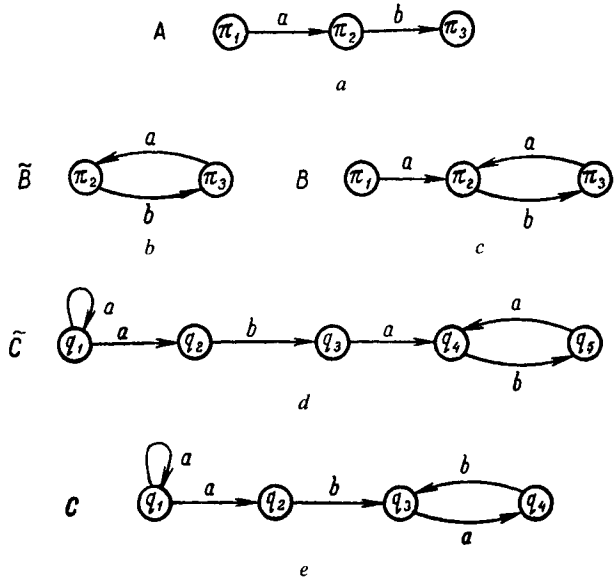


Figure 10

or both nonfinal, any one of them may be deleted. Edges previously reaching the deleted vertex are connected to the remaining absorbing vertex. Example: Merging of the absorbing vertices v_3 and v_6 in the source $\langle \mathfrak{B}', v_1, v_4 \rangle$ of Figure 12 *b* yields the equivalent source $\langle \mathfrak{B}'', v_1, v_4 \rangle$ of Figure 12 *c*.

The analogous transformations for macrosources are easily shown to be admissible:

I′. *Deletion of inaccessible vertices*—exactly as before.

II′. *Merging of equivalent vertices* is admissible when neither of them is an element of a limit macrostate.

III′. *Merging of absorbing vertices* q and q' is admissible if $\{q\}$ and $\{q'\}$ are either both limit macrostates or both nonlimit macrostates.

There is yet another simplification procedure for macrosources:

IV. *Elimination of fictitious macrostates.* Given a macrosource $\langle B, Q', \mathfrak{C} \rangle$, it may turn out that the ω-language $\Omega(B, Q', \Gamma)$ is empty for some $\Gamma \in \mathfrak{C}$. In a finite macrosource such *fictitious* limit macrostates may obviously be effectively found (see Remark III, p. 43) and eliminated from $\mathfrak{C}$. For example, in the macrosource $\langle \mathfrak{B}', v_1, \{\{v_1, v_2\}, \{v_4, v_5\}\} \rangle$ of Figure 12*b* the limit macrostate $\{v_1, v_2\}$ is fictitious, and so

$$\Omega(\mathfrak{B}', v_1, \{\{v_1, v_2\}, \{v_4, v_5\}\}) = \Omega(\mathfrak{B}', v_1, \{v_4, v_5\}).$$

I.6. Operations on sources (macrosources) and on the languages (ω-languages) represented by them

The general nature of the concepts "source" and "macrosource" often makes it very easy to prove that the class of languages (ω-languages) representable in finite sources (macrosources) is closed under various operations. For example, consider the sources

$$\mathfrak{B}_1 = \langle B_1, Q_1', Q_1'' \rangle, \qquad \mathfrak{B}_2 = \langle B_2, Q_2', Q_2'' \rangle$$

or macrosources

$$\langle B_1, Q_1', \mathfrak{C}_1 \rangle, \qquad \langle B_2, Q_2', \mathfrak{C}_2 \rangle.$$

To obtain a source (macrosource) representing the union of the corresponding languages (ω-languages), we need only combine the diagrams B_1 and B_2 into one diagram, the sets of initial states into a single set of initial states $Q_1 \cup Q_2$, and the sets of final states (limit macrostates) into a single set. This construction is clearly simpler than the multiplication of automata in Theorem 1.1.* Another typical example is the reflection of a language $\mathfrak{A}$, i.e., construction of the language $\mathfrak{A}^{-1}$ consisting of all words of $\mathfrak{A}$ written in the reverse order. All we need do is exchange the roles of the initial and final states in a source representing $\mathfrak{A}$, and change the sense of all edges. The result is obviously a source representing $\mathfrak{A}^{-1}$. However, the complement of a language (ω-language) represented by a given source (macrosource) *cannot* be obtained from the latter by simply using the complement of the set of final states (limit macrostates), as in the case of automata. Here complementation involves far more complicated constructions, which will be considered later. In this section we shall concentrate on operations for which there are simple and obvious constructions, as for union and reflection. All these operations will be defined in terms of concatenation (multiplication) of words or concatenation of a word and an ω-word.

a) *The concatenation product*** *of two languages* $\mathfrak{A}_1$ *and* $\mathfrak{A}_2$ (denoted by $\mathfrak{A}_1 \cdot \mathfrak{A}_2$) is the language consisting of all concatenations $a = a_1 a_2$ of two words $a_1 \in \mathfrak{A}_1, a_2 \in \mathfrak{A}_2$.

Since concatenation of words is associative, so is concatenation of languages, and we may therefore omit the parentheses in products of several languages. The products $\mathfrak{A}\mathfrak{A}, \mathfrak{A}\mathfrak{A}\mathfrak{A}, \ldots$ will be denoted by $\mathfrak{A}^2, \mathfrak{A}^3, \ldots$

* I.e., the direct product of automata with identified inputs.

** *Translator's note*: Often called *complex product*.

Before proceeding to the next operations, we introduce some notation.

Given a word $x = x(1)x(2)\dots x(k)\dots x(l)\dots x(n)$ or an ω-word $x = x(1)\dots x(k)\dots x(l)\dots x(n)\dots$, we shall denote the word $x(k+1)\dots x(l)$ by x_k^l and the ω-word $x(k+1)\dots x(n)\dots$ by x_k^∞.

b) *The concatenation product of a language* $\mathfrak{A}$ *and an* ω*-language* $\mathfrak{B}$ (denoted by $\mathfrak{A}\cdot\mathfrak{B}$) is the ω-language consisting of all ω-words $p = p_0^i p_i^\infty$, where $p_0^i \in \mathfrak{A}$, $p_i^\infty \in \mathfrak{B}$.

c) *The iteration closure*[†] *of a language* $\mathfrak{A}$ (denoted by $\mathfrak{A}^*$) is the language consisting of all words a expressible as

$$a_0^{i_1} a_{i_1}^{i_2} \dots a_{i_{k-1}}^{i_k} \qquad (k = 1, 2, 3, \dots),$$

where all factors are words in $\mathfrak{A}$. In other words, $\mathfrak{A}^*$ is the set-theoretic union of the languages $\mathfrak{A}$, $\mathfrak{A}^2$, $\mathfrak{A}^3, \dots$[††]

d) *The strong iteration closure of a language* (denoted by $\mathfrak{A}^\infty$) is the ω-language consisting of all words p expressible as an infinite product.

$$p = p_0^{i_1} p_{i_1}^{i_2} p_{i_2}^{i_3} \dots,$$

where all factors are words in $\mathfrak{A}$.

THEOREM 1.7. *The class of languages representable by finite sources is closed under the operations of union, reflection, projection, cylindrification, concatenation, iteration closure. The class of* ω*-languages representable in finite macrosources is closed under the operations of union, projection, cylindrification. If the language* $\mathfrak{A}_1$ *and the* ω*-language* $\mathfrak{A}_2$ *are both representable in finite sources, then so is their concatenation product. In all these cases, there is an algorithm which, given the original sources and macrosources, constructs the required source (macrosource).*

Proof. We have already dealt with union and reflection. For cylindrification and projection we apply the procedure of multiple edges and erasing of superfluous labels, respectively, as described above for automata. Note that each of these procedures applied to a source (macrosource) again yields a source (macrosource), whereas erasure of labels in the diagram of an automaton need not give the diagram of an automaton.

For the remaining operations we shall assume that the original sources are two-terminals and the original macrosources have a single input terminal, since this can always be effectively achieved (Theorem 1.6).

[†] *Translator's note*: Often called *star closure.*

[††] In the literature, the iteration closure of a language $\mathfrak{A}$ is often defined as the language $\mathfrak{A}^* \cup \{\wedge\}$.

Concatenation product of languages $\mathfrak{A}_1, \mathfrak{A}_2$. A representing two-terminal $\mathfrak{B}$ may be obtained by connecting a two-terminal $\mathfrak{B}_2$ representing $\mathfrak{A}_2$ in series with a two-terminal $\mathfrak{B}_1$ representing $\mathfrak{A}_1$. This is done as follows. Identify the input terminal of $\mathfrak{B}_2$ and the output terminal of $\mathfrak{B}_1$, forming a single vertex (junction point); the input and output terminals of $\mathfrak{B}$ are defined to be the input terminal of $\mathfrak{B}_1$ and the output terminal of $\mathfrak{B}_2$, respectively. That this construction meets our needs follows from the fact that there are no "false paths": every path from the input terminal of $\mathfrak{B}$ to its output terminal must pass through the "junction" point, and it does so only once. The same holds for the constructions described below.

Concatenation product of a language $\mathfrak{A}_1$ *and an* ω*-language* $\mathfrak{A}_2$. A representing macrosource $\mathfrak{B}$ is obtained by connecting a macrosource $\mathfrak{B}_2$ having one input terminal and representing $\mathfrak{A}_2$ in series with a two-terminal $\mathfrak{B}_1$ representing $\mathfrak{A}_1$, by the same identification procedure as in the previous case. The input terminal of $\mathfrak{B}$ is that of $\mathfrak{B}_1$, the limit macrostates are precisely those of $\mathfrak{B}_2$.

Iteration closure of a language $\mathfrak{A}_1$. If $\mathfrak{B}_1$ is a source representing $\mathfrak{A}_1$, a representing source $\mathfrak{B}$ for the closure of $\mathfrak{A}_1$ may be obtained by identifying the input and output terminals of $\mathfrak{B}_1$ and defining the "junction" to be the only initial vertex and the only final vertex. This procedure of converting $\mathfrak{B}_1$ into $\mathfrak{B}$ may be called *cycling the source* $\mathfrak{B}_1$. If desired, $\mathfrak{B}$ may then be converted into a two-terminal.

Strong iteration closure $\mathfrak{A}_1^{\infty}$. A representing macrosource is again obtained by identifying the input and output terminals of the original source $\mathfrak{B}_1$. The junction is the only initial vertex, and any set of vertices containing the junction is a limit set. Again, if required, one can adjoin an input terminal. This completes the proof.

REMARK. Consider the languages represented by sources $\mathfrak{B}_1, \mathfrak{B}_2$. As mentioned at the beginning of this section, we obtain a source $\mathfrak{B}$ representing the union of these languages by combining $\mathfrak{B}_1$ and $\mathfrak{B}_2$ into a single source. Another useful construction for two-terminals $\mathfrak{B}_1$ and $\mathfrak{B}_2$ is to define a two-terminal $\mathfrak{B}$ from $\mathfrak{B}_1$ and $\mathfrak{B}_2$ by the following procedure, which might be termed parallel connection: The input (output) terminals of $\mathfrak{B}_1$ and $\mathfrak{B}_2$ are identified, forming the input (output) terminal of the source $\mathfrak{B}$. Thus, serial connection of two-terminals corresponds to concatenation of languages, parallel connection to union.

A few examples will illustrate Theorem 1.7. Denote the source of Figure 10*a* by A. The iteration closure of the language $\omega(A, \pi_1, \pi_3)$ is represented

by the source $\langle \widetilde{B}, \pi_3, \pi_3 \rangle$ (Figure 10*b*), whose diagram is derived from that of A by identifying the initial vertex π_1 and the final vertex π_3. If we make π_3 the initial vertex in the diagram of Figure 10*b* and $\{\pi_2, \pi_3\}$ the limit macrostate, we get a macrosource $\langle \widetilde{B}, \pi_3, \{\pi_2, \pi_3\} \rangle$ representing the strong iteration closure of $\omega(A, \pi_1, \pi_3)$. Figure 10*c* illustrates the diagram of a source B obtained from $\widetilde{B}$ by introducing an input terminal π_1. The sources $\langle \widetilde{B}, \pi_3, \{\pi_2, \pi_3\} \rangle$ and $\langle B, \pi_3, \{\pi_2, \pi_3\} \rangle$ are clearly equivalent. Figure 10*d* gives the diagram of a macrosource $\langle \widetilde{C}, q_1, \{q_4, q_5\} \rangle$ representing the concatenation product of the language $\omega(M, q_1, q_3)$ (the diagram of Figure 11*a*) and the ω-language $\Omega(B, \pi_1, \{\pi_2, \pi_3\})$ (the diagram of Figure 10*c*). Identification of the vertices q_5 and q_3 (merging of equivalent vertices) in the macrosource $\langle \widetilde{C}, q_1, \{q_4, q_5\} \rangle$ (Figure 10*d*) converts it into an equivalent macrosource $\langle C, q_1, \{q_3, q_4\} \rangle$ (Figure 10*e*).

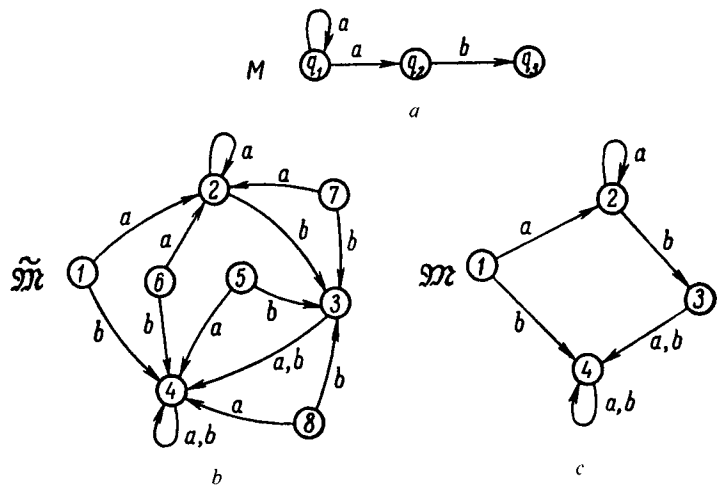


Figure 11

I.7. Determinization of sources. Operations preserving representability of languages in finite automata

Since finite automata are special cases of finite sources, any language representable in a finite automaton is representable in some finite source. The following theorem shows that the converse is also true.

THEOREM 1.8 (DETERMINIZATION). *There exists an algorithm which, given any source (possibly with empty edges)* $\mathfrak{B} = \langle B, Q', Q'' \rangle$ *with* n *states, constructs an equivalent finite automaton with at most* 2^n *states.*

Proof. Call a set π of vertices of a diagram B *closed* if it has the following property: if $q_i \in \pi$ and there exists an empty edge from q_i to q_j, then $q_j \in \pi$. Of course, for ordinary sources (in which all edges are labeled) any set of vertices is closed. The *closure* of a set π of vertices is the smallest closed set including π. It follows from this definition that the language $\omega(B, Q', Q'')$ remains unchanged if Q' and Q'' are replaced by their closures; we may therefore assume once and for all that Q' and Q'' are closed. Fix some word $i = i(1)i(2)\dots i(t)$ and a set π of vertices; consider the set of all paths in B which begin in π and carry the word i. The ends of these paths form a closed set of vertices π'. We shall say that the flow from π in the direction of the word i leads to π', or, briefly, that i takes π to π'; in symbols,

$$\pi i = \pi'.$$

Obviously, if $i = i_1 i_2$, then

$$\pi i = (\pi i_1) i_2.$$

It therefore suffices to consider the products of sets of vertices (macrostates) by one-letter words—elements of the alphabet X. In this fashion, we associate with the diagram B an automaton 2^B defined as follows:

(I) the states of 2^B are closed macrostates (sets of vertices) of the diagram B;

(II) the next-state function Ψ is defined by

$$\Psi(\pi, x) = \pi x.$$

Thus, if B contains n vertices, the automaton 2^B contains at most 2^n states.

We now claim that the language $\omega(B, Q', Q'')$ is representable in the automaton 2^B, anchored as follows: initial state, Q'; the set $\mathfrak{C}$ of final states consists of all macrostates of B which are not disjoint from Q''. Denote this anchored automaton by $\langle 2^B, Q', \mathfrak{C} \rangle$.

(I) Let $i \in \omega(B, Q', Q'')$; then the flow from Q' in the direction of i contains a path ending at some vertex $q_s \in Q''$. Thus i takes Q' to a macrostate whose intersection with Q'' contains an element q_s (and possibly other elements as well). Consequently, $i \in \omega(2^B, Q', \mathfrak{C})$.

(II) Suppose that $Q'i = \pi$ and $q_s \in Q'' \cap \pi$ for suitable π and q_s; then one of the paths from Q' to q_s carries the word i. Thus $i \in \omega(B, Q', q_s) \subseteq \subseteq \omega(B, Q', Q'')$. Q.E.D.

EXAMPLE 1. Let us apply the theorem to the source $\langle M, q_1, q_3 \rangle$ whose diagram is illustrated in Figure 11*a*. The states of the automaton $\mathfrak{M}$ equivalent to this source are all eight subsets of $\{q_1, q_2, q_3\}$: $\{q_1\}$, $\{q_2\}$, $\{q_3\}$,

TABLE 7a

x	$\{q_1\}$	$\{q_2\}$	$\{q_3\}$	$\{q_1, q_2\}$	$\{q_1, q_3\}$	$\{q_2, q_3\}$	$\{q_1, q_2, q_3\}$	$\varnothing$
a	$\{q_1, q_2\}$	$\varnothing$	$\varnothing$	$\{q_1, q_2\}$	$\{q_1, q_2\}$	$\varnothing$	$\{q_1, q_2\}$	$\varnothing$
b	$\varnothing$	$\{q_3\}$	$\varnothing$	$\{q_3\}$	$\varnothing$	$\{q_3\}$	$\{q_3\}$	$\varnothing$

TABLE 7b

x	$\{\pi_3\}$	$\{\pi_2\}$	$\varnothing$
a	$\{\pi_2\}$	$\varnothing$	$\varnothing$
b	$\varnothing$	$\{\pi_3\}$	$\varnothing$

TABLE 7c

x	v_1	v_2	v_3	v_4	v_5	v_6	v_7
a	v_2	v_2	v_3	v_5	v_6	v_6	v_5
b	v_3	v_4	v_3	v_6	v_4	v_6	v_6

$\varnothing, \{q_1, q_2\}, \{q_1, q_3\}, \{q_2, q_3\}, \{q_1, q_2, q_3\}$. The initial state of $\tilde{\mathfrak{M}}$ is $\{q_1\}$, the final states $\{q_3\}$, $\{q_1, q_3\}$, $\{q_2, q_3\}$, $\{q_1, q_2, q_3\}$. Table 7a is the transition matrix of $\tilde{\mathfrak{M}}$. For example, the table shows that $\{q_1, q_2\}a = \{q_1, q_2\}$, since any path from the macrostate $\{q_1, q_2\}$ in the source M, along edges labeled a, must end in $\{q_1, q_2\}$; $\{q_3\}b = \varnothing$, since there are no edges issuing from q_3. Figure 11b illustrates the diagram of the automaton $\tilde{\mathfrak{M}}$, in which we use the abbreviated notation $\{q_1\} = 1$; $\{q_2\} = 5$; $\{q_3\} = 3$; $\{q_1, q_2\} = 2$; $\{q_1, q_3\} = 6$; $\{q_2, q_3\} = 8$; $\{q_1, q_2, q_3\} = 7$, $\varnothing = 4$. It is not hard to see that in this automaton $\langle \tilde{\mathfrak{M}}, 1, \{3, 6, 7, 8\} \rangle$ the states 5, 6, 7, 8 are inaccessible from state 1. Thus, by eliminating these states we get an equivalent automaton $\langle \mathfrak{M}, 1, 3 \rangle$ (Figure 11c), $\omega(\mathfrak{M}, 1, 3) = \omega(\tilde{\mathfrak{M}}, 1, \{3, 6, 7, 8\})$. It follows that the automaton $\langle \mathfrak{M}, 1, 3 \rangle$ is equivalent to the source $\langle M, 1, 3 \rangle$.

REMARK. In constructing the automaton we need not have written out all its states, since many of them prove to be inaccessible from the initial state and may therefore be eliminated (see Section I.5). A more convenient procedure is to start from the initial state, filling in the transition table [column by column] with new accessible states as they arise, until the procedure breaks off naturally. The idea will be clear from the next example.

EXAMPLE 2. Consider the source $\langle \widetilde{B}, \pi_3, \pi_3 \rangle$ of Figure 10*b*. The initial state of an automaton $\mathfrak{B}$ equivalent to $\langle \widetilde{B}, \pi_3, \pi_3 \rangle$ will be $\{\pi_3\}$. The transition matrix will be:

$$\{\pi_3\} \cdot a = \{\pi_2\}, \qquad \{\pi_2\} \cdot a = \varnothing, \qquad \varnothing \cdot a = \varnothing,$$

$$\{\pi_3\} \cdot b = \varnothing, \qquad \{\pi_2\} \cdot b = \{\pi_3\}, \qquad \varnothing \cdot b = \varnothing.$$

Thus, in filling the column for $\{\pi_3\}$ we find the new states $\{\pi_2\}$ and $\varnothing$. The columns for $\{\pi_2\}$ and $\varnothing$ reveal no new states. The procedure thus breaks off and we get Table 7b.

The resulting automaton $\mathfrak{B}$ has only three states, the final state being $\{\pi_3\}$. Figure 12*a* illustrates the diagram of the automaton $\mathfrak{B}$, with the states $\{\pi_3\}$, $\{\pi_2\}$, $\varnothing$ denoted respectively by p_1, p_2, p_3. Thus, $\omega(\widetilde{B}, \pi_3, \pi_3) = \omega(\mathfrak{B}, p_1, p_1)$. The source $\langle \widetilde{B}, \pi_3, \pi_3 \rangle$, and so also the automaton $\langle \mathfrak{B}, p_1, p_1 \rangle$, represent the iteration closure of the language consisting of the single word *ab*.

The determinization theorem and procedure are of immense significance. Recall that the original motivation for the concept of a source was the projection operation. We now see that the class of finite-state languages is indeed closed under projection. In addition, we are now in a position to prove that this class is closed under all language operations considered hitherto, and also under many other operations. One of these is the operation of *a-annihilation* of a language $\mathfrak{A}$, where a is an element of the alphabet X: this operation eliminates all words of $\mathfrak{A}$ in which the letter a appears.

THEOREM 1.9. *The class of finite-state languages is closed under the following operations*:

a) *union, intersection, subtraction, cylindrification, projection*;

b) *concatenation, iteration closure*;

c) *reflection, a-annihilation.*

There is an algorithm which, given automata representing the original language(s), constructs an automaton representing the resultant language.

Proof. Union, intersection, subtraction and cylindrification were treated

in Theorem 1.1. Projection, concatenation, iteration closure and reflection have been dealt with for sources; the required automaton is constructed by determinization of the corresponding source. As to a-annihilation, the assertion follows from the fact that, removing all labels a appearing in the diagram of the original automaton, we get a source (with empty edges!) representing the required language. Determinization of this source completes the proof.

If the language $\mathfrak{A}$ is representable in an automaton with n states, reflection, projection and iteration closure all yield languages representable in sources with n states, therefore in automata with at most 2^n states. Can this upper bound be improved? The answer is in the negative: for each of these operations one can construct languages whose representing automaton has exactly 2^n states. This is because the upper bound for the number of states of the automaton $\mathfrak{M}$ indicated in the determinization theorem cannot be improved.

THEOREM 1.10. *For a fixed alphabet $X = \{a, b, c\}$ and any n there exists a source $\mathfrak{B}_n$ (over the alphabet X) with n states such that the language $\mathfrak{A}$ represented by $\mathfrak{B}_n$ has left-interchangeability index 2^n (so that $\mathfrak{A}$ cannot be represented in any automaton with less than 2^n states).*

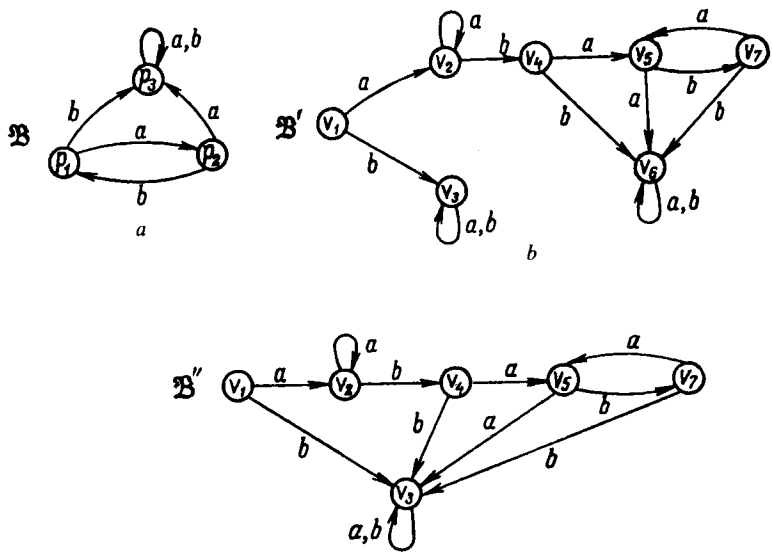


Figure 12

Proof. The diagram B_n of the source $\mathfrak{B}_n$ is defined by Table 6b; the particular case $n = 4$ is illustrated in Figure 9*a*. The only difference between B_n and the diagram of the automaton $\mathfrak{M}_n$ (Theorem 1.3 and Table 6a) is that no edges labeled c issue from the vertex 1, while two edges labeled c issue from the vertex 2 (one of them to 1, the other to 2). Thus the source $\mathfrak{B}_n$, like the automaton $\mathfrak{M}_n$, satisfies the following condition:

(I) For any two macrostates Q', Q'' containing the same number of elements there exists a word x over the subalphabet $\{a, b\}$ such that $Q'x = Q''$.

Let Q'' be any nonempty macrostate containing 2. Then $Q''c = Q'' \cup \{1\}$. But if Q'' contains 1 and not 2, then $Q''_c = Q''\backslash\{1\}$. Together with (I), these observations imply the proposition:

(II) Any nonempty macrostate of the diagram B_n is taken to any other macrostate by some word over the alphabet $\{a, b, c\}$.

Now consider the language $\mathfrak{A} = \omega(B_n, \{1\}, \{1\})$; we claim that its left-interchangeability index is at least (therefore, *exactly*) 2^n. With each of the 2^n macrostates Q' we associate a word taking $\{1\}$ to Q'; this is possible by (II). We shall now prove that no two of these 2^n words are interchangeable. Let p and r take $\{1\}$ to Q' and Q'', respectively. Assume that Q'' contains some vertex (say the vertex j) not contained in Q'. Let x denote the word $\underbrace{aa\ldots a}_{n+1-j}$. It is easy to see that the word rx takes 1 to 1, and is therefore in $\mathfrak{A}$; but the word px takes 1 to a state different from 1, and so is not in $\mathfrak{A}$. This proves the theorem.

Given any source with n states, one can use the determinization procedure described in the proof of Theorem 1.8 to construct an equivalent automaton with 2^n states; Theorem 1.10 shows that in certain cases this automaton cannot be simplified. However, in many cases (as in our examples above) the resultant automaton may be considerably simplified by the simple expedients of elimination of inaccessible states, and merging of equivalent and absorbing states. Our next remark concerns the number of final states required for representing languages in finite automata and finite sources. We have shown that the analysis may be restricted to sources with one final state, without thereby reducing the class of representable languages. However, the following simple example shows that, for any natural number k, there exists a language which is representable in a finite automaton with k final states but is not representable in any (even infinite!) automaton with less than k final states. Consider the language consisting of k words $\{a, aa, aaa, \ldots, \underbrace{aa\ldots a}_{k}\}$ and any finite automaton representing it. All states on the path from the initial state carrying the word $\underbrace{aa\ldots a}_{k}$, except

the initial state, are final. Thus, were the number of final states less than k, the path would necessarily pass twice through some state, forming a loop. But then the language would contain words of arbitrary length over the alphabet $\{a\}$ (by repetition of the loop). Thus the number of final states is at least k. Construction of a finite automaton with k final states representing this language is trivial.

Thus, while the number of final states (like the number of initial states) is immaterial for the representation of languages in sources, it is significant, and not a priori bounded, for representation of the same languages in finite automata.

I.8. Determinization of macrosources. Operations preserving representability of ω-languages in finite automata

Like sources, macrosources were introduced in order to deal with the operation of projection (of an ω-language representable in a finite automaton). We shall use the term *determinization of* $\mathfrak{B}$ for construction of a finite automaton equivalent to a given macrosource $\mathfrak{B}$. If we could prove that any finite macrosource can be determinized, this would imply that the class of finite-state ω-languages is closed under projection. Moreover, with effective determinization procedures at our disposal, as in the case of finite automata, we would have algorithms for the construction of automata representing the required projections, concatenation products and strong iteration closures. In other words, the analogue of Theorem 1.8 (together with Theorem 1.1) would imply the analogue of Theorem 1.9, as in the preceding section. Though these assertions are in point of fact all true, both the sequence and method of proof are quite different from those obtaining in the case of sources. We shall first prove closure under concatenation and strong iteration closure (and this is by no means trivial), and only then shall we be able to prove the analogue of the determinization theorem and its corollary on projections. We thus have the following theorems.

THEOREM 1.11 (CONCATENATION). *For any finite-state language* $\mathfrak{A}$ *and finite-state* ω*-language* $\mathfrak{B}$ *over the same alphabet* X*, the concatenation product* $\mathfrak{A}\mathfrak{B}$ *is a finite-state* ω*-language. There is an algorithm which given automata representing* $\mathfrak{A}$ *and* $\mathfrak{B}$*, constructs an automaton representing* $\mathfrak{A}\mathfrak{B}$.

THEOREM 1.12 (STRONG ITERATION CLOSURE). *For any finite-state language* $\mathfrak{A}$*, the strong iteration closure* $\mathfrak{A}^{\infty}$ *is a finite-state* ω*-language. There is*

an algorithm which, given a finite automaton representing $\mathfrak{A}$, *constructs a finite automaton representing* $\mathfrak{A}^\infty$.

The proofs of these theorems will be postponed to the following sections. We shall use them immediately to prove the following

THEOREM 1.13 (DETERMINIZATION OF MACROSOURCES). *There is an algorithm which, given any finite macrosource, constructs an equivalent finite automaton.*

Proof. Consider the macrosource $\langle B, Q_0, \mathfrak{C}\rangle$. Since $\Omega(B, Q_0, \mathfrak{C})$ is the union of all possible $\Omega(B, q_0, \Gamma)$, where q_0 and Γ range over Q_0 and $\mathfrak{C}$, respectively, it will suffice to prove the theorem for singletons Q_0 and $\mathfrak{C}$. Let $\mathfrak{R} = \Omega(B, q_0, \Gamma)$; to fix ideas, suppose that $\Gamma = \{q_1, q_2, \ldots, q_s\}$. Let B' denote the subdiagram of B containing all vertices of Γ and all edges incident on these vertices. If $\mathfrak{R}' = \Omega(B', q_1, \Gamma)$ is empty (and this is effectively decidable), the original ω-language $\mathfrak{R}$ is also empty. Otherwise, we shall use the easily verified identity.

$$\mathfrak{R}' = \Omega(B', q_1, \Gamma) = \{\omega(B', q_1, q_2)\cdot\omega(B', q_2, q_3)\ldots \ldots\omega(B', q_{s-1}, q_s)\cdot\omega(B', q_s, q_1)\}^\infty,$$

which shows how an automaton representing the ω-language $\mathfrak{R}'$ may be obtained from automata representing the factor-languages. Finally, note that $\mathfrak{R} = \omega(B, q_0, q_1)\,\mathfrak{R}'$ if the first factor is nonempty, and $\mathfrak{R} = \mathfrak{R}'$ if the first factor is empty but $q_0 = q_1$. Thus, an automaton representing $\mathfrak{R}$ is effectively constructible from automata representing the language $\omega(B, q_0, q_1)$ and the ω-language $\mathfrak{R}'$. This proves the theorem.

COROLLARY. *There is an algorithm which, given any macroanchored automaton* $\langle \mathfrak{M}, Q_0, \mathfrak{C}\rangle$, *constructs a finite automaton representing the projection of the* ω*-language* $\Omega(\mathfrak{M}, Q_0, \mathfrak{C})$.

I.9. Proof of the Concatenation Theorem (Theorem 1.11)

Let $\mathfrak{A} = \omega(\mathfrak{M}, q_0, Q')$, $\mathfrak{B} = \Omega(\mathfrak{N}, \pi_1, \mathfrak{C})$; then $\mathfrak{A}\cdot\mathfrak{B}$ is the union of all possible concatenation products of the form

$$\omega(\mathfrak{M}, q_0, q')\cdot\Omega(\mathfrak{N}, \pi_1, \Pi'),$$

where q' is an arbitrary state in Q' and Π' an arbitrary macrostate in $\mathfrak{C}$. Since the class of finite-state ω-languages is closed under union, it suf-

fices to prove the theorem for the case $\mathfrak{A} = \omega(\mathfrak{M}, q_0, q')$ and $\mathfrak{B} = \Omega(\mathfrak{N}, \pi_1, \Pi')$. We first describe the basic idea underlying the construction. Suppose that we have one copy of the automaton $\mathfrak{M}$ and an unlimited stock $\{\mathfrak{N}_1, \mathfrak{N}_2, \mathfrak{N}_3, \ldots\}$ of copies of $\mathfrak{N}$. Consider the following imaginary experiment, aimed at determining whether an ω-word $x = x(1)x(2)\ldots x(t)\ldots$ belongs to the ω-language $\mathfrak{A} \cdot \mathfrak{B}$:

1) Start $\mathfrak{M}$, i.e., apply $x(1)x(2)\ldots$ to the initialized automaton $\langle \mathfrak{M}, q_0 \rangle$.

2) If the automaton $\mathfrak{M}$ first passes into state q' at some time τ_1, start the first copy $\mathfrak{N}_1$ from the reserve $\{\mathfrak{N}_i\}$ at this instant, i.e., apply the "tail" $x(\tau_1)x(\tau_1 + 1)\ldots$ to the initialized automaton $\langle \mathfrak{N}_1, \pi_1 \rangle$.

3) If $\mathfrak{M}$ passes into state q' for the second time at time τ_2, start $\mathfrak{N}_2$ at this instant.

Similarly, a new copy of $\mathfrak{N}$ is started whenever $\mathfrak{M}$ goes into state q'. Let $v_0(t)$ denote the state of $\mathfrak{M}$ at time t, $v_\mu(t)$ the state of the μ-th copy of $\mathfrak{N}$ at time t if $\mathfrak{N}_\mu$ has already been started, $v_\mu(t) = \wedge$ (empty symbol) otherwise. It is obvious that $x \in \mathfrak{A}\mathfrak{B}$ if and only if at least one of the copies in the stock has been started and at least one of the functioning copies produces the limit macrostate Π'. Thus $x \in \mathfrak{A}\mathfrak{B} \equiv \exists \mu\, [\lim v_\mu(t) = \Pi']$. Now note that if $v_\mu(\tau) = v_\nu(\tau) \neq \wedge$ for some τ, then $v_\mu(\sigma) = v_\nu(\sigma) \neq \wedge$ for all $\sigma > \tau$. Consequently, examination of the copy $\mathfrak{N}_\nu$ and its state $v_\nu(t)$ yields the same information concerning the question "$x \in \mathfrak{A}\mathfrak{B}$?" as the examination of $\mathfrak{N}_\mu$. In other words, once the above equality has been detected at some instant, we may return the copy $\mathfrak{N}_\nu$ to the stock. Hence, since the automaton $\mathfrak{N}$ has only finitely many states, say $\pi_1, \pi_2, \ldots, \pi_r$, we may modify the original experiment and confine ourselves to a finite stock containing $r + 1$ copies $\mathfrak{N}_1, \mathfrak{N}_2, \ldots, \mathfrak{N}_{r+1}$. It is assumed that any copy returned to the stock may be used again provided certain precautionary measures are taken. The final product of a rigorous implementation of this idea is an automaton $\mathfrak{H}$ representing the ω-language $\mathfrak{A}\mathfrak{B}$.

Description of the automaton $\mathfrak{H}$. *Interpretation.* The states of $\mathfrak{H}$ are vectors $\mathbf{v} = \langle v_0, v_1, \ldots, v_{r+1} \rangle$, where v_0 is a state of $\mathfrak{M}$, $v_1, v_2, \ldots, v_{r+1}$ are either states of $\mathfrak{N}$ or $\wedge$ such that the components $v_1, v_2, \ldots, v_{r+1}$ different from $\wedge$ (we call these the *π-components*) are pairwise distinct. The vector $\mathbf{v}(t) = \langle (v_0(t), v_1(t), \ldots, v_{r+1}(t) \rangle$ "describes" the situation at time t: it indicates the state of $\mathfrak{M}$, the copies of $\mathfrak{N}$ still in the stock (components $\wedge$ in $\mathbf{v}(t)$ and the states of those copies active at this time (i.e., those receiving symbols). We repeat that the active copies are all in different states; thus there cannot be more than r of them and the stock always contains at least one copy.

The initial state of the automaton $\mathfrak{H}$ is $\langle q_0, \wedge, \wedge, \ldots, \wedge \rangle = v_0$. It remains to define the product $\langle v_0, v_1, \ldots, v_i, \ldots, v_{r+1} \rangle \cdot a$ for any letter $a \in X$. To this end, multiply each of the nonempty components by a and, if $v_0 a = q'$, replace one of the empty components (say the leftmost) by π_1. The resulting vector $\langle v'_0, v'_1, \ldots, v'_{r+1} \rangle$ is not always a state of $\mathfrak{H}$, since it may contain equal nonempty components. The meaning of this vector is that each of the active copies has functioned and, if necessary, another copy from the stock has been started. The actual *state* $\langle v_0, \ldots, v_{r+1} \rangle \cdot a$ is obtained by "clearing" the vector $\langle v'_0, \ldots, v'_{r+1} \rangle$: replace any component on whose left there is an equal component by the empty symbol (interpretation: return all "superfluous" copies to the stock). This completes the definition of $\mathfrak{H}$.

Now assume that the automaton $\langle \mathfrak{H}, \mathbf{v}_0 \rangle$, receiving the input ω-word $x = x(1)x(2)\ldots$, passes through states

$$\mathbf{v}_0 = \mathbf{v}(1), \mathbf{v}(2), \ldots, \mathbf{v}(t),$$

where

$$\mathbf{v}(t) = \langle v_0(t), v_1(t), \ldots, v_\mu(t), \ldots, v_{r+1}(t) \rangle.$$

We shall show (see **A** and **B** below) that, as before, $x \in \mathfrak{A} \cdot \mathfrak{B} \equiv \exists \mu \, [\lim v_\mu(t) = \Pi']$. This will show that $\mathfrak{H}$ represents the ω-language $\mathfrak{A} \cdot \mathfrak{B}$ when limit macrostates $\{\Gamma_i\}$ are defined by the following condition: for some μ, $1 \leqq \mu \leqq r+1$, the set of μ-components in the vectors of Γ_i coincides with the set Π'. This will complete the proof of the theorem.

A. *If* $\lim v_\mu(t) = \Pi'$, *then* $x \in \mathfrak{A} \cdot \mathfrak{B}$.

Let $\lim v_\mu(t) = \Pi'$. Then from some time on, $v_\mu(t) \neq \wedge$. If τ is the very last instant at which $v_\mu(\tau) = \wedge$, then the component $v_\mu(\tau + 1)$ of $\mathbf{v}(\tau + 1)$ is π_1; this in turn could happen only if $v_0(\tau + 1) = q'$. It follows, first, that $x(1)\ldots x(\tau)$ takes the automaton $\mathfrak{M}$ from q_0 to q', i.e., $x_0^\tau \in \mathfrak{A}$. Now note that $v_\mu(\tau + 1)$, $v_\mu(\tau + 2)$, $v_\mu(\tau + 3), \ldots$ are precisely the states through which the copy $\mathfrak{N}_\mu$ passes (from time $r + 1$ on, the copy $\mathfrak{N}_\mu$ is never returned to the stock). Hence, since

$$\lim_{\lambda \to \infty} v_\mu(\tau + \lambda) = \lim_{t \to \infty} v_\mu(t) = \Pi',$$

the "tail" x_τ^∞ applied to $\mathfrak{N}_\mu$ satisfies the condition $x_\tau^\infty \in \mathfrak{B}$. Thus $x_0^\tau \in \mathfrak{A}$ & $x_\tau^\infty \in \mathfrak{B}$, that is, $x \in \mathfrak{A}\mathfrak{B}$.*

* Recall that x_μ^∞ denotes the ω-word $x(\mu + 1)x(\mu + 2)\ldots$, and x_0^μ the word $x(1)x(2)\ldots x(\mu)$.

B. *If* $x \in \mathfrak{AB}$, *then* $\lim v_\mu(t) = \Pi'$ *for some* μ.

Let $x \in \mathfrak{AB}$, i.e., $x_0^\tau \in \mathfrak{A}$, $x_\tau^\infty \in \mathfrak{B}$ for some τ. It follows from $x_0^\tau \in \mathfrak{A}$ that the vector $\mathbf{v}(\tau + 1)$ and a suitable index s satisfy the equalities $v_0(\tau + 1) = q'$ and $v_s(\tau + 1) = \pi_1$. Application of the "tail" $x(\tau + 1)x(\tau + 2)\ldots$ to the automaton $\langle \mathfrak{N}, \pi_1 \rangle$ takes it through a certain sequence of states:

$$\pi(1) = \pi_1, \pi(2), \pi(3), \ldots, \pi(\lambda), \ldots$$

It follows from $x_\tau^\infty \in \mathfrak{B}$ that $\lim\limits_{\lambda \to \infty} \pi(\lambda) = \Pi'$. It will therefore suffice to find a μ such that some "tail" of the sequence

$$v_\mu(1), v_\mu(2), \ldots, v_\mu(t), \ldots$$

coincides with a suitable tail of the sequence $\pi(1), \pi(2), \pi(3), \ldots$ By assumption, $\pi(1) = v_s(\tau + 1)$; therefore $\pi(2)$ is some $v_{s'}(\tau + 2)$, where either $s' = s$ (if the s-th component of the vector $\mathbf{v}(\tau + 2)$ was not cleared) or $s' < s$ (if the s-th component was cleared). Similarly, $\pi(3)$ is some $v_{s''}(\tau + 3)$, where $s'' \leqq s'$. The numbers $s, s', s'', \ldots$ cannot decrease indefinitely, and hence they "level off" at some instant ρ and assume a constant value which we denote by μ. This means that the "tail"

$$v_\mu(\tau + \rho), v_\mu(\tau + \rho + 1), \ldots$$

coincides with the "tail"

$$\pi(\rho), \pi(\rho + 1), \ldots$$

This completes the proof.

EXAMPLE. Given automata $\langle \mathfrak{M}, 1, 3 \rangle$ (Figure 11*c*) and $\langle \mathfrak{B}, p_1, \{p_1, p_2\} \rangle$ (Figure 12*a*), we shall construct an automaton $\mathfrak{C}$ representing the concatenation product of the language $\omega(\mathfrak{M}, 1, 3)$ and the ω-language $\Omega(\mathfrak{B}, p_1, \{p_1, p_2\})$.

According to the algorithm described in the proof of the Concatenation Theorem, the initial state is the vector $\mathbf{v}_1 = \langle 1, \wedge, \wedge, \wedge, \wedge \rangle$. We shall not write down all states of the automaton $\mathfrak{C}$ (as prescribed by the algorithm), since many of them will turn out to be inaccessible from the initial state $\mathbf{v}_1$. Following the remark in Section I.7, we shall introduce new states, accessible from $\mathbf{v}_1$, as they are entered in the transition matrix for $\mathfrak{C}$. To define the product $\mathbf{v}_1 \cdot a$, we must multiply the vector $\langle 1, \wedge, \wedge, \wedge, \wedge \rangle$ componentwise by a. But $1 \cdot a = 2$ (see the diagram of $\mathfrak{M}$, Figure 11*c*), and so $\mathbf{v}_1 \cdot a = \langle 2, \wedge, \wedge, \wedge, \wedge \rangle$. Denote this new vector by $\mathbf{v}_2$, i.e., $\mathbf{v}_2 = \langle 2, \wedge, \wedge, \wedge, \wedge \rangle$.

Thus,

$\mathbf{v}_1 \cdot a = \mathbf{v}_2$.

Similarly:

$\mathbf{v}_1 \cdot b = \langle 4, \wedge, \wedge, \wedge, \wedge \rangle, \mathbf{v}_3 = \langle 4, \wedge, \wedge, \wedge, \wedge \rangle$, i.e., $\mathbf{v}_1 \cdot b = \mathbf{v}_3$,

$\mathbf{v}_2 \cdot a = \langle 2, \wedge, \wedge, \wedge, \wedge \rangle$, $\mathbf{v}_2 \cdot a = \mathbf{v}_2$,

$\mathbf{v}_2 \cdot b = \langle 3, p_1, \wedge, \wedge, \wedge \rangle$

(since 3 is the final state of $\mathfrak{M}$, the first symbol $\wedge$ is replaced by p_1);

$\mathbf{v}_4 = \langle 3, p_1, \wedge, \wedge, \wedge \rangle$, $\mathbf{v}_2 \cdot b = \mathbf{v}_4$;

$\mathbf{v}_3 \cdot a = \langle 4, \wedge, \wedge, \wedge, \wedge \rangle$, $\mathbf{v}_3 \cdot a = \mathbf{v}_3$;

$\mathbf{v}_3 \cdot b = \langle 4, \wedge, \wedge, \wedge, \wedge \rangle$, $\mathbf{v}_3 \cdot b = \mathbf{v}_3$;

$\mathbf{v}_4 \cdot a = \langle 4, p_2, \wedge, \wedge, \wedge \rangle$, since $p_1 \cdot a = p_2$ (see Figure 12*a*);

$\mathbf{v}_5 = \langle 4, p_2, \wedge, \wedge, \wedge \rangle$, $\mathbf{v}_4 \cdot a = \mathbf{v}_5$;

$\mathbf{v}_4 \cdot b = \langle 4, p_3, \wedge, \wedge, \wedge \rangle$, $\mathbf{v}_6 = \langle 4, p_3, \wedge, \wedge, \wedge \rangle$, $\mathbf{v}_4 \cdot b = \mathbf{v}_6$;

$\mathbf{v}_5 \cdot a = \langle 4, p_3, \wedge, \wedge, \wedge \rangle = \mathbf{v}_6$, $\mathbf{v}_5 \cdot a = \mathbf{v}_6$;

$\mathbf{v}_5 \cdot b = \langle 4, p_1, \wedge, \wedge, \wedge \rangle$, $\mathbf{v}_7 = \langle 4, p_1, \wedge, \wedge, \wedge, \rangle$, $\mathbf{v}_5 \cdot b = \mathbf{v}_7$;

$\mathbf{v}_6 \cdot a = \langle 4, p_3, \wedge, \wedge, \wedge \rangle = \mathbf{v}_6$, $\mathbf{v}_6 \cdot a = \mathbf{v}_6$;

$\mathbf{v}_6 \cdot b = \langle 4, p_3, \wedge, \wedge, \wedge \rangle = \mathbf{v}_6$, $\mathbf{v}_6 \cdot b = \mathbf{v}_6$;

$\mathbf{v}_7 \cdot a = \langle 4, p_2, \wedge, \wedge, \wedge \rangle = \mathbf{v}_5$, $\mathbf{v}_7 \cdot a = \mathbf{v}_5$;

$\mathbf{v}_7 \cdot b = \langle 4, p_3, \wedge, \wedge, \wedge \rangle = \mathbf{v}_6$. $\mathbf{v}_7 \cdot b = \mathbf{v}_6$.

Thus, only seven states—those indicated in the transition matrix (Table 7c)—are accessible from $\mathbf{v}_1$. It remains to define the terminal macrostates. To this end, note that the states $p_1 p_2$ entering the limit macrostate of the original automaton $\mathfrak{B}$ (Figure 12*a*) appear only as second components, and only in the vectors $\mathbf{v}_4, \mathbf{v}_5, \mathbf{v}_7$. Consequently, the limit macrostates of the resultant automaton must be $\{\mathbf{v}_4, \mathbf{v}_5\}$, $\{\mathbf{v}_5, \mathbf{v}_7\}$, $\{\mathbf{v}_4, \mathbf{v}_5, \mathbf{v}_7\}$. Figure 12*b* illustrates the diagram $\mathfrak{B}'$ of this automaton. The macrostates $\{\mathbf{v}_4, \mathbf{v}_5\}$ and $\{\mathbf{v}_4, \mathbf{v}_5, \mathbf{v}_7\}$ are fictitious and may therefore be eliminated. Merging of the absorbing states $\mathbf{v}_3, \mathbf{v}_6$ converts the automaton $\langle \mathfrak{B}', \mathbf{v}_1, \{\mathbf{v}_5, \mathbf{v}_7\} \rangle$ into an equivalent automaton $\langle \mathfrak{B}'', \mathbf{v}_1, \{\mathbf{v}_5, \mathbf{v}_7\} \rangle$ (Figure 12*c*).

I.10. Proof of the Strong Iteration Theorem (Theorem 1.12)

We first introduce some notation and prove a lemma, before proceeding with the actual proof of the theorem.

Consider an ω-word $x = x(1)x(2)\ldots$ and an initialized automaton $\langle \mathfrak{M}, q_0 \rangle$. Suppose that a "tail" x_τ^∞ of this ω-word is applied to a copy of the automaton, and somewhat later, at time τ'', another copy of the same initialized automaton begins to receive the tail $x_{\tau''}^\infty$. It may happen that

after some time interval v both copies reach the same state (and therefore remain henceforth in this state). We shall then say that the tails x_τ^∞ and $x_{\tau'}^\infty$ of the ω-word x are *merged* by the automaton $\langle \mathfrak{M}, q_0 \rangle$. It is clear that "mergeability" is an equivalence relation which partitions the set of all tails (including the ω-word x_0^∞ as a "tail" of itself) into mergeability classes. Actually, mergeability classes were already encountered in the proof of the Concatenation Theorem, though the term was not used explicitly there. The essential fact is that the number of mergeability classes is finite: it can never exceed the number r of states of the automaton $\langle \mathfrak{M}, q_0 \rangle$. This idea will be used below to prove the useful Stability Lemma. We first introduce some terminology and notation. Given an anchored finite automaton $\langle \mathfrak{M}, q_0, Q' \rangle$, we shall call an ω-word x *stable* with respect to this automaton if there exists an infinite set of tails $x_{\mu_1}^\infty, x_{\mu_2}^\infty, \ldots$ such that

(I) all the words $x_0^{\mu_i}$ belong to $\omega(\mathfrak{M}, q_0, Q')$;

(II) the automaton $\langle \mathfrak{M}, q_0 \rangle$ merges each of the "tails" $x_{\mu_i}^\infty$ with the original ω-word x.

The stability of an ω-word can be determined by the following imaginary experiment. Suppose that an unlimited stock of copies of the automaton $\langle \mathfrak{M}, q_0, Q' \rangle$ is available. Given an ω-word ξ, start the first copy and, the instant it reaches one of the states in Q', start the second copy. Then, when the first copy again reaches a state in Q', start yet another (third) copy, and so on. The ω-word is stable if and only if there is an infinite set of activated copies which, at suitable times, reach a state coinciding with that of the first, "basic" copy. By suitable interpretation of this experiment, analogous to the construction in the Concatenation Theorem, we can prove the folowing proposition:

STABILITY LEMMA. *There is an algorithm which, given any finite automaton* $\langle \mathfrak{M}, q_0, Q' \rangle$, *constructs a macroanchored finite automaton* $\langle \mathfrak{N}, \pi_1, \mathfrak{C} \rangle$ *representing the set of all* ω*-words which are stable with respect to* $\langle \mathfrak{M}, q_0 \rangle$.

Proof. Let $\mathfrak{M}$ have r states $q_0, q_1, \ldots, q_{r-1}$. The states of $\mathfrak{N}$ are defined to be $(r+2)$-vectors $\mathbf{v} = \langle v_1, v_2, \ldots, v_{r+2} \rangle$, where v_1 is a state of $\mathfrak{M}$, v_{r+2} is $\wedge$ (empty state) or $*$ (star), and the remaining components $v_2, \ldots, v_{r+1}$ may be either states of $\mathfrak{M}$ or empty states, provided the components $v_1, v_2, \ldots, v_{r+1}$ that are states of $\mathfrak{M}$ (we call these q-coordinates) are pairwise distinct. Thus $\mathbf{v}$ always contains at least one empty component. The initial state is the vector $\langle q_0, \wedge, \ldots, \wedge, * \rangle$. To define the product of a vector $\mathbf{v}$ and an input letter a, multiply each component of $\mathbf{v}$ by the letter a, setting $\wedge \cdot a = \wedge$ and $* \cdot a = \wedge$; if $v_1 \cdot a \in Q'$, one of the empty components (say

the leftmost) is replaced by the letter q_0.† The resulting vector $\langle v'_1, v'_2, \ldots, v'_{r+2} \rangle$ is then cleared in the following way:

(I) If at least one of the components $v'_2, \ldots, v'_{r+1}$ coincides with v'_1, then $\mathbf{v} \cdot a$ is defined to be the vector $\langle v'_1, \wedge, \wedge, \ldots, \wedge, * \rangle$.

(II) Otherwise, $\wedge$ replaces every q-component on whose left an equal component appears.

Assume that the automaton $\langle \mathfrak{N}, \mathbf{v}_1 \rangle$, having received an ω-word x, passes through the states

$$\mathbf{v}_1 = \mathbf{v}(1), \mathbf{v}(2), \ldots, \mathbf{v}(t), \ldots$$

Call the states of $\mathfrak{N}$ one of whose components is $*$ (these all have the form $\langle q_i, \wedge, \wedge, \ldots, \wedge, * \rangle$) *star states.* We now prove the following proposition:

An ω-word x is stable with respect to $\langle \mathfrak{M}, q_0, Q' \rangle$ if and only if the sequence $\mathbf{v}(1), \mathbf{v}(2), \mathbf{v}(3), \ldots$ contains infinitely many star states.

This proposition directly implies that the automaton $\langle \mathfrak{N}, \pi_1 \rangle$ represents the set of stable ω-words when the limit macrostates $\{\Gamma\}$ are defined by the condition: each Γ contains at least one star state. Therefore, to complete the proof of the theorem it remains to verify the truth of this proposition. This is most simply done by means of an imaginary experiment with an unlimited stock of copies $\mathfrak{M}_1, \mathfrak{M}_2, \ldots$ started at times $\mu_1, \mu_2, \ldots$, where $x_0^{\mu_i} \in \omega(\mathfrak{M}, q_0, Q')$.

Let $\mathbf{v}(\tau_1)$ be the first star state (if these exist). This means that τ_1 is the earliest time at which one of the previously started copies reaches the same state as $\mathfrak{M}$. Since $\mathbf{v}(\tau_1)$ contains no q-components other than $v_1(\tau_1)$, the next passage through a star state may occur only at the first instant τ_2 when one of the copies started later than τ_1 but no later than τ_2 reaches the same state as $\mathfrak{M}_1$. Similarly, the third passage through a star state occurs at the first instant τ_3 when the state of a copy started later than τ_2 but no later than τ_3 coincides with the state of $\mathfrak{M}_2$, and so on. It is now clear that the existence of an infinite set of star states implies that x is stable. But it is also clear that if x is stable one can always recursively define an increasing sequence $\tau_1 < \tau_2 < \ldots$ with the above property. This proves the lemma.

Proof of the Strong Iteration Theorem. Let $\mathfrak{A} = \omega(\mathfrak{M}, q_0, Q')$. Without loss of generality we may assume that $\mathfrak{A} = \mathfrak{A}^*$, since otherwise we can replace $\mathfrak{A}$ by its (ordinary) iteration closure $\mathfrak{A}^*$ (since $(\mathfrak{A}^*)^\infty = \mathfrak{A}^\infty$ and, moreover, an automaton representing $\mathfrak{A}^*$ may be effectively constructed from an automaton representing $\mathfrak{A}$). We shall show (see **A** and **B** below)

† Meaning that the next copy is started.

that under these assumptions the ω-language $\mathfrak{B} = \mathfrak{A}^\infty$ may be expressed as $\mathfrak{A}\mathfrak{A}_{st}$, where $\mathfrak{A}_{st}$ denotes the set of all ω-words stable with respect to the automaton $\langle \mathfrak{M}, q_0, Q' \rangle$. Consequently, the ω-language $\mathfrak{B}$ is representable in an automaton whose construction is guaranteed by the Concatenation Theorem and the Stability Lemma. It remains to prove the propositions **A** and **B** stated below.

A. *If* $x \in \mathfrak{A}\mathfrak{A}_{st}$, *then* $x \in \mathfrak{A}^\infty$.

It will suffice to show that $\mathfrak{A}_{st} \subseteq \mathfrak{A}^\infty$, i.e., that every ω-word $\xi = \xi(1)\xi(2)\ldots\xi(t)\ldots$ stable with respect to $\langle \mathfrak{M}, q_0, Q' \rangle$ splits into segments $\xi_0^{\nu_1}, \xi_{\nu_1}^{\nu_2}, \ldots$ belonging to the language $\mathfrak{A} = \omega(\mathfrak{M}, q_0, Q')$. Since ξ is stable, there exists an increasing sequence $\mu_1 < \mu_2 < \mu_3 < \ldots$ such that

(I) $q_0\xi_0^{\mu_i} \in Q' \qquad (i = 1, 2, \ldots)$,

(II) each of the tails $\xi_{\mu_i}^\infty$ merges with the original ω-word ξ.

Set $\nu_1 = \mu_1$; then, by definition, $\xi_0^{\nu_1} \in \mathfrak{A}$. Now set ν_2 equal to the smallest μ_i greater than ν_1 such that $\xi_{\nu_1}^\infty$ and ξ merge no later than μ_i. It follows that $q_0 \cdot \xi_0^{\nu_2} \in Q'$, i.e., $\xi_{\nu_1}^{\nu_2} \in \mathfrak{A}$. Set ν_3 equal to the smallest μ_i greater than ν_2 such that $\xi_{\nu_2}^\infty$ and ξ merge no later than μ_i, and so the corresponding segment $\xi_{\nu_2}^{\nu_3}$ belongs to $\mathfrak{A}$. Continuing this process *ad infinitum*, we see that ξ can be split into the required segments belonging to $\mathfrak{A}$. Note that here we have not used the assumption that $\mathfrak{A} = \mathfrak{A}^*$; this will be used to prove the following proposition.

B. *If* $x \in \mathfrak{A}^\infty$, *then* $x \in \mathfrak{A}\mathfrak{A}_{st}$.

Assume given some division of x

$$x = x_0^{\nu_1} \cdot x_{\nu_1}^{\nu_2} \cdot x_{\nu_2}^{\nu_3} \ldots$$

into segments belonging to $\mathfrak{A}$. Since $\mathfrak{A} = \mathfrak{A}^*$, segments of the form $x_0^{\nu_i}$ and $x_{\nu_i}^{\nu_j}$ also belong to $\mathfrak{A}$; in other words, any of these words takes q_0 to Q'. The infinite set of tails $x_{\nu_1}^\infty, x_{\nu_2}^\infty, x_{\nu_3}^\infty, \ldots$ contains an infinite set $x_{\mu_1}^\infty, x_{\mu_2}^\infty, \ldots$ $(\mu_1 < \mu_2 < \ldots)$ of tails, each two of which are merged by the automaton $\langle \mathfrak{M}, q_0 \rangle$.

Now consider the representation $x = x_0^{\mu_1} x_{\mu_1}^\infty$. On the one hand, $x_0^{\mu_1} \in \mathfrak{A}$, since μ_1 is one of the ν_i. On the other hand, the ω-word $x_{\mu_1}^\infty$ is stable with respect to $\langle \mathfrak{M}, q_0, Q' \rangle$. In fact, by the choice of $\{\mu_i\}$ all $x_{\mu_i}^\infty$ merge with $x_{\mu_1}^\infty$, but moreover all $x_{\mu_1}^{\mu_i}$ belong to $\omega(\mathfrak{M}, q_0, Q')$, since the μ_i are ν_j. Thus $x \in \mathfrak{A}\mathfrak{A}_{st}$. This proves **B**, and hence the entire theorem.

EXAMPLE. To illustrate the algorithm described in the proof of the Strong Iteration Theorem, we deliberately choose a simple example: the strong iteration closure of the language consisting of the single word *ab*.

The algorithm starts with an automaton representing the iteration closure of the language $\{ab\}$ (the automaton $\langle \mathfrak{B}, p_1, p_1 \rangle$ of Figure 12*a*; see Example 2 of Section I.7), and constructs an automaton $\mathfrak{N}$ representing the set of all ω-words stable with respect to $\langle \mathfrak{B}, p_1, p_1 \rangle$.

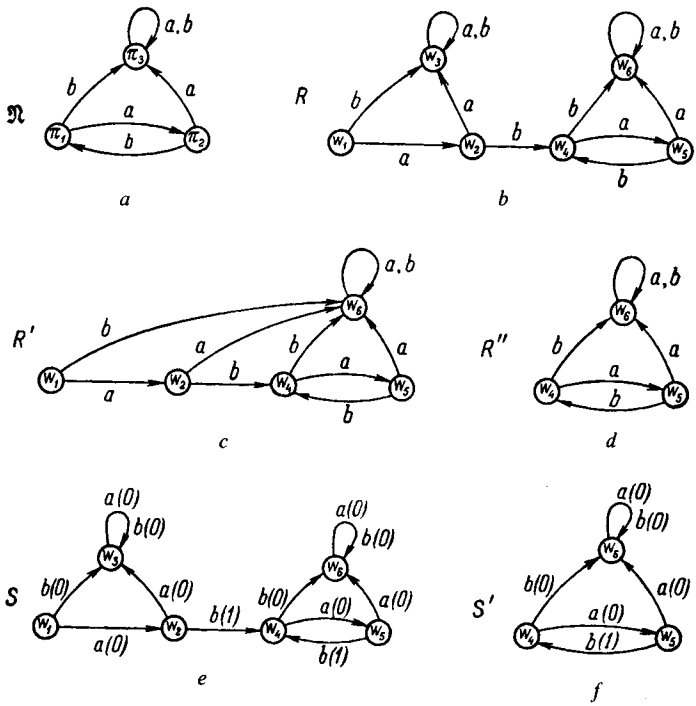


Figure 13

Following the algorithm described in the Stability Lemma, we define the initial state of the automaton $\mathfrak{N}$ to be the vector $\pi_1 = \langle p_1, \wedge, \wedge, \wedge, * \rangle$. As in the example for concatenation, we introduce new states as they arise in the transition table.

$$\pi_1 \cdot a = \langle p_2, \wedge, \wedge, \wedge, \wedge \rangle = \pi_2,$$

$$\pi_1 \cdot b = \langle p_3, \wedge, \wedge, \wedge, \wedge \rangle = \pi_3,$$

$$\pi_2 \cdot a = \langle p_3, \wedge, \wedge, \wedge, \wedge \rangle = \pi_3,$$

$$\pi_2 \cdot b = \langle p_1, p_1, \wedge, \wedge, \wedge \rangle,$$

since p_1 is the initial state of $\mathfrak{B}$. By clearing the vector $\langle p_1, p_1, \wedge, \wedge, \wedge \rangle$ in which the second component is the same as the first, we get the vector $\langle p_1, \wedge, \wedge, \wedge, * \rangle$, i.e.,

$$\pi_2 \cdot b = \pi_1.$$

Continuing, we get

$$\pi_3 \cdot a = \langle p_3, \wedge, \wedge, \wedge, \wedge \rangle = \pi_3,$$

$$\pi_3 \cdot b = \langle p_3, \wedge, \wedge, \wedge, \wedge \rangle = \pi_3,$$

and here the entries in the table break off naturally. Figure 13*a* illustrates the diagram of $\mathfrak{N}$ (apart from the renaming of states it is the same as Figure 12*a*). The limit macrostates are all sets of vertices in $\mathfrak{N}$ containing the "star" state π_1; these are $\{\pi_1\}, \{\pi_1, \pi_2\}, \{\pi_1, \pi_3\}, \{\pi_1, \pi_2, \pi_3\}$. However, Figure 13*a* shows only one loop containing the vertex π_1—the loop passing through π_1 and π_2. Thus the macrostates $\{\pi_1\}, \{\pi_1, \pi_3\}, \{\pi_1, \pi_2, \pi_3\}$ are fictitious and may be omitted.

Thus the automaton $\langle \mathfrak{N}, \pi_1, \{\pi_1, \pi_2\} \rangle$ represents the set of all ω-words stable with respect to the automaton $\langle \mathfrak{B}, p_1, p_1 \rangle$.

Finally, in order to construct an automaton representing the strong iteration closure of $\{ab\}$, we must construct an automaton R representing $\omega(\mathfrak{B}, p_1, p_1) \cdot \Omega(\mathfrak{N}, \pi_1, \{\pi_1, \pi_2\})$. Since we have already demonstrated the concatenation-product construction in Section I.2, we carry out the construction of R without going into details.

The initial state of R is $\mathbf{w}_1 = \langle p_1, \wedge, \wedge, \wedge, \wedge \rangle$. The transition table for R is

$$\mathbf{w}_1 \cdot a = \langle p_2, \wedge, \wedge, \wedge, \wedge \rangle = \mathbf{w}_2, \qquad \mathbf{w}_4 \cdot a = \langle p_2, \pi_2, \wedge, \wedge, \wedge \rangle = \mathbf{w}_5,$$

$$\mathbf{w}_1 \cdot b = \langle p_3, \wedge, \wedge, \wedge, \wedge \rangle = \mathbf{w}_3, \qquad \mathbf{w}_4 \cdot b = \langle p_3, \pi_3, \wedge, \wedge, \wedge \rangle = \mathbf{w}_6,$$

$$\mathbf{w}_2 \cdot a = \langle p_3, \wedge, \wedge, \wedge, \wedge \rangle = \mathbf{w}_3, \qquad \mathbf{w}_5 \cdot a = \langle p_3, \pi_3, \wedge, \wedge, \wedge \rangle = \mathbf{w}_6,$$

$$\mathbf{w}_2 \cdot b = \langle p_1, \pi_1, \wedge, \wedge, \wedge \rangle = \mathbf{w}_4, \qquad \mathbf{w}_5 \cdot b = \langle p_1, \pi_1, \wedge, \wedge, \wedge \rangle = \mathbf{w}_4,$$

$$\mathbf{w}_3 \cdot a = \langle p_3, \wedge, \wedge, \wedge, \wedge \rangle = \mathbf{w}_3, \qquad \mathbf{w}_6 \cdot a = \langle p_3, \pi_3, \wedge, \wedge, \wedge \rangle = \mathbf{w}_6,$$

$$\mathbf{w}_3 \cdot b = \langle p_3, \wedge, \wedge, \wedge, \wedge \rangle = \mathbf{w}_3, \qquad \mathbf{w}_6 \cdot b = \langle p_3, \pi_3, \wedge, \wedge, \wedge \rangle = \mathbf{w}_6.$$

The diagram of R is given in Figure 13*b*. The limit macrostate is $\{\mathbf{w}_4, \mathbf{w}_5\}$. Thus the strong iteration closure of the language $\{ab\}$ (consisting of the single ω-word *ababab*...) is represented by the automaton $\langle R, \mathbf{w}_1, \{\mathbf{w}_4, \mathbf{w}_5\} \rangle$.

This macroanchored automaton can be simplified. Merging the absorbing states $\mathbf{w}_3$ and $\mathbf{w}_6$ converts the automaton into the equivalent automaton $\langle R', \mathbf{w}_1, \{\mathbf{w}_4, \mathbf{w}_5\}\rangle$ of Figure 13*c*. Now merge the equivalent states $\mathbf{w}_2$ and $\mathbf{w}_6$, and then the equivalent states $\mathbf{w}_1$ and $\mathbf{w}_4$. The result is the automaton $\langle R'', \mathbf{w}_4, \{\mathbf{w}_4, \mathbf{w}_5\}\rangle$ of Figure 13*d*.

The diagram R'' (Figure 13*d*) is isomorphic to the diagram $\mathfrak{B}$ of Figure 12*a*, under the mapping $\mathbf{w}_4 \to p_1$, $\mathbf{w}_5 \to p_2$, $\mathbf{w}_6 \to p_3$, and so the automaton $\langle \mathfrak{B}, p_1, \{p_1, p_2\}\rangle$ also represents $\{ab\}^\infty$.

I.11. Probabilistic automata

Probabilistic automata are a generalization of ordinary (nonprobabilistic or deterministic) automata. We shall confine ourselves to finite probabilistic automata, but it should be clear from the context how to extend the discussion to infinite automata (automata with denumerably many states). In a deterministic automaton, an input letter $a \in X$ takes each state $q_i \in Q = \{q_1, q_2, \ldots, q_k\}$ to a completely determined state $\Psi(q_i, a) \in Q$. In a probabilistic automaton, however, the input letter a may take q_i into any state $q_j \in Q$ with a certain probability $\pi(a, q_i, q_j)$. These transition probabilities are assumed constant and independent of time and the preceding inputs; for any fixed $a \in X$ and $q_i \in Q$:

$$\sum_{q_j \in Q} \pi(a, q_i, q_j) = 1.$$

We may thus formulate the following definition:

A *probabilistic automaton* $\mathfrak{M}$ is a triple $\langle Q, X, \pi\rangle$, where Q and X are finite alphabets (internal and input, respectively), and π (the transition probability function) is a mapping of $X \times Q \times Q$ into the interval $[0, 1]$ such that

$$\sum_{q_j \in Q} \pi(a, q_i, q_j) = 1 \qquad (a \in X, q_i \in Q).$$

In a certain sense, a deterministic automaton may be regarded as a special (degenerate) case of a probabilistic automaton, in which, for any fixed a, q_i, there is a single state $\Psi(q_i, a)$ such that $\pi(a, q_i, \Psi(q_i, a)) = 1$; for all q_v other than $\Psi(q_i, a)$ we have $\pi(a, q_i, q_v) = 0$. The other extreme, in a certain sense, is represented by the so-called actual probabilistic automata, in which $\pi(a, q_i, q_j) \neq 0$ for every a, q_i, q_j (and so also $\pi(a, q_i, q_j) \neq 1$). The transition probability function π may be specified by a system of square matrices corresponding to the input letters: each input letter a is associated

with a matrix $\pi(a)$ whose rows and columns correspond to the internal states of the automaton; the entry at the intersection of the i-th column and the j-th row is $\pi(a, q_i, q_j)$. Given the function π, one can determine and compute the probabilities of various events arising in the automaton $\mathfrak{M}$, in a natural manner. The definition of the function π may be extended to include arbitrary input words instead of letters. Thus, given an input word $x = x(1)x(2)\dots x(\mu)$ we define $\pi[x, q_i, q_j]$ to be the probability that the automaton $\mathfrak{M}$ will go from state q_i to state q_j when the input word x is received. We can then associate with every input word x a transition probability matrix $\pi[x]$. It is easily seen that $\pi[x]$ is the product of the matrices corresponding to the input letters:

$$\pi[x] = \pi[x(1)]\pi[x(2)]\dots\pi[x(\mu)].$$

For example, consider the probabilistic automaton $\mathfrak{M}_1$ with two states q_1, q_2, input alphabet $\{0, 1\}$ and the following transition probability matrices:

$$\pi(0) = \begin{pmatrix} 1 & 0 \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}, \qquad \pi(1) = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ 0 & 1 \end{pmatrix}.$$

In this automaton, state q_1 is maintained with probability 1 when zero is received, state q_2 when 1 is received. In the other cases transitions to either of the states q_1, q_2 are equiprobable. The transition probability matrix for the input word 10 is:

$$\begin{pmatrix} \frac{3}{4} & \frac{1}{4} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}$$

In particular, we see that the probability of going from q_1 to q_2 in response to the input word 10 (the probability we have denoted by $\pi[10, q_1, q_2]$) is $\frac{1}{4}$, or, in binary notation, 0.01. Comparing the digits after the point with the input word 10, we notice that they form the reflection of the word 10. This is no accident, and we shall prove the following assertion (which will also be used later): *For any input word* $x(1)x(2)\dots x(\mu)$ *of the automaton* $\mathfrak{M}_1$,

$$\pi[x(1)x(2)\dots x(\mu), q_1, q_2] = 0.x(\mu)x(\mu - 1)\dots x(1),$$

where the right-hand side represents a number in binary expansion.

The proof proceeds by induction on the length of words. For a one-letter word the assertion is clear from the matrices $\pi(0)$ and $\pi(1)$: for $\pi(0, q_1, q_2)$ and $\pi(1, q_1, q_2)$ are the elements at the upper right-hand corners of these

matrices, and in binary notation they are 0.0 and 0.1, respectively. Assuming the assertion true for words of length $\mu - 1$, let us compute $\pi[x(1)\dots x(\mu), q_1, q_2]$:

$$\begin{aligned}\pi[x(1)\dots x(\mu), q_1, q_2] &= \pi[x(1)\dots x(\mu-1), q_1, q_2]\alpha + \\ &+ \pi[x(1)\dots x(\mu-1), q_1, q_1]\beta = \\ &= \pi[x(1)\dots x(\mu-1), q_1, q_2][\alpha - \beta] + \beta,\end{aligned}$$

where $\alpha = \pi[x(\mu), q_2, q_2], \beta = \pi[x(\mu), q_1, q_2]$.

Now, by the induction hypothesis,

$$\pi[x(1)\dots x(\mu), q_1, q_2] = (0.x(\mu-1)\dots x(1))[\alpha - \beta] + \beta.$$

If $x(\mu) = 0$, then $\alpha - \beta = \frac{1}{2}$ and $\beta = 0$, and so the rough probability is $0.0x(\mu-1)\dots x(1)$. But if $x(\mu) = 1$, then $\alpha - \beta = \beta = \frac{1}{2}$ and we get $0.1x(\mu-1)\dots x(1)$.

In general, any fixed input word $x = x(1)\dots x(\mu)$ and initial state q_i of a probabilistic automaton induce a measure (probability) over the set (space) of all words of length $\mu + 1$ over the alphabet Q that begin with the letter q_i. To be precise, given any set $\mathfrak{A}$ of such words, one can consider the probability that, when the word $x(1)\dots x(\mu)$ is applied to the automaton in state q_i, it will go through a sequence of states $q_{s_1}, q_{s_2}, \dots, q_{s_\mu}$ such that the word $q_i q_{s_1} \dots q_{s_\mu}$ belongs to $\mathfrak{A}$. In view of the analogy with deterministic automata, we shall speak of the probability that application of the word $x(1)\dots x(\mu)$ generates a path in $\mathfrak{A}$. This probability is clearly the sum of probabilities of the separate paths in $\mathfrak{A}$, and the probability of each path is the product of the corresponding transition probabilities. Similarly, one can consider the probability induced over the space of all ω-words over the alphabet Q for a fixed input ω-word $x = x(1)x(2)\dots$ and initial state q_i and, in particular, the probability that this ω-word will generate an ω-path with some prescribed property.

For example, consider the probability that application of the ω-word $x = 1111\dots1$ (all ones) to the automaton $\mathfrak{M}_1$ will generate an ω-path whose limit macrostate is the singleton $\{q_2\}$. If $\mathfrak{M}_1$ is started in state q_2, this probability is 1; moreover, the probability that the ω-path $q_2q_2q_2\dots$ will be generated is 1. If $\mathfrak{M}_1$ is started in state q_1, the probability of the limit macrostate $\{q_2\}$ is again 1, but the probability of the ω-path $q_2q_2q_2\dots$ is 0. Given any probabilistic automaton $\mathfrak{M}$, input word x, state q_i and macrostate Γ, the probability that application of x in state q_i generates an ω-path with limit macrostate Γ will be denoted by $p(x, q_i, \Gamma)$.

The analogy between probabilistic and deterministic automata extends to the representation of languages and ω-languages. However, here the language (ω-language) represented by the automaton will depend not only on the initial and final states (limit macrostates), but also on a real parameter λ, $0 \leqq \lambda < 1$, which may be interpreted as the "confidence level" of the representation in question. We shall employ the following definitions.

An *anchored probabilistic automaton* is a quadruple $\langle \mathfrak{M}, q_0, Q', \lambda \rangle$, where $\mathfrak{M}$ is a probabilistic automaton, q_0 a designated initial state, Q' a designated set of final states and λ a real number in the half-interval $[0, 1]$. The *language represented by this anchored automaton* (denoted by $\omega(\mathfrak{M}, q_0, Q', \lambda)$) is defined as the set of all words which, with probability greater than λ, take the state q_0 to a state in Q'. A similar definition applies to a *macroanchored probabilistic automaton* $\langle \mathfrak{M}, q_0, \mathfrak{C}, \lambda \rangle$ and the *ω-language* $\Omega(\mathfrak{M}, q_0, \mathfrak{C}, \lambda)$ that it *represents.*

We have already explained in what sense any deterministic automaton may be considered as a special (degenerate) case of a probabilistic automaton. A word (ω-word) belongs to the language (ω-language) represented by a finite deterministic automaton if and only if it generates an appropriate path (ω-path) with probability 1 when the automaton is regarded as probabilistic. Thus any language representable in a finite deterministic automaton is trivially representable in a finite probabilistic automaton. Moreover, the transition from deterministic to probabilistic automata does not increase the number of internal states. However, the converse proposition is false, as the following theorem will show.

THEOREM 1.14. *Consider the family of all languages over the alphabet* $\{0, 1\}$ *(ω-languages over the alphabet* $\{0, 1, 2\}$*) representable in probabilistic automata with two states. Then: a) This family contains languages (ω-languages) which are not representable in any finite automaton. b) For any n, there are languages (ω-languages) in this family which are representable only in finite automata with at least n states.*

Proof. a) Consider the automaton $\mathfrak{M}_1$ described on page 67. There are rational numbers between any two real numbers $\lambda_1, \lambda_2, 0 < \lambda_1 < \lambda_2 < 1$; let the binary expansion of one of these rationals be $0.\xi_1\xi_2 \ldots \xi_\mu$ ($\xi_i = 0$ or 1). Then the word $\xi_\mu\xi_{\mu-1} \ldots \xi_1$ belongs to the language $\omega(\mathfrak{M}_1, q_1, q_2, \lambda_1)$ but not to $\omega(\mathfrak{M}_1, q_1, q_2, \lambda_2)$. In other words, different λ give rise to different languages $\omega(\mathfrak{M}_1, q_1, q_2, \lambda)$. Since the set of possible λ is nondenumerable, while there are only denumerably many languages over $\{0, 1\}$ representable in finite automata, a standard set-theoretic argu-

ment implies the existence of languages $\omega(\mathfrak{M}_1, q_1, q_2, \lambda)$ which are not representable in finite automata. To prove this for ω-languages, consider the following probabilistic automaton $\mathfrak{M}_2$ with input alphabet $\{0, 1, 2\}$ and states $\{q_1, q_2\}$: the transition probability matrices for the input letters 0, 1 are the same as for $\mathfrak{M}_1$; the letter 2 maintains the state with probability 1 (and therefore takes each state into the other with probability 0).

Now consider any ω-word $x(1)\dots x(\mu)x(\mu+1)\dots$, where $x(1), \dots, x(\mu)$ are 0 or 1 and $x(\mu+1), x(\mu+2), \dots$ are all 2's. It is easily verified that if this word is applied to $\mathfrak{M}_2$ in the initial state q_1, the limit macrostate will be $\{q_2\}$ with probability $0.x(\mu)x(\mu-1)\dots x(1)$. Thus, as in the previous case, it follows that different λ correspond to different ω-languages $\Omega(\mathfrak{M}_2, q_1, \{q_2\}, \lambda)$, and so some of these ω-languages are not representable in deterministic finite automata.

b) It suffices to prove that the set of languages $\omega(\mathfrak{M}_1, q_1, q_2, \lambda)$ contains an infinite subset of languages representable in finite deterministic automata, and so the number of states in the corresponding representing automata cannot be bounded.

Let λ_n be the number whose binary expansion is $0.11\dots1$ (n ones). The language $\omega(\mathfrak{M}_1, q_1, q_2, \lambda_n)$ consists of all words $x(1)\dots x(\mu)$ such that

$$0.x(\mu)x(\mu-1)\dots x(1) > \lambda_n.$$

In other words, $\omega(\mathfrak{M}_1, q_1, q_2, \lambda_n)$ consists of precisely those words which contain at least $n+1$ ones and whose last n letters are ones. Thus the languages $\omega(\mathfrak{M}_1, q_1, q_2, \lambda_n)$ ($n = 1, 2, 3, \dots$) are representable in finite deterministic automata; moreover, they are pairwise distinct.

A similar argument shows that each of the ω-languages $\Omega(\mathfrak{M}_2, q_1, \{q_2\}, \lambda_n)$ is representable in a deterministic finite automaton, and since these ω-languages are different for different λ_n, the number of states in the corresponding deterministic automata cannot be bounded. In fact, the ω-language $\Omega(\mathfrak{M}_2, q_1, \{q_2\}, \lambda_n)$ is the union of ω-languages Ω'_n and Ω''_n, where

1) Ω'_n consists of all ω-words containing only finitely many occurrences of the letter 0 and infinitely many occurrences of the letter 1; the number of occurrences of the letter 2 is unrestricted; all these ω-words generate the limit set $\{q_2\}$ with probability 1;

2) Ω''_n consists of all ω-words of type $x(1)\dots x(\mu)x(\mu+1)\dots$, where a) $x(\mu+1) = x(\mu+2) = \dots = 2$; b) the word derived from $x(1)\dots x(\mu)$ by deleting all occurrences of 2 contains at least $n+1$ ones, and the last n letters are ones.

I.12. Grammars and automata

The concepts of production grammars and the languages that they generate are of central importance in mathematical linguistics; at the same time, they are related to the concept of automata and the languages they represent. Therefore, in principle, various important propositions of the theory of grammars may be related to the general theory of automata. There are many different conceptions of grammar, differing in their degree of generality. Any production grammar G is specified by giving its vocabulary V and its system of (grammatical) rules π [generally called *productions*]. The vocabulary V is a finite set of symbols. Here the terms "symbol" and "vocabulary" are preferred to our previous terms "letter" and "alphabet," since in the linguistic interpretation each individual symbol in V is a word (or, more precisely, a word-form), and a string of symbols from V (which we have been calling a word over V) represents a sentence. The vocabulary V consists of symbols of two types: basic symbols,* forming a subvocabulary V_T, and auxiliary symbols—subvocabulary V_N. One of the auxiliary symbols (denoted by S) is designated as an initial symbol.

The productions single out a subset of *well-formed* strings (in the sense of the given grammar) from the set of all possible strings over V. This subset is called the language generated by the grammar G. In other words, in the linguistic interpretation the language generated by a grammar is the set of all grammatically well-formed sentences. Each production of the grammar has the form $\phi \to \psi$, where ϕ, ψ are strings over V and $\to$ does not belong to V. The production $\phi \to \psi$ is applicable to a string P if P contains at least one occurrence of ϕ as a subword; application of the production to P replaces one occurrence of ϕ by the word ψ. A sequence of strings $D = (\phi_1, \ldots, \phi_n)$ is said to be an *ω-derivation* of the string ρ if the following conditions hold: $\omega = \phi_1, \rho = \phi_n$, and for every $i < n$, ϕ_{i+1} is derived from ϕ_i by (a single) application of a production.

A string is said to be *well-formed* if

(I) it consists of only basic letters;

(II) it has an S-derivation (i.e., it may be derived from the initial auxiliary symbol S);

(III) no production of the grammar is applicable to it (i.e., the derivation of this string cannot be continued).

As mentioned above, the set of all strings which are well-formed with

* *Translator's note*: Usually called *terminal symbols* in Western literature.

respect to a given grammar is called the language generated by the grammar.

Narrower classes of grammars may be defined by imposing various restrictions on the productions. A grammar whose productions are of the form $\eta_1 A \eta_2 \to \eta_1 \xi \eta_2$, where $A \in V_N$ and η_1, η_2 are strings over V, while ξ is a nonempty string over V, is known as an *immediate constituent grammar** (IC-grammar). In turn, the class of IC-grammars contains smaller classes.

For example, consider the class of *right-linear grammars*. These have productions of type $q \to q'x$ (nonterminal production) or $q \to x$ (terminal production), where q and q' are auxiliary symbols and x a basic symbol. It is clear that all S-derivable strings over the basic vocabulary, and only these, belong to the language generated by this grammar. One can associate a source with every right-linear grammar, in the following way. The states (vertices) of the source are all auxiliary symbols of the grammar, plus one special symbol (say $\wedge$) which is the single final state of the source. The sole initial vertex of the source is S (the initial auxiliary symbol). Now, for each nonterminal production $q \to q'x$, draw an edge from q to q' and label it x. For each terminal production $q \to x$, draw an edge from q to $\wedge$ labeled x. The result is a source whose input alphabet is precisely the basic vocabulary of the grammar, and it is easy to see that a string is regular with respect to the grammar if and only if it is carried by some path from S to $\wedge$. Thus the language generated by a right-linear grammar is precisely the language represented by a special type of source: a source with a single initial vertex and a single final vertex, the latter being an output terminal. Since, as we know, every source is equivalent to a two-terminal, the class of languages generated by right-linear grammars is precisely the class of languages representable in finite automata.

By considering productions of the form $q \to xq'$ instead of $q \to q'x$, we arrive in a natural way at the concept of a *left-linear* grammar. Here again one can associate a source with every left-linear grammar and prove that the class of languages generated by such grammars is identical with the class of finite-state languages.

Left-linear and right-linear grammars are special cases of linear grammars.

A grammar is said to be *linear* if it contains productions of the form $q \to x$, right-linear productions $q \to q'x$, left-linear productions $q \to xq'$, and also two-sided-linear productions $q \to xq'y$.

* *Translator's note*: Sic. The definition is that of a *context-sensitive grammar*.

As an example, we describe a linear grammar G_1 generating

The language $L_1 = \{0^n 1 0^n\}$ $(n = 0, 1, 2, \dots)$. Since this language is not representable in a finite automaton, this will show that the class of languages generated by linear grammars (or, as they are called, linear languages) is larger than the class of finite-state languages. The vocabulary V of G_1 consists of basic symbols 0, 1 and one auxiliary symbol S. There is one nonterminal production $S \to 0S0$, and one terminal production $S \to 1$.

Similarly, we can construct a linear grammar G_2 generating

The language L_2. This language consists of all strings over the alphabet $\{0, 1, *\}$ having the form $w_1 * w_2$, where $w_2 = w_1^{-1}$ (i.e., w_2 is the reflection of w_1) and moreover w_2 (therefore also w_1) does not contain the symbol $*$.

The following examples of linear grammars and languages will not only illustrate the mechanics of language-generation by grammars, but we shall also use them later in proving a theorem in Section III.3. In all these examples the basic vocabulary is $\{0, 1, c, *\}$; $w_1, w_2, \dots$ will always denote strings over the subvocabulary $\{0, 1\}$.†

The language L_3 consists of all strings of the form $w_1 * w_2$, where $w_2 \neq w_1^{-1}$. Each of these strings has (at least) one of the following forms:

1) $w_1' 0 w * w^{-1} 1 w_2'$ or $w_1' 1 w * w^{-1} 0 w_2'$,

2) $vw * w^{-1}$,

3) $w * w^{-1} v$,

where v is a nonempty word.

The grammar G_3 contains three auxiliary symbols S', S'', S''', from which words of these three types are derived, respectively.

The productions of the grammar are as follows:

0) $S \to S'$, $S \to S''$, $S \to S'''$,

1) $S' \to S'0$, $S' \to S'1$, $S' \to 0S'$, $S' \to 1S'$, $S' \to 0q1$, $S' \to 1q0$.

Productions 0) and 1) generate all words of the form $w_1' 0 q 1 w_2'$ and $w_1' 1 q 0 w_2'$.

2) $S'' \to 0S''$, $S'' \to 1S''$, $S'' \to 0q$, $S'' \to 1q$,

3) $S''' \to S'''0$, $S''' \to S'''1$, $S''' \to q0$, $S''' \to q1$.

These productions generate all words of the form vq and qv. Finally, the grammar is completed by the productions

4) $q \to 0q0$, $q \to 1q1$, $q \to *$.

The language L_4 consists of all strings

$$cw_1 cw_2 c \dots cw_n c * cw_{n+1} cw_{n+2} c \dots cw_{n+k} c$$

† Here and below $w_1, w_2, \dots$ may also be empty strings.

such that the string $w_1 w_2 \dots w_n$ (the concatenation of the strings $w_1, w_2, \dots, w_n$) is not the reflection of the string $w_{n+1} w_{n+2} \dots w_{n+k}$. A suitable linear grammar G_4 may be constructed by a natural modification of the grammar G_3; the details are left to the reader.

The language L_5 depends on a fixed system of pairs of strings over the subvocabulary $\{0, 1\}$:

$$(\xi_1, \eta_1), (\xi_2, \eta_2), \dots, (\xi_n, \eta_n). \qquad (\#)$$

The language consists of all strings of type $cw_1c * cw_2c$ in which the pair w_1, w_2 is not one of the pairs (#). It is easily seen that L_5 is a finite-state language. A suitable right-linear grammar G_5 may be constructed by first constructing a two-terminal source representing L_5, and then using the above-described correspondence between right-linear grammars and sources.

The language L_6 also depends on the system (#); it consists of the strings

$$cw_1cw_2c \dots cw_nc * cw_{n+1}cw_{n+2}c \dots cw_{n+k}c$$

satisfying the condition: for at least one i, the pair of strings w_{n-i+1}, w_{n+i} (situated "symmetrically" with respect to the symbol $*$) is not one of the pairs (#).

The corresponding grammar contains the following productions:

1) $S \to cS'c$; $S' \to S'0$; $S' \to S'1$; $S' \to 0S$;
$S' \to 1S'$; $S' \to cS'$; $S' \to S'c$; $S' \to cS''c$;
$S \to ccS''cc$.

The strings derived from S by productions of this group and containing the symbol S'' are precisely the strings $c\alpha cS''c\beta c$, where α, β are arbitrary (possibly empty) strings over the vocabulary $\{0, 1, c\}$.

2) The productions of the second group are slight modifications of the productions of the grammar G_5. Together with the productions of group 1), they generate all strings

$$c\alpha cw'cS'''cw''c\beta c, \qquad (1)$$

where α, β are arbitrary strings over $\{0, 1, c\}$, the pair w', w'' is not one of the pairs (#) and S''' is an auxiliary symbol, replacing the basic symbol $*$ of G_5.

3) The third and last group of productions is used to extend the strings (1) by inserting the same number of words w braced by separating symbols

c on both sides of S''', and finally replacing the auxiliary symbol S''' by $*$:

$$S''' \to S'''0;\ S''' \to S'''1;\ S''' \to 0S''';\ S''' \to 1S''';$$

$$S''' \to cS'''c;\ S''' \to c * c.$$

It is easy to see that this linear grammar generates L_6.

A wider class than linear grammars is that of *context-free grammars* (CF-grammars). Their productions have the form

$$q \to \alpha,$$

where q is an auxiliary symbol and α an arbitrary string over the initial vocabulary. If α contains no occurrence of auxiliary symbols, application of the production reduces the number of such occurrences; as in the special case of linear grammars, we call productions of this type terminal. If the production is not terminal and contains more than one occurrence of auxiliary symbols, its application to a string increases the number of occurrences of auxiliary symbols. Thus, for derivations in CF-grammars there is no general upper bound for the number of occurrences of auxiliary letters; by contrast, at every step of a derivation in a linear grammar, up to application of a terminal production, there is exactly one occurrence of an auxiliary symbol.

All our examples hitherto have of course been CF-grammars and CF-languages. We now give a simple example of a language which is not context-free.

The language L_7 consists of all strings $a^n b^n a^n$. Suppose that L_7 is generated by a CF-grammar G, i.e., all words of the form $a^n b^n a^n$, and only such words (an infinite set), are derivable from the initial symbol of G. Without loss of generality we may assume that if any other auxiliary symbol q is considered as initial symbol, it also generates an infinite set of strings over the basic vocabulary $\{a, b\}$. In fact, if only finitely many strings $z_1, z_2, \ldots, z_\nu$ can be derived from q, replace all occurrences of q in the right-hand sides of the productions of G by the strings $z_1, z_2, \ldots, z_\nu$; the result is an equivalent CF-grammar which does not contain the auxiliary symbol q (and no new auxiliary symbols are added). Now assume that the maximum number of symbols in the right-hand sides of the productions of G is r, and consider some derivation of the string $a^{r+1}b^{r+1}a^{r+1}$. The last step of this derivation must be the application of a terminal production $q \to \alpha$, converting a string $w_1 q w_2$, where w_1 and w_2 are strings over $\{a, b\}$, into the string $a^{r+1}b^{r+1}a^{r+1}$:

$$\overbrace{a \ldots a}^{w_1}\ \overbrace{b \ldots b}^{\alpha}\ \overbrace{a \ldots a}^{w_2} \text{ (the length of } \alpha \text{ is at most } r\text{)}.$$
$$\underbrace{\qquad\qquad}_{r+1}$$

Now let β be some other string over $\{a, b\}$, different from α but derivable from q.* Then the string $w_1\beta w_2$ is also derivable from S and so belongs to L_7. But it is clear that if $\beta \neq \alpha$ the string $w_1\beta w_2$ cannot have the form $a^n b^n a^n$ for any $n = 1, 2, \ldots$ This contradiction shows that the language L_7 is not generated by any CF-grammar.

To conclude this section, we present a theorem concerning the closure properties of the class of CF-languages under set-theoretic operations.

THEOREM 1.15. *The class of CF-languages is closed under set-theoretic union, but not under intersection and complementation.*

Proof. Let L' and L'' be languages generated by CF-grammars G' and G'', respectively. Rename the auxiliary symbols of G' and G'' in such a way that G' and G'' contain no common auxiliary symbols other than the initial symbol S. Combining the productions of both grammars, we get a CF-grammar $G = G' \cup G''$.

We now present an example of two CF-languages L' and L'' whose intersection is not a CF-language. L' consists of the strings $a^n b^n a^m$, L'' of the strings $a^m b^n a^n$ $(n = 1, 2, \ldots; m = 1, 2, \ldots)$. L' is generated by the grammar whose productions are

$$S \to S'S''; \qquad S'' \to aS''; \qquad S'' \to a;$$
$$S' \to aS'b; \qquad S' \to ab.$$

The construction of a CF-grammar for L'' is analogous. Now the intersection of L' and L'' is precisely the set of all strings $a^n b^n a^n$ and, as we have just shown, this language is not context-free. It follows immediately that the class of CF-languages is not closed under complementation (since intersection may be expressed in terms of union and complementation). Q.E.D.

REMARK. It is clear from the proof that a CF-grammar generating $L' \cup L''$ may be effectively constructed from the grammars generating L' and L''.

* By assumption, there are infinitely many such strings.

Supplementary material, problems, examples

I. *Definition.* Let U be the universal language over an alphabet A. An equivalence relation R over U is said to be *right (left) invariant* if xRy implies $xzRyz$ ($zxRzy$) for any word z (x, y, z may be empty). R is a *congruence relation* if it is both right and left invariant.

A language $\mathfrak{A}$ is representable in a finite automaton if and only if there exists an equivalence relation R which is right invariant (left invariant, a congruence relation) and partitions U into a finite number of equivalence classes such that $\mathfrak{A}$ is the union of some of these classes (Nerode [111], Myhill [110]).

II. Following Chomsky and Miller [82], one can associate a function $v_R(n)$ with any language R over a finite alphabet, setting $v_R(n)$ equal to the number of words in R of length n.

If the language R is finite-state, the function $v_R(n)$ may be expressed in the form

$$v_R(n) = \sum a_i(n) n^{k_i} \rho_i^n.$$

The summation extends over a certain finite set of indices (depending on the language R), $a_i(n)$ is a periodic function, k_i a positive integer, ρ_i a positive number, and moreover $a_i(n)$ and ρ_i are algebraic (Chomsky–Miller [82], Plesnevich [50]). Chomsky and Miller assert that the general form of the function $v_R(n)$ is $\sum a_i \lambda_i^n$ where a_i and λ_i are complex; the following counterexample shows that this is false. Let R be the finite-state language consisting of all binary words containing exactly one occurrence of 1. Then $v_R(n) = n$; but this function cannot be expressed in the above form.)

III. Show that there exists an algorithm which, for any two anchored (macroanchored) automata, determines whether the intersection of the languages (ω-languages) that they represent is empty or infinite.

IV. Let $\mathfrak{M}$ be an automaton with r states. The language represented by $\mathfrak{M}$ is infinite if and only if it contains at least one word of length n, where $r \leqq n \leqq 2r$ (Rabin–Scott [114]).

V. Given any language $\mathfrak{A}$, define a function $\mathfrak{A}(t)$ for $t = 0, 1, 2, \ldots$ whose values are (possibly empty) languages, as follows: $\mathfrak{A}(t)$ consists of all words that can be obtained from words in $\mathfrak{A}$ of length greater than t by deleting initial segments of length t.

Show that if $\mathfrak{A}$ is representable in a finite automaton, then the sequence of languages

$$\mathfrak{A}(0), \mathfrak{A}(1), \mathfrak{A}(2), \ldots$$

is ultimately periodic (i.e., there exist $t^0 \geqq 0$—the phase—and $T > 0$—the period—such that $\mathfrak{A}(t + T) = \mathfrak{A}(t)$ for $t \geqq t^0$).

Let $\mathfrak{A}$ and its complement $\neg\mathfrak{A}$ be languages such that the functions $\mathfrak{A}(t)$ and $\neg\mathfrak{A}(t)$ are purely periodic ($t^0 = 0$). Then $\mathfrak{A}$ is representable in a finite automaton (Lyubich [46]).

VI. Let $\rho(\mathfrak{A}, \mathfrak{B})$ be an arbitrary meaningful expression formed from the symbols $\mathfrak{A}, \mathfrak{B}$, designating languages, and the operation symbols $\cup, \cdot, *$ (an expression of this type is called a *term* in the signature $\mathfrak{A}, \mathfrak{B}, \cup, \cdot, *$). Show that for any languages $\mathfrak{A}$, $\mathfrak{B}$

$$\rho(\mathfrak{A}, \mathfrak{B}) \subseteq (\mathfrak{A} \cup \mathfrak{B})^*$$

(McNaughton [103]).

Show that for any three languages $\mathfrak{A}, \mathfrak{B}, \mathfrak{C}$ the equality $\mathfrak{A} = \mathfrak{A} \cdot \mathfrak{B} \cup \mathfrak{C}$ holds if and only if $\mathfrak{A} = \mathfrak{C}\mathfrak{B}^* \cup \mathfrak{C}$.

VII. Call $f(\mathfrak{A}, \mathfrak{B})$ a *binary substitution operation* on languages over an alphabet X if there exists a language σ over the alphabet $X \cup \{a, b\}$ $(a, b \notin X)$ such that for any languages $\mathfrak{A}, \mathfrak{B}$ over X the language $f(\mathfrak{A}, \mathfrak{B})$ consists of all words obtained from words of σ by replacing each occurrence of a by a word of $\mathfrak{A}$ and each occurrence of b by a word of $\mathfrak{B}$. Different occurrences of a (or b) may be replaced by different words of $\mathfrak{A}$ (or $\mathfrak{B}$). The definition of unary, ternary, etc., substitution operations is analogous (McNaughton [103]).

Which of the operations on languages considered in Chapter I are substitution operations and which are not? State a natural definition of unary, binary, etc., substitution operations transforming languages into ω-languages (or ω-languages into ω-languages)?

VIII. The class of all languages (over a fixed alphabet) forms an algebra (the so-called algebra of events) under the operations $\cup, \cdot, *$.

Which of the identities listed below are valid in this algebra?

1) $\mathfrak{A}(\mathfrak{B}\mathfrak{A})^* = (\mathfrak{A}\mathfrak{B})^*\mathfrak{A}$,
2) $(\mathfrak{A} \cup \mathfrak{B})^* = (\mathfrak{A}^*\mathfrak{B}^*)^*$,
3) $\mathfrak{A}(\mathfrak{B} \cup \mathfrak{C}) = \mathfrak{A}\mathfrak{B} \cup \mathfrak{A}\mathfrak{C}$,
4) $(\mathfrak{A}^*\mathfrak{B})^* = (\mathfrak{A} \cup \mathfrak{B})^*\mathfrak{B}$
5) $\mathfrak{A}^* = \mathfrak{A} \cup \mathfrak{A}^2 \cup \ldots \cup \mathfrak{A}^{k-1} \cup \mathfrak{A}^k \cdot \mathfrak{A}^*$.

IX. The class of languages over an alphabet X which are representable in finite automata is closed under the following operations, each of which transforms a language $\mathfrak{A}$ into a language $\mathfrak{A}'$:

1) $\mathfrak{A}'$ consists of all words with an initial segment belonging to $\mathfrak{A}$.
2) $\mathfrak{A}'$ consists of all words *all* of whose initial segments belong to $\mathfrak{A}$.

Operations 1′ and 2′ are derived from operations 1 and 2 by replacing the word "initial" by "final."

3) Let a_i, a_j be letters from the alphabet $X = \{a_1, a_2, \ldots, a_m\}$. $\mathfrak{A}'$ is obtained from $\mathfrak{A}$ by replacing all occurrences of a_i in the words of $\mathfrak{A}$ by occurrences of a_j.

X. Following Medvedev, let us call the following languages over a fixed alphabet $X = \{a_1, a_2, \ldots, a_m\}$ *elementary languages*:

$\mathfrak{A}_1$ = the language consisting of all one-letter words, $\mathfrak{A}_2$ = the language consisting of all two-letter words, $\mathfrak{B}_i$ = the language consisting of all words ending in the letter $a_i (i \leqq m)$;

$\mathfrak{C}_i$ = the language consisting of all words of length at least 2 in which the penultimate letter is $a_i (i \leqq m)$.

The class of languages (over X) representable in finite automata is the smallest class of languages that a) contains all elementary languages, b) is closed under the Boolean operations $\cup$, $\neg$, &, and also under operations 1) and 3) of the preceding problem (Medvedev [48]).

XI. Consider the following four modifications of the concept of a finite macrosource, which we shall call $\exists$-sources, $\exists^\infty$-sources, $\forall^\infty$-sources, $\forall$-sources. In all cases, the set Q_0 of initial states is fixed, as is the set C of final states, and a certain "admissible" type of ω-paths is indicated. This will also define corresponding varieties of ω-languages representable in $\exists(\exists^\infty, \forall^\infty, \forall)$-sources: these consist of all ω-words carried by the admissible ω-paths. It remains to define these admissible ω-paths:

1) $\exists$-source: all ω-paths passing at least once through a vertex of C;

2) $\exists^\infty$-source: all ω-paths passing infinitely often through vertices of C;

3) $\forall^\infty$-source: all ω-paths which, from some vertex on, pass through vertices of C alone;

4) $\forall$-source: all ω-paths passing through vertices of C alone.

Let $K\exists$, $K\exists^\infty$, $K\forall^\infty$, $K\forall$ denote the classes of ω-languages representable in $\exists$-, $\exists^\infty$, $\forall^\infty$-, $\forall$-sources, respectively. Show that

$$K\forall \subset K\exists = K\forall^\infty \subset K\exists^\infty.$$

Show that $K\exists^\infty$ coincides with the class of ω-languages representable in finite automata.

XII. There exists a source with n states ($n = 1, 2, 3, \ldots$) over a two-letter alphabet (say $\{0, 1\}$) such that the minimal equivalent automaton has exactly 2^n states (Lupanov [42], Ershov [29]). Compare with Theorem 1.10 of Section I.7, which considers sources over a three-letter alphabet.

For a one-letter alphabet and a source with n states, there is always an

equivalent finite automaton with at most $2^{\mu(n)}$ states, where $\mu(n)$ is a function asymptotic to $\sqrt{n \ln n}$, and this estimate cannot be improved (Lyubich [44]). Thus, in determinization of sources over this alphabet the upper bound for the number of states of the equivalent automaton is much lower than in the many-letter case.

XIII. For every n, there is a language over $\{0, 1\}$, representable in an automaton with n states, whose reflection is not representable in any automaton with less than 2^n states.

XIV. Assume an arbitrary partition of the universal language U over a fixed alphabet (say over $\{0, 1\}$) into a finite system of pairwise disjoint languages $\mathfrak{A}_1, \mathfrak{A}_2, \ldots, \mathfrak{A}_s$. Then any ω-word over this alphabet belongs to at least one of the ω-languages $\mathfrak{A}_i \cdot \mathfrak{A}_j^{\omega}$ $(i \leqq s, j \leqq s)$ (corollary of Ramsey's Theorem [116]).

XV. Assume given a probabilistic automaton with fixed initial state q_0 and set Q_0 of final states. For each input word x, let $p(x)$ denote the probability that x takes q_0 to some state of Q_0. We shall say that a real number $\lambda (0 < \lambda < 1)$ is a δ-isolated cut-point for $\mathfrak{M}$, q_0 and Q_0 if $|p(x) - \lambda| \geqq \delta$ for any input word x. If λ is a δ-isolated cut-point for $\mathfrak{M}, q_0, Q_0$, then the language $\omega(\mathfrak{M}, q_0, Q_0, \lambda)$ is representable in a finite deterministic automaton. If $\mathfrak{M}$ has n states and Q_0 consists of r states, then a deterministic automaton representing $\omega(\mathfrak{M}, q_0, Q_0, \lambda)$ may be constructed with l states, where $l \leqq (1 + r/\delta)^{n-1}$ (Rabin [113]).

XVI. Show that if $\mathfrak{M}$ is an actual automaton (none of the transition probabilities vanish) then, for any λ $(0 < \lambda < 1)$, the ω-language $\Omega(\mathfrak{M}, q_0, \mathfrak{C}, \lambda)$ is either empty or coincides with the set of all ω-words over the input alphabet (Rabin [113]).

XVII. Let $\mathfrak{M}$ be an actual automaton and λ an isolated cut-point for fixed q_0 and Q_0. In contrast to the preceding problem, in which the proof is quite simple, the following assertions require more subtle arguments.

1. Call a language R *definite*, if there exists a natural number k such that any word of length greater than k is in R if and only if the word formed from its last k letters is in R. Prove that the language $\omega(\mathfrak{M}, q_0, Q_0, \lambda)$ is definite.

2. *Stability Theorem.* There exists $\varepsilon > 0$ such that for any probabilistic automaton $\mathfrak{M}'$ with the same alphabets (input and internal) as $\mathfrak{M}$, and with transition probabilities different from those of $\mathfrak{M}$ by less than ε, λ is an isolated cut-point for fixed q_0, Q and

$$\omega(\mathfrak{M}', q_0, Q, \lambda) = \omega(\mathfrak{M}, q_0, Q, \lambda)$$

(Rabin [113]).

XVIII. The following generalization of the notion of representability in a finite automaton is due to Rabin and Scott [114]. Define a *two-way automaton* to be an automaton $\mathfrak{M} = \langle Q, X, \Psi \rangle$ with the following designated sets: 1) initial state q_0; 2) set of final states Q'; 3) three pairwise disjoint sets of states Q_{-1}, Q_0, Q_1 (states of left, zero and right shift, respectively) whose union is Q. When presented with an input word $x(1)\dots x(\mu)$, the automaton is set to its initial state q_0 and scans the letter $x(1)$. At each step of its operation, with the automaton in an internal state q and scanning a letter $a \in X$, it passes as usual, to the state $\Psi(q, a)$, but at the same time either moves one letter to the left, remains in position, or moves one letter to the right, depending on whether $\Psi(q, a) \in Q_{-1}$, $\Psi(q, a) \in Q_0$, $\Psi(q, a) \in Q_1$.

A word $x = x(1)\dots x(\mu)$ is said to belong to the language $\omega(\mathfrak{M}, q_0, Q', Q_{-1}, Q_0, Q_1)$ if the automaton $\mathfrak{M}$, "reading" x in the above way, finally "falls off" x on the right and is then in one of the states of Q'. We say that the two-way automaton $\langle \mathfrak{M}, q_0, Q', Q_{-1}, Q_0, Q_1 \rangle$ represents the language $\omega(\mathfrak{M}, q_0, Q', Q_{-1}, Q_0, Q_1)$.

The following theorem holds: For any finite two-way automaton, one can effectively construct an (ordinary) anchored finite automaton which represents the same language (Rabin–Scott [114], Shepherdson [122]).

XIX. (Stearns and Hartmanis [124]). Let R be a finite-state language and Q an arbitrary language. Consider the languages P_1, P_2, P_3 defined as follows:

$$x \in P_1 \underset{df}{=} \exists y (y \in Q \,\&\, xy \in R),$$

$$x \in P_2 \underset{df}{=} \exists y (y \in Q \,\&\, yx \in R),$$

$$x \in P_3 \underset{df}{=} \exists y, z, u (x = yz \,\&\, u \in Q \,\&\, yuz \in R).$$

Show that P_1, P_2, P_3 are finite-state languages.

Assume that the language R contains no words of odd length. Show that the left (right) halves of the words in R form a finite-state language.

Assume that R contains only words whose lengths are multiples of three. Show that the language consisting of the "middle thirds" of the words in R is finite-state, but the language obtained by deleting the "middle thirds" from the words of R need not be finite-state. To verify the latter assertion, consider the finite-state language R consisting of all binary words whose lengths are multiples of three, which do not contain two consecutive ones.

Notes

The closure of the class of finite-state languages under set-theoretic operations was already noticed by Kleene [100]. Myhill [110] and Nerode [111] characterized finite-state languages in terms of interchangeability. The problem of the decidability of properties of finite automata is formulated in the paper of Rabin and Scott [114], which also considers analogous problems for various generalizations of finite automata (see Problem XVIII, etc.). The same paper introduces sources, calling them "nondeterministic automata," and proves a theorem on the determinization of sources which implies various corollaries on the closure of the class of finite-state languages under many operations. Essentially, though not explicitly, this theorem was established previously by Medvedev [48]. Theorem 1.10 is due to Lupanov [42], but an analogous result was proved independently by Ershov [29] (see also Korpelevich [34]). Related problems concerning the determinization of automaton sources (one-letter input alphabet) were studied in detail by Lyubich [43, 44] (see Problem XII). The class of finite-state ω-languages has been investigated by Muller [109] and mainly by McNaughton [102]. To the latter are due Theorems 1.11 (Concatenation Theorem) and 1.12 (Strong Iteration Theorem), which are fundamental for the theory of finite-state ω-languages. The proofs presented in Sections I.9 and I.10 are almost identical with those given by McNaughton in [102] (though they were reconstructed by the present authors on the basis of his announcement).

The idea of the probabilistic automaton is already clear in numerous papers; the analogy between automata and Markov chains has been pointed out by many authors. However, the first systematic investigation of probabilistic automata is by Rabin [113], from whose paper we have taken the substance of Section I.11 (see also Problems XV to XVII). Probabilistic automata are also studied in [23, 81].

Grammars and their relation to automata are studied in the papers of Bar-Hillel, Perles, Shamir [70] and Chomsky [83] (see also [5]).

Operations on languages, involving concatenation, were defined and studied by Kleene [100]. Further investigations were carried out by McNaughton [102], Yanov [65–67], Red'ko [51], Bondarchuk [20, 21], Salomaa [119, 120], Brzozowski [72] and others. Yanov and Red'ko have investigated the existence of a complete system of identities for the operations $\cup$, $\cdot$, $*$.

CHAPTER II

BEHAVIOR OF AUTOMATA WITH OUTPUT

II.1. Anticipation

The theory of automata studies how certain "input" information arriving at discrete times $t = 1, 2, 3, \ldots$ is processed to yield "output" information, also related to times $t = 1, 2, 3, \ldots$ The idea of this process is made rigorous by the concept of an operator transforming input words (ω-words) over a certain alphabet, the input alphabet, into output words (ω-words) over an output alphabet.

Consider fixed input and output alphabets X and Y. The operators we are studying may be classified according to two criteria:

1) Are the argument and the value of the operator words (word operator) or ω-words (ω-word operator)?

2) Is the operator defined over the entire set of input words or ω-words (everywhere defined operator) or only over a certain subset (partial operator)?

Wherever the context makes it clear to which class the operator belongs, or whenever this is immaterial, we shall use the unmodified term "operators."

In the sequel we shall deal mainly with everywhere defined operators; as a rule, partial operators will not be discussed, except for operators whose domain of definition (therefore also range) consists of finitely many words.

Let the operator T transform the word

$$x = x(1)x(2)\ldots x(t)\ldots x(\mu)$$

(ω-word $x(1)x(2)\ldots x(t)\ldots$) into the word

$$Tx = y = y(1)y(2)\ldots y(t)\ldots y(\mu)$$

(ω-word $y = y(1)y(2)\ldots y(t)\ldots$). The argument t usually represents the time at which the letter $x(t)$ is applied at the input or the letter $y(t)$ appears at the output. It is thus important to ascertain to what degree $y(t)$ depends on the "future," i.e., on $x(t+1), x(t+2), \ldots$ and on the "past," $x(1), \ldots, x(t)$.

We intend to make this dependence rigorous in terms of anticipation, and, in the next section, via the concept of weight (memory).

T is said to be a *nonanticipatory* [or causal] *ω-word operator* if, for any ω-words

$$x' = x'(1)x'(2)\dots x'(t)\dots,$$

$$x'' = x''(1)x''(2)\dots x''(t)\dots$$

in the domain of T, and the corresponding outputs

$$Tx' = y' = y'(1)y'(2)\dots y'(t)\dots,$$

$$Tx'' = y'' = y''(1)y''(2)\dots y''(t)\dots,$$

whenever $x'(1) = x''(1),\dots, x'(\tau) = x''(\tau)$ for some natural number τ, then also $y'(1) = y''(1),\dots, y'(\tau) = y''(\tau)$. In other words, for any t, the output letter $y(t)$ is uniquely determined by the input word $x(1)\dots x(t)$ and is independent of the "future" $x(t+1)x(t+2)\dots$ Otherwise, T is said to be an *anticipatory operator*. T is said to be a *nonanticipatory word operator* if the following conditions are satisfied:

1) If $Tx = y$, then x and y are words of the same length.

2) If x' and x'' are input words with identical initial segments of length τ, i.e., $x'(1) = x''(1),\dots, x'(\tau) = x''(\tau)$, then the corresponding output words y' and y'' also have identical initial segments of length τ.

3) (Completeness condition) If T is defined on a word x, it is defined on all its initial segments.

REMARKS. I. The second condition is analogous to that defining nonanticipatory ω-word operators. Any partial operator τ satisfying only conditions 1) and 2), but not condition 3), can always be extended to a (unique) nonanticipatory operator, by defining it as follows for all initial segments of words in its domain. If $Tx = y$ and x' is an initial segment of x of length τ, define Tx' as the corresponding initial segment of the word y. Henceforth, when considering partial word operators satisfying conditions 1) and 2), we shall call them nonanticipatory operators (whenever no misunderstandings can arise), having the above extension in mind.

II. It is clear from the definitions that any (everywhere defined) nonanticipatory ω-word operator T induces a unique (everywhere defined) nonanticipatory word operator T', which maps the initial segments of the input ω-words onto the corresponding initial segments of the corresponding output ω-words. Conversely, given an everywhere defined nonanticipatory word operator T', one can always construct a unique nonanticipatory

ω-word T which induces T' in the above sense. Because of this one-to-one correspondence, in our subsequent discussions of everywhere defined nonanticipatory operators we shall not specify whether these are word or ω-word operators.

We now present some examples of operators. With the exception of T_8 and T_9, they will all be everywhere defined. Nevertheless, $T_1, \ldots, T_7$ may all be associated with partial operators by imposing suitable restrictions on their domains. $T_1, \ldots, T_7$ will be defined as ω-word operators; those that prove to be nonanticipatory may be regarded as word operators as well.

EXAMPLES. Let $X = \{0, 1\}$, $Y = \{0, 1\}$. First consider the operators T_1, T_2, T_3, T_4, T_5 defined as follows:

1) T_1: $y(t) = x(2t)$.

2) T_2:

$$y(t) = \begin{cases} 1, \text{ if the word } x(1)x(2)\ldots x(t) \text{ contains more ones than zeros;} \\ 0 \text{ otherwise.} \end{cases}$$

3) T_3:

$$y(t) = \begin{cases} 1, \text{ if } \exists\tau(\tau > t \,\&\, x(\tau) = 1); \\ 0 \text{ otherwise.} \end{cases}$$

4) T_4:

$$y(t) = \begin{cases} 1, \text{if the word } x(1)x(2)\ldots x(t)\ldots x(t+5) \text{ contains more ones than zeros;} \\ 0 \text{ otherwise.} \end{cases}$$

5) T_5:

$$y(t) = \begin{cases} 1, \text{ if the number of ones in the word } x(1)x(2)\ldots x(t) \\ \text{is a multiple of three;} \\ 0 \text{ otherwise} \end{cases}$$

(similarly, one can define for any k: $y(t) = 1$ if and only if the number of ones in the word $x(1)x(2)\ldots x(t)$ is a multiple of k).

In the following operators T_6 and T_7, the output alphabet will be $Y = \{0, 1\}$, as before, but the input alphabet will be the cartesian product of $\{0, 1\}$ with itself; thus X consists of four letters, which may be written in column notation: $X = \left\{ \begin{matrix} 0 \\ 0 \end{matrix}; \begin{matrix} 0 \\ 1 \end{matrix}; \begin{matrix} 1 \\ 0 \end{matrix}; \begin{matrix} 1 \\ 1 \end{matrix} \right\}$.

Any input ω-word $x = x(1)x(2)\dots x(t)\dots$ defines a pair of ω-words $x' = x'(1)x'(2)\dots x'(t)\dots$ and $x'' = x''(1)x''(2)\dots x''(t)\dots$, the projections of x onto the alphabet $\{0, 1\}$. For any μ ($\mu = 1, 2, 3, \dots$), the initial segments of these ω-word of length μ, when written "from right to left,"

$$x'(\mu)\dots x'(2)x'(1),$$

$$x''(\mu)\dots x''(2)x''(1),$$

represent two nonnegative integers in binary expansion; we denote these integers by x'_μ and x''_μ, respectively.

6) T_6(serial addition): T_6x is defined as the ω-word $y = y(1)\dots y(\mu)\dots$, where $y(\mu)$ is the μ-th digit in the binary expansion of the sum $x'_\mu + x''_\mu$.

7) T_7(serial multiplication): T_7x is defined as the ω-word $y = y(1)\dots y(\mu)\dots$, where $y(\mu)$ is the μ-th digit in the binary expansion of the product $x'_\mu \cdot x''_\mu$.

REMARK. The motive for the above designation of T_6 and T_7 is as follows.

Suppose that a natural number ξ with the binary expansion $\xi_\nu \dots \xi_2\xi_1$ (i.e., $\xi = \sum_{i=1} \xi_i 2^{i-1}$) is encoded as an ω-word $\xi_1\xi_2 \dots \xi_\nu 00\dots 0\dots$ (ξ_ν is followed by zeros). A pair of natural numbers ξ, η is encoded as an ω-word x over the cartesian product of the alphabet $\{0, 1\}$ with itself. Thus, for example, the decimal numbers 7, 12 are encoded as

$$1\ 1\ 1\ 0\ 0\ 0\ 0\dots$$

$$0\ 0\ 1\ 1\ 0\ 0\ 0\dots$$

over the four-letter alphabet $\left\{\begin{matrix}0 & 0 & 1 & 1\\ 0 & 1 & 0 & 1\end{matrix}\right\}$. Then the operators T_6 and T_7 applied to the ω-word representing the pair of numbers ξ, η transform it into the ω-word codes of their sum and product, respectively.

8) T_8 is defined for two words by the table

x	T_8x
011	000
1001	0011

9) T_9 is defined over all eight words of length 3 over the alphabet $\{0, 1\}$ by the table

x	000	001	010	011	100	101	110	111
T_9x	000	000	010	011	000	001	000	001

It is easy to see that $T_2, T_5, T_6, T_7, T_8, T_9$ (T_8, T_9 may be extended in a natural manner to the initial segments of the input words appearing in the tables) are nonanticipatory operators. T_1, T_3, T_4 are anticipatory operators; they clearly depend on the "future." T_1 and T_4 have the following property:

(1) There exists a function $\phi(t)$ such that $y(t)$ is uniquely determined by the word $x(1)x(2)\dots x(\phi(t))$.

Operators possessing property (1) will be called *finitely anticipatory operators* (in contradistinction to infinitely anticipatory operators). In particular, if $\phi(t) = t + c$, where c is a constant, we shall say that T has *bounded anticipation.* Clearly, if $c = 0$ the operator T is nonanticipatory.

Special cases of nonanticipatory operators are constant and truth-table operators.

An operator T is said to be *constant* if it maps every input ω-word x onto a fixed output ω-word y.

We recall the definition of a truth-table operator ("literal translation" operator).

T is said to be a *truth-table operator* if there exists a mapping λ of X into Y such that $y(t) = \lambda(x(t))$ $(t = 1, 2, 3, \dots)$.

Recall that, given any initialized automaton $\langle \mathfrak{M}, q_0 \rangle$, where $\mathfrak{M} = \langle Q, X, Y, \Psi, \Phi \rangle$, we have identified its behavior with the operator $T(\mathfrak{M}, q_0)$ that it realizes, i.e., the operator defined by the recurrence relations

$$q(1) = q_0,$$

$$q(t+1) = \Psi[q(t), x(t)],$$

$$y(t) = \Phi[q(t), x(t)].$$

Now the behavior of any initialized automaton is obviously a nonanticipatory operator; this follows directly from the recursive definition of $y(t)$ $(t = 1, 2, 3, \dots)$. Conversely, every nonanticipatory operator T with input alphabet $X = \{x_1, x_2, \dots, x_m\}$ and output alphabet Y is the behavior of a suitable initialized automaton $\langle \mathfrak{M}, q_0 \rangle$. $\langle \mathfrak{M}, q_0 \rangle$ may always be constructed as the initialized automaton whose diagram is an infinite m-branching tree with a root; the root is labeled by the initial state q_0, and the m edges issuing from each vertex by the input letters $x_1, x_2, \dots, x_m$. The output labels of the tree obey the following rule. Suppose that an edge γ on the level t terminates a path from the root which carries the input word $x(1)x(2)\dots x(t)$; if T transforms this word into $y(1)y(2)\dots y(t)$, label the edge γ with the output letter $y(t)$.

Figure 14*a* illustrates the finite tree v consisting of the three lowest levels of the tree for the operator T_2.

Now if T is a *partial* nonanticipatory operator, analogous reasoning will show how to represent it as a "partial tree"; in a partial tree, certain paths may break off at certain vertices—the number of edges issuing from a vertex may now be less than m (the number of input letters), even zero. Figures 14*b* and 14*c* illustrate partial trees for the operators T_8 and T_9 or, more precisely, for their natural extensions to the initial subwords. Any partial tree may be extended to a complete infinite tree, indeed in many ways. Thus any nonanticipatory partial operator may be extended (generally in many ways) to a nonanticipatory operator defined everywhere. For example, one might introduce the missing edges and label them with the same input letter (see the dotted edges in Figure 14*b*, all labeled 0). We shall say that an *initialized automaton* $\langle \mathfrak{M}, q_0 \rangle$ *realizes a partial operator* T_0 if $T(\mathfrak{M}, q_0)$ is an extension of T_0. The preceding remarks may be summarized in the following theorem.

THEOREM 2.1. *The behavior of any initialized automaton is an everywhere defined nonanticipatory operator. Any nonanticipatory operator (everywhere defined or partial) is realizable by a suitable initialized automaton.*

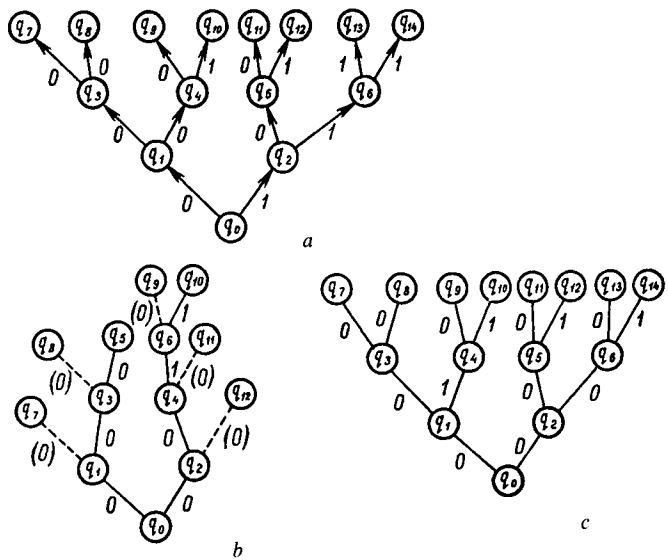


Figure 14

We might mention that some authors call nonanticipatory operators *automaton operators* or *deterministic operators.*

II.2. Memory (weight)

It is clear from Theorem 2.1 that investigation of nonanticipatory operators ("automaton operators") has significance for automata theory. Nevertheless, many concepts and facts revealed by an investigation of this kind carry over (or have at least partial analogues) to anticipatory operators. For this reason, wherever this does not affect the simplicity of the exposition, we shall so phrase the discussion as to make it applicable to the general case.

Our next task is to introduce concepts which describe how operators "store" or "remember" information.

Let us call two word (ω-word) operators $T^{(1)}$ and $T^{(2)}$ *distinguishable* if there exists an input word (ω-word) which they transform into different output words (ω-words); otherwise we call them *indistinguishable.* If $T^{(1)}$ and $T^{(2)}$ are indistinguishable *everywhere defined* operators, they must coincide; in the case of *partial* operators, indistinguishable operators may differ in their domains of definition. We shall denote the indistinguishability relation by $T^{(1)} \approx T^{(2)}$ (read: $T^{(1)}$ is indistinguishable from $T^{(2)}$, $T^{(2)}$ is indistinguishable from $T^{(1)}$), and the distinguishability relation by $T^{(1)} \not\approx T^{(2)}$.

Assume given a family of operators $\tau = \{T^{(1)}, T^{(2)}, \ldots\}$, where the $T^{(i)}$ are either all word operators or all ω-word operators.

We define the *weight of the system* τ (denoted by $\mu(\tau)$) to be the maximal number (possibly ∞) of pairwise distinguishable operators in τ. A *basis* of τ is any subset of pairwise distinguishable operators such that any other operator in τ is indistinguishable from some member of this subset. If τ contains only everywhere defined operators, indistinguishability is an equivalence relation partitioning τ into indistinguishability classes. It is obvious that then $\mu(\tau)$ is precisely the index of this partition, and each indistinguishability class is a set of copies of the same operator. Any set of representatives, one from each indistinguishability class, will be a basis. However, in general indistinguishability is not an equivalence relation, since it may not be transitive. For example, let $T^{(1)}$ and $T^{(2)}$ be distinguishable everywhere defined operators and $T^{(3)}$ an operator defined only for words x such that $T^{(1)}x = T^{(2)}x$, in which case $T^{(3)}x = T^{(1)}x = T^{(2)}x$; then $T^{(1)} \approx T^{(3)}$, $T^{(2)} \approx T^{(3)}$, $T^{(1)} \not\approx T^{(2)}$. Thus, in the general case the concepts

of weight and basis are not related to a partition into indistinguishability classes.

Before going on to the next definitions, we stipulate that henceforth all *word* operators will be assumed to satisfy condition 1) in the definition of nonanticipatory operators: the words x and Tx are equal in length. This restriction is in principle unnecessary, and is adopted only for convenience.

Now, let T be an arbitrary word operator satisfying this condition, or an arbitrary ω-word operator. Let p be an arbitrary word over the input alphabet such that the domain of T contains at least one word (ω-word) of the form px (i.e., having an initial segment p). We now associate an operator T_p with each such word p (whose length we denote by ν). To define the word y into which T_p transforms a word (ω-word) x, form the concatenation px and apply T; now drop the first ν letters from the resulting word (ω-word). Thus, if x is a word of length λ or an ω-word,

$$T_p x = T(px)_\nu^{\nu+\lambda} \qquad \text{or} \qquad T_p x = (T(px))_\nu^\infty.$$

Any operator T_p defined in this way will be called the *residual operator of T relative to the input word p.* The operator T itself is a residual operator relative to the empty word. For formal reasons it is convenient to consider the "nowhere defined" operator, which is indistinguishable from any operator, by definition. If T is applicable to a word p, but to no longer word with initial segment p, we shall consider p to correspond to the nowhere defined operator. Call two words p_1 and p_2 *residually indistinguishable by T* (notation: $p_1 \approx p_2(T)$) if the corresponding residual operators are indistinguishable. The *weight* and *basis of an operator T* are defined to be the weight and basis, respectively, of the system τ of all its residual operators. We shall also say that a set of input words forms a basis if the corresponding family of residual operators is a basis.

We now illustrate these concepts for the operators $T_1, \ldots, T_9$ of Section II.1.

For the operator T_3, all input words (including the empty word!) are residually indistinguishable; the associated residual operators are indistinguishable from the operator T_3. This mathematically represents the fact that the value of $y(t)$ is not affected by the "past" input $x(1)x(2)\ldots x(t)$. In this sense, the operator T_3 "remembers" nothing. Its weight is 1 and the empty word alone is a basis.

At first sight, one might think (wrongly!) that the same holds for the

operator T_1, since $y(t)$ depends on a "later" input letter $x(2t)$. But this is not so; two input words p_1 and p_2 are residually indistinguishable by T_1 if and only if they are equal in length. Thus, T_1 "stores" the length of the input word. Its weight is ∞; for a basis one can take the set $\{\wedge, 0, 00, 000, \ldots\}$.

Let $n_0(p)$ denote the number of zeros in the word p, $n_1(p)$ the number of ones in p. The nonanticipatory operator T_2 and the operator T_4 with bounded anticipation display the same behavior as regards residual indistinguishability; each has weight ∞. For p_1 and p_2 are residually indistinguishable if and only if

$$n_1(p_1) - n_0(p_1) = n_1(p_2) - n_0(p_2).$$

Thus each of the operators T_2, T_4 "stores" the excess of the number of ones over the number of zeros in the input word.

A basis for T_2 or T_4 is the set

$$\{\wedge, 0, 1, 00, 11, 000, 111, 0000, 1111, \ldots\}.$$

The operator T_5 "remembers" which of the following three relations holds for the input word p:

$$n_1(p) = 0 (\mathrm{mod}\ 3), \qquad n_1(p) = 1 (\mathrm{mod}\ 3), \qquad n_1(p) = 2 (\mathrm{mod}\ 3).$$

Its weight is 3, and a possible basis is $\{\wedge, 1, 11\}$.

The operator T_6 "remembers" whether, in adding the numbers x'_μ, x''_μ corresponding to the input word $x(1)\ldots x(\mu)$, there is a carry operation in the highest-order ($(\mu + 1)$-th) digit or not. Its weight is 2 and a basis is $\left\{\wedge; \begin{matrix} 1 \\ 1 \end{matrix}\right\}$.

As for the operator T_7, it can be shown that all input words are pairwise indistinguishable—we leave the proof to the reader. Thus, the weight of T_7 is ∞; the only basis is the set of all input words.

The operators T_8 and T_9 have weights 2 and 3, respectively. Possible bases are $\{\wedge, 1\}$ and $\{\wedge, 0, 1\}$, respectively.

It is clear from these examples that the weight of an operator is, in a certain sense, a quantitative measure of its "internal" memory. For example, the value of this parameter figures in the following simple theorem.

THEOREM 2.2. *An ω-word operator T of weight k transforms any periodic ω-word x (in its domain) with period η into an (ultimately) periodic ω-word y with period $\eta' \leqq k \cdot \eta$.*

Proof. Let $x = ppp\ldots$, where p is a word of length η. Let p^0 denote the empty word, p^s the word $pp\ldots p$ (s times). The sequence $p^0, p^1, p^2, \ldots, p^k$ must contain two residually indistinguishable words, say p^μ and p^ν, $\nu > \mu$. The ω-word $y = Tx$ has the form $s_1 s_2 s_3 \ldots$, where each s_i is of length η. Since p^μ and p^ν are indistinguishable, the ω-words $s_{\mu+1}s_{\mu+2}\ldots$ and $s_{\nu+1}s_{\nu+2}\ldots$ are equal. Let r denote the word $s_{\mu+1}s_{\mu+2}\ldots s_\nu$; then the ω-word y has the form $s_1 s_2 \ldots s_\mu rrr\ldots$, and the length of the word r, which is $(\nu - \mu)\eta$, is at most $k\eta$. Q.E.D.

The ω-word $y = Tx$ has the form $s_1 s_2 s_3 \ldots$, where each s_i is of length η.

Let T be a constant operator generating the ω-word $sppp\ldots$; it is easy to see that the weight of the operator T is at most the sum of the lengths of s (the phase) and p (the period). Combined with Theorem 2.2, this directly implies

COROLLARY. *A constant operator has finite weight if and only if its output ω-word is periodic.*

In the previous section we saw how a nonanticipatory operator T is defined by a (possibly partial) tree. Each input word in the domain of T is associated with a well-defined path from the root of this tree v and a well-defined vertex at which this path ends. Now the branch of the tree v issuing from this vertex is itself a tree; it defines the corresponding residual operator. This interpretation enables us to carry over all terms and notation introduced above for operators, input words and residual operators (weight, distinguishability, basis) to the tree v, its vertices and branches. For example, the vertices q_2 and q_6 in the finite tree for the operator T_8 (Figure 14*b*) are indistinguishable, while the vertices q_0, q_4 form a basis; the tree has weight 2. In the finite tree of Figure 14*c*, which has weight 3, the vertices q_0, q_1, q_2 form a basis; the vertex q_3 is indistinguishable from both q_0 and q_2, and the latter are distinguishable.

The final vertices are indistinguishable from any other vertices, since they are roots of empty branches, describing nowhere defined operators.

II.3. Equivalent automata

We shall call two initialized automata $\langle \mathfrak{M}', q_0' \rangle$ and $\langle \mathfrak{M}'', q_0'' \rangle$ *equivalent* if they realize the same operator: $T(\mathfrak{M}', q_0') = T(\mathfrak{M}'', q_0'')$. Two (noninitialized) automata $\mathfrak{M}'$ and $\mathfrak{M}''$ are said to be equivalent if any operator realizable in one of them (by a suitable initialization) is realizable in the other (suitably initialized). Obviously, isomorphic initialized (noninitialized)

automata are equivalent; it is easily seen that the converse is in general false. In studying the behavior of automata it is sometimes necessary to replace an automaton by an equivalent automaton which is more manageable in the situation in question. This section presents a few simple relevant concepts and facts.

Call two states q_i, q_j of an automaton $\mathfrak{M}$ *indistinguishable* (notation: $q_i \approx q_j(\mathfrak{M})$) if the operators $T(\mathfrak{M}, q_i)$ and $T(\mathfrak{M}, q_j)$ are indistinguishable; otherwise q_i and q_j are *distinguishable*. Indistinguishability is an equivalence relation which partitions the state set Q of the automaton into indistinguishability classes; the number of these classes (possibly ∞) is termed the *reduced weight of the automaton* $\mathfrak{M}$. The unqualified term *weight of an automaton* will be used as a synonym for the number of its states.

A (noninitialized) automaton $\mathfrak{M}$ is said to be *reduced* if all its states are pairwise distinguishable. For a finite automaton $\mathfrak{M}$, this is clearly true if and only if the weight of the automaton is equal to its reduced weight. An *initialized automaton* $\langle \mathfrak{M}, q_0 \rangle$ is said to be *reduced* if:

a) the (noninitialized) automaton $\mathfrak{M}$ is reduced;

b) all states of $\mathfrak{M}$ are accessible from q_0 (i.e., for every state q there is an input word taking q_0 to q).

Note that a tree always satisfies the second condition, but not, in general, the first.

Let $\mathfrak{M}$, $\mathfrak{M}'$ be two reduced equivalent automata. It is easy to see that the mapping ϕ which takes each state q of $\mathfrak{M}$ to the state q' of $\mathfrak{M}'$ such that $T(\mathfrak{M}', q') = T(\mathfrak{M}, q)$ is an isomorphism. The situation is analogous for $T(\mathfrak{M}', q') = T(\mathfrak{M}, q)$ is an isomorphism. The situation is analogous for initialized automata. Consequently, *reduced initialized (noninitialized) automata are isomorphic if and only if they are equivalent.*

The following proposition also holds:

For any automaton $\mathfrak{M} = \langle Q, X, Y, \Psi, \Phi \rangle$, *there exists an equivalent reduced automaton* $\mathfrak{M}' = \langle \Pi, X, Y, \Psi', \Phi' \rangle$.

To convert $\mathfrak{M}$ into $\mathfrak{M}'$, *indistinguishable states must be merged.* In other words, to each class K_i of indistinguishable states in $\mathfrak{M}$ corresponds one state of $\mathfrak{M}'$, which we denote by π_i. Let q_i and q_j be arbitrary representatives of the classes K_i, K_j such that $\Psi(q_i, a) = q_j$, $\Phi(q_i, a) = b$. Then we define $\Psi'(\pi_i, a) = \pi_j$, $\Phi'(\pi_i, a) = b$.

It is easy to see that

1) the definition is independent of the choice of representatives;

2) the automaton $\mathfrak{M}'$ obtained by this "factorization" procedure is equivalent to $\mathfrak{M}$ and is reduced.

Now let $\langle \mathfrak{M}, q_0 \rangle$ be a given initialized automaton. If $\mathfrak{M}$ has states inaccessible from q_0, consider the subautomaton $\mathfrak{M}' = \langle Q', X, Y, \Psi, \Phi \rangle$, where Q' is the subset of Q consisting of q_0 and all states in Q accessible from q_0. It is clear that $\langle \mathfrak{M}', q_0 \rangle$ and $\langle \mathfrak{M}, q_0 \rangle$ are equivalent. By carrying out any required merging of indistinguishable states in $\mathfrak{M}'$, we finally get a reduced initialized automaton equivalent to $\langle \mathfrak{M}, q_0 \rangle$.

It is easily seen that this procedure is effective whenever the automaton $\mathfrak{M}$ is finite. For let k denote the number of its states. First, inaccessible states may be found by inspection of all simple paths of length at most k issuing from the initial vertex of the diagram of $\mathfrak{M}$. Second, for any pair of states q and q' one can effectively determine whether they are distinguishable. This ensues from the following easily verified proposition:

A. *Let q and q' be two distinguishable states of an automaton $\mathfrak{M}$. Then there exists an input word of length at most k^2 for which the initialized automata $\langle \mathfrak{M}, q \rangle$ and $\langle \mathfrak{M}, q' \rangle$ generate different output words.*

Thus, in order to determine whether q and q' are distinguishable it suffices to check the outputs of the automata $\langle \mathfrak{M}, q \rangle$ and $\langle \mathfrak{M}, q' \rangle$ for input words of length at most k^2.

Let us prove Proposition A.

Suppose that a word $x = x(1)x(2)\dots x(t)$ is transformed by the automata $\langle \mathfrak{M}, q \rangle$ and $\langle \mathfrak{M}, q' \rangle$ into different words

$$y = y(1)\dots y(t) \qquad \text{and} \qquad y' = y'(1)\dots y'(t);$$

in the process, the automata $\langle \mathfrak{M}, q \rangle$ and $\langle \mathfrak{M}, q' \rangle$ go through sequences of states $q(1) = q,\ q(2), \dots, q(t+1)$ and $q'(1) = q',\ q'(2), \dots, q'(t+1)$, respectively. We may assume that the last letters $y(t)$ and $y'(t)$ are distinct; otherwise, for some τ, $\tau < t$, we would have $y(\tau) \neq y'(\tau)$, but then there would be a shorter word $x(1)\dots x(\tau)$ for which the last output letters are distinct. Assume that, for some τ_1 and τ_2 $(\tau_1 < \tau_2 \leqq t)$ we have $q(\tau_1) = q(\tau_2)$ and $q'(\tau_1) = q'(\tau_2)$; then, by dropping the subword $x(\tau_1)\dots x(\tau_2 - 1)$ from the word $x(1)\dots x(\tau_1)\dots x(\tau_2)\dots x(t)$ we obtain a shorter word $x(1)\dots x(\tau_1 - 1)x(\tau_2)\dots x(t)$ which is transformed into different words

$$y(1)\dots y(\tau_1 - 1)y(\tau_2)\dots y(t)$$

and

$$y'(1)\dots y'(\tau_1 - 1)y'(\tau_2)\dots y'(t).$$

Thus, we may assume that for any $\tau_1, \tau_2 (\tau_1 < \tau_2 \leqq t)$ the pair $\langle q(\tau_1), q'(\tau_1) \rangle$

is distinct from the pair $\langle q(\tau_2), q'(\tau_2)\rangle$. Since the number of all different pairs of states is k_2, the length of the word x cannot exceed k^2.

THEOREM 2.3. *For any initialized (noninitialized) automaton, there exists a unique (up to isomorphism) equivalent [reduced] automaton. There is an algorithm which, given any finite automaton (initialized or noninitialized), constructs the equivalent reduced automaton.*

If the reduced automaton is finite, the number of its states is strictly smaller than the number of states in any other equivalent (nonisomorphic) automaton. No analogous comparison of cardinalities is valid for infinite reduced automata; nevertheless, it is clear that the reduced automaton is more "economical" than other equivalent (nonisomorphic) automata in a natural sense. For this reason, the procedure whereby an equivalent reduced automaton is derived from a given automaton is known as *minimization*. The above minimization algorithm is quite cumbersome, mainly because one must inspect all words of length k^2 to determine whether states are distinguishable. We shall see later (Section II.13) that in effect the number of words inspected can be substantially reduced. Moreover, there are much simpler algorithms for smaller classes of automata. Special minimization procedures lie beyond the scope of this book; some examples will be given in Sections II.5 and II.13.

Though in practice one is always interested in minimizing automata, in the theoretical context it is sometimes convenient to use equivalent automata having perhaps more states than the original automata but possessing certain other useful properties. In such cases, the merging operation is replaced by what is, in a certain sense, its inverse: *splitting of states*. This is the case when one replaces an arbitrary automaton by an equivalent Moore automaton (for the definition see Section 0.2).

THEOREM 2.4. *For any automaton $\mathfrak{M}$, there is an equivalent Moore automaton $\mathfrak{M}'$.*

Proof. Let $\mathfrak{M}$ have states $q_1, q_2, \ldots, q_k$ and output letters $y_1, y_2, \ldots, y_n$. For each pair $\langle q_i, y_j\rangle$, define q_{ij} to be a state of $\mathfrak{M}'$. The functions Ψ' and Φ' of $\mathfrak{M}'$ are defined in terms of Ψ and Φ as follows.

Let

$$\Psi(q_i, x) = q_s, \qquad \Phi(q_i, x) = y_r.$$

Then, for any j,

$$\Psi'(q_{ij}, x) = q_{sr}, \qquad \Phi'(q_{ij}, x) = y_r.$$

All edges in the diagram of $\mathfrak{M}'$ which are incident on a vertex q_{sr} are assigned the output label y_r, irrespective of the other index of the state. It is easily seen that, for any i,

$$T(\mathfrak{M}, q_i) = T(\mathfrak{M}', q_{i1}) = T(\mathfrak{M}', q_{i2}) = \ldots = T(\mathfrak{M}', q_{in}),$$

and so $\mathfrak{M}$ and $\mathfrak{M}'$ are equivalent.

REMARK. If $\mathfrak{M}$ is a finite automaton with k states and n output letters, the above procedure effectively constructs an automaton $\mathfrak{M}'$ with nk states.

Apart from the usual concepts of equivalence, indistinguishability, etc., certain analogues of these concepts are convenient for Moore automata. The definitions follow.

Two initialized Moore automata $\langle \mathfrak{M}, q_0 \rangle$ and $\langle \mathfrak{M}', q_0' \rangle$ are said to be *Moore-equivalent* if they are equivalent in the usual sense and, in addition, $\lambda(q_0) = \lambda'(q_0')$ (where λ, λ' are the shifted output functions). States q_i, q_j of a Moore automaton $\mathfrak{M}$ are *Moore-indistinguishable* if the initialized automata $\langle \mathfrak{M}, q_i \rangle$ and $\langle \mathfrak{M}, q_j \rangle$ are Moore-equivalent. Finally, a Moore automaton $\mathfrak{M}$ is *Moore-reduced* if all its states are pairwise Moore-distinguishable (for an initialized automaton, add the usual requirement that all states be accessible from the initial state). Following the arguments which have lead to Theorem 2.3, we easily verify that they carry over in a natural manner to Moore automata, provided we prefix the qualification "Moore" to the terms "equivalent," "indistinguishable," "reduced." We thus have the following analogue of Theorem 2.3.

THEOREM 2.3′. *For any initialized (noninitialized) Moore automaton, there exists a unique (up to isomorphism) Moore-equivalent reduced automaton. There is an algorithm (minimization algorithm) which, given any finite Moore automaton, constructs an equivalent reduced Moore automaton.*

II.4. Comparison of the weight of an operator with the weight of an automaton realizing it

Let T be a nonanticipatory (everywhere defined or partial) operator, and $\langle \mathfrak{M}, q_0 \rangle$ an initialized automaton realizing T. We wish to examine in greater detail the connection between the weight of T and the number of states of $\mathfrak{M}$. If the words p_1, p_2 are residually distinguishable by the operator T, then, as is easily seen, they take the initial state q_0 to different (even

distinguishable) states. Consequently, the weight of the automaton $\mathfrak{M}$ (even its reduced weight) cannot exceed the weight of T. Is any operator T of weight μ realizable in a suitable automaton of the same weight? That the answer to this question is positive for everywhere defined operators follows directly from the fact that the minimization procedure described above is applicable to the tree of the operator T. Since all vertices (states) of a tree are accessible from its root (initial state), it follows that the reduced automaton may be constructed by simply merging indistinguishable states (vertices) of the tree. Recall that the vertices (states) of a tree are indistinguishable if and only if the corresponding words are residually indistinguishable with respect to T. Thus, in the reduced automaton realizing T there will be exactly as many states as there are residual indistinguishability classes for T.

Thus, we have two ways of associating an initialized automaton with an everywhere defined operator: trees and reduced automata. In a certain sense, these represent two extremes. While in trees new states are introduced "generously," wherever possible, the new states in a reduced initialized automaton are introduced "economically," only when absolutely necessary. The above procedure for converting a tree into an equivalent reduced automaton is called *contraction of an infinite tree.* If T is an operator with infinite weight, its tree may already be a reduced automaton, and so contraction yields no economy. This is the case for the tree of the operator T_7 (serial multiplication), since any two input words are residually distinguishable by T_7 and so any two states in its tree are distinguishable. In the general case, however (in particular, when T has finite weight), there are many situations intermediate between trees and reduced automata, involving incomplete merging of indistinguishable states.

EXAMPLES. Consider the operators T_2, T_5, T_6; we defined bases for these operators in Section II.2. This makes it possible to associate the states of the corresponding reduced automata with the elements of the bases (since these words are representatives of the indistinguishability classes). In all three cases the initial state is the empty word $\wedge$. Let us define the next-state function Ψ and output function Φ for these operators.

1) T_2. Let q_0 be associated with the empty word $\wedge$; $q_1, q_2, \ldots$ with the words $1, 11, \ldots$; $q_{-1}, q_{-2}, \ldots$ with the words $0, 00, \ldots$ Then $\Psi(q_s, 0) = q_{s-1}$, $\Psi(q_s, 1) = q_{s+1}$. Let $\Psi(q_s, a) = q_j (a = 0, 1)$; then, for $a = 0, 1$, $\Phi(q_s, a)$ depends only on q_j, so that the reduced automaton is a Moore automaton. In fact, if $j > 0$, then $\Phi(q_s, a) = 1$; otherwise $\Phi(q_s, a) = 0$.

2) T_5. Let q_0, q_1, q_2 correspond to the words $\wedge, 1, 11$. Then $\Psi(q_s, 0) =$

$= q_s$; $\Psi(q_s, 1) = q_{s+1 \pmod 3}$. Again we obtain a Moore automaton, with $\Phi(q_s, a) = 1$ if and only if $\Psi(q_s, a) = q_0$.

3) T_6. If q_0 and q_1 denote the states corresponding to the words $\wedge$ and $\begin{matrix}1\\1\end{matrix}$, the required automaton is defined by Table 4 (p. 15).

Hitherto all operators have been everywhere defined. We now proceed to partial operators. It is immediate that if a partial operator T has weight μ, this no longer guarantees realizability in an automaton having μ states. An example is the operator T_8, whose weight is 2. If we extend this operator to an everywhere defined operator $\tilde{T}$ (i.e., extend the finite tree of Figure 14*b* to an infinite tree by suitably labeling the dashed edges), the operator $\tilde{T}$ has weight at least 3. In fact, if the dashed edge issuing from the vertex q_2 is labeled zero, the vertices q_0, q_4, q_6 become pairwise distinguishable; if it is labeled 1 the vertices q_0, q_2, q_4 are pairwise distinguishable.

In the sequel we shall be interested in partial operators satisfying the following condition: There exists a natural number h such that the domain of T is the set of all words of length at most h. These operators are sometimes called *multiple experiments of length h.** They are defined by finite trees of height h which we shall call *complete trees of height h.* (The operator T_9 was first defined only for words of length 3, but it may be regarded as a multiple experiment if we extend the definition to all possible initial segments of the words in its domain. The resulting complete tree of height 3 is illustrated in Figure 14*c*.)

We shall now describe a procedure whereby, given any multiple experiment (or finite complete tree) of weight μ, one can construct a finite automaton with μ states that defines it. This automaton must obviously be reduced. In general, there may be several reduced (nonisomorphic) automata with μ states defining the same multiple experiment of weight μ; when this is so, our procedure will yield all of them. The procedure itself is quite simple, and we shall confine ourselves to a description, omitting the formal justification, which—for all its transparency—is extremely lengthy.

Given a finite complete tree v. Our procedure will convert the tree v into the diagram D of the required automaton (in general, several such diagrams!). Number the vertices $\alpha_0, \alpha_1, \alpha_2, \ldots$ of the tree v in order of increasing

* *Translator's note:* This term is used in a different sense in Western literature (see Chapter IV).

rank; to fix ideas, suppose the vertices within each rank to be ordered in the same way as in the corresponding level (from left to right). This means that the input words represented by these vertices are ordered lexicographically.

Step 1 (*Selection of the vertices of D*). Going through the sequence of vertices $\alpha_0, \alpha_1, \alpha_2, \ldots$ in order, delete every vertex which is indistinguishable from a vertex occurring earlier in the sequence. It is clear that the remaining vertices (one of which must be the root of the tree v) are pairwise distinguishable in v; these are precisely the vertices of D.

Step 2 (*Connection of edges*). Now consider the edges of v (together with their labels) issuing from the vertices selected for D, and the set $\mathfrak{B}$ of all vertices on which these edges are incident. Consider an edge going from a vertex $\alpha \in D$ to a vertex $\beta \in \mathfrak{B}$. If β also belongs to D, the edge is directed to β. But if $\beta \notin D$, the set of vertices selected for D must contain a vertex γ (possibly more than one!) which is indistinguishable from β. Direct the edge to one of these vertices γ.

This completes the description of the procedure, which we shall call *contraction of a finite tree.* The procedure is unambiguous if the following condition holds:

Contraction-uniqueness condition: For every vertex α in $\mathfrak{B}$ there is a unique vertex in D which is indistinguishable from α.

If this condition does not hold, then by varying the connection of edges one obtains a finite family of automata.

It is obvious that the weight of any contraction of a tree cannot exceed the weight of the tree itself, i.e., the weight of the given operator T. On the other hand, it is not hard to show that any automaton obtained in this way indeed realizes the operator T, whence it follows that the weight of the automaton is at least equal to that of T. Consequently, the weight of the automaton is exactly that of T. In other words, selection of the vertices of D in effect constructs a basis of the tree v. Now, once a basis of v has been selected in Step 1, the most natural admissible connections of edges which give the automaton weight μ are precisely those indicated in Step 2. We can now summarize the relation between the weight of a nonanticipatory operator and the weight of an automaton realizing it:

THEOREM 2.5. (*I*) *The weight of any automaton realizing an operator of weight* μ *is at least* μ.

(*II*) *Any everywhere defined operator* T *of weight* μ *is realized by a* (*unique*) *reduced automaton of weight* μ.

(III) Any multiple experiment of weight μ is realized by at least one reduced automaton of weight μ.

(IV) There is an algorithm which, given any multiple experiment of weight μ, constructs all (nonisomorphic) reduced automata of weight μ that realize it.

To conclude this section we illustrate the contraction procedure for the tree of Figure 15*a*. The weight is 5; the basis contains the vertices q_1, q_2, q_3, q_4, q_5. The other vertices of the tree are labeled with the states of the basis from which they are indistinguishable. Figure 15*b* illustrates the corresponding contraction.

II.5. Representation of languages (ω-languages) and realization of operators. The uniformization problem

The behavior of an outputless automaton has been defined in terms of the languages (ω-languages) that it represents, the behavior of an autom-

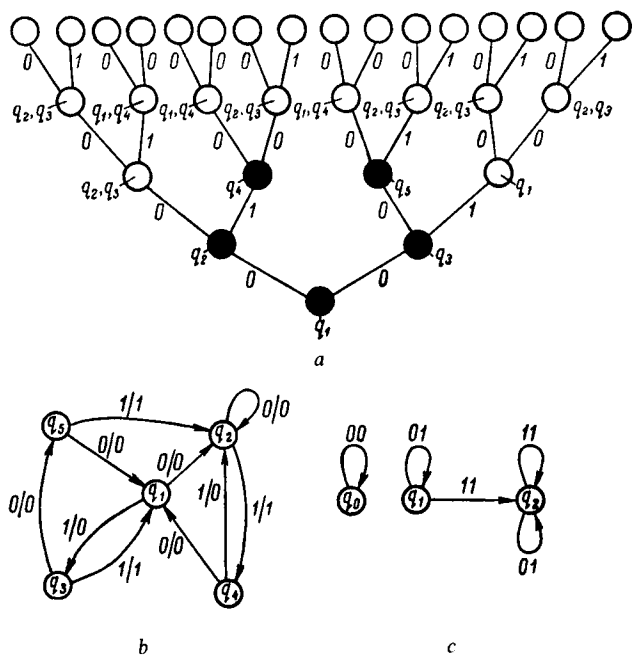


Figure 15

aton with output in terms of the operator that it realizes. There is a natural connection between these concepts of behavior which we shall briefly consider in this section. The connection is based on the routine correspondence between sets and functions: any set (regarded as a subset of some universal set U) is uniquely determined by its characteristic function, while any function f is uniquely determined by a set of "points"—its graph. In our case, the role of sets is played by languages (ω-languages), that of functions by operators.

First note that, in analogy to the situation for anchored outputless automata, one can define representation of languages and ω-languages by nonanticipatory operators (and thereby also by automata with output).

With each subalphabet Y' of the output alphabet Y of a nonanticipatory operator T one can associate a language TY': $x(1)\dots x(t) \in TY'$ if and only if T transforms this word into an output word $y(1)\dots y(t)$ such that $y(t) \in Y'$.

We shall then say that *T represents the language TY' by the set of outputs Y'* (the output letters of Y' are "used" by T to accept words of TY', the output letters of $\neg TY'$ to reject words of $\neg TY'$). For example, the operator T_2 accepts by output 1 those words over the alphabet $\{0, 1\}$ which contain more occurrences of 1 than of 0.

Now, with any system $b = \{Y'\}$ of subalphabets of the output alphabet Y the operator T associates an ω-language Tb, consisting of all ω-words $x = x(1)x(2)\dots x(t)\dots$ which T maps onto ω-words $y = y(1)y(2)\dots y(t)\dots$ such that $\lim y \in b$. We shall say that *T represents the ω-language Tb* (accepts the words of Tb and rejects the words of $\neg Tb$).

These terms, which apply to nonanticipatory operators, may also be defined for initialized automata, via the operators that the latter realize. Thus, we shall say that an automaton $\langle \mathfrak{M}, q_0 \rangle$ represents a language $\mathfrak{A}$ by outputs Y' if the operator $T(\mathfrak{M}, q_0)$ represents the language by outputs Y'.

There is a natural and obvious connection between representation of a language (ω-language) by an anchored outputless automaton and its representation by outputs (sets of outputs) of a Moore automaton. With any outputless automaton $\mathfrak{M} = \langle Q, X, \Psi \rangle$ one can associate a Moore automaton $\mathfrak{M}' = \langle Q, X, Y, \Psi, \Phi \rangle$ by adjoining an output alphabet Y and an output function Φ, as follows: *a*) If $Q = \{q_0, q_1, \dots, q_k\}$, then $Y = \{y_0, y_1, \dots, y_k\}$. Thus each state (set of states) has a corresponding output letter (subset of output letters). *b*) If $\Psi(q_i, x) = q_j$, then $\Phi(q_i, x) = y_j$. Now suppose that $\mathfrak{A} = \omega(\mathfrak{M}, q_0, Q')$ and Y' corresponds to Q'. Then the initialized Moore automaton $\langle \mathfrak{M}', q_0 \rangle$ represents the same language $\mathfrak{A}$

by outputs Y'. In the same way, given an ω-language $\mathfrak{A} = \Omega(\mathfrak{M}, q_0, \mathfrak{C}_0)$, if $\tilde{\mathfrak{C}}$ is the system of subalphabets of Y corresponding to the macrostates of $\mathfrak{C}$, the initialized Moore automaton $\langle \mathfrak{M}', q_0 \rangle$ represents the same ω-language $\mathfrak{A}$ by the system $\tilde{\mathfrak{C}}$.

Now consider a Moore automaton $\mathfrak{N} = \langle Q, X, Y, \Psi, \Phi \rangle$. By disregarding the output alphabet Y and the output function Φ we get an outputless automaton $\tilde{\mathfrak{N}} = \langle Q, X, \Psi \rangle$. Recall that, since $\mathfrak{N}$ is a Moore automaton, $\Phi(q, x) = \lambda[\Psi(q, x)]$, where λ is a mapping of Q into Y. We can thus associate with each output letter $y \in Y$ a subalphabet $\lambda^{-1}y$, the complete preimage of y in Q; this also defines a macrostate for each subalphabet of Y.

Suppose that $\langle \mathfrak{N}, q_0 \rangle$ represents a language $\mathfrak{A}$ by outputs Y'. Let Q' denote the macrostate corresponding to Y'. It is then obvious that this language is represented by the anchored automaton $\langle \tilde{\mathfrak{N}}, q_0, Q' \rangle$. Similarly, if the automaton $\langle \mathfrak{N}, q_0 \rangle$ represents an ω-language $\mathfrak{B}$ by a system of subalphabets $\mathfrak{C}$, then $\mathfrak{B} = \Omega(\tilde{\mathfrak{N}}, q_0, \mathfrak{C}')$, where $\mathfrak{C}'$ corresponds to $\mathfrak{C}$. We have shown (Theorem 2.4) that any automaton with output $\mathfrak{M}$ is equivalent to some Moore automaton $\mathfrak{N}$; moreover, if $\mathfrak{M}$ is finite so is $\mathfrak{N}$, and it is effectively constructible from $\mathfrak{M}$. We summarize these arguments in the following proposition.

THEOREM 2.6. *The class of languages (ω-languages) representable in finite automata with output coincides with the class of languages (ω-languages) representable in finite autputless automata. There is an algorithm which, given any language (ω-language) in one of these representations, constructs the other representation.*

This theorem reveals that the theory of outputless automata presented in Chapter I is in effect the theory of Moore automata. We can therefore return to certain questions touched upon in that chapter, whose solution is treated with greater facility in terms of automata with output. For example, consider the *minimization of an anchored finite automaton:* Given a finite anchored automaton $\langle \mathfrak{M}, q_0, Q' \rangle$, to construct an equivalent anchored automaton with the minimal number of states. This problem reduces in an obvious manner to the minimization problem for the associated Moore automaton. Thus the minimization algorithm for anchored automata is a simple combination of the algorithms from Theorems 2.6 and 2.3′.

The procedure described in Theorem 2.3′ for identification of indistinguishable states is rather unwieldy. This is precisely the reason why, in

certain cases, indistinguishable states are conveniently envisaged in terms of properties of the diagram. We used this device in formulating our rules for simplification of sources (Section I.5). Theorem 2.3′ implies that the rules for merging of absorbing states or equivalent states are valid.

EXAMPLE. *Minimization of the anchored automaton* $\langle R, w_1, w_4 \rangle$ (Figure 13*b*). The Moore automaton associated with $\langle R, w_1, w_4 \rangle$ is $\langle S, w_1 \rangle$ (Figure 13*e*), in which the edges ending at the vertex w_4 are assigned the output label 1 and all other edges the output label 0. The following pairs of indistinguishable states are merged in $\langle S, w_1 \rangle$:

1) w_3 and w_6 (corresponding to the merging of the absorbing vertices w_3, w_6 in the anchored automaton $\langle R, w_1, w_4 \rangle$);

2) w_2 and w_5 (w_2 and w_5 are equivalent);

3) w_1 and w_4 (it is easily seen from the diagram of Figure 13*e* that the operators $T(S, w_1)$ and $T(S, w_4)$ coincide).

The result of these merging operations is a Moore automaton $\langle S', w_4 \rangle$ (Figure 13*f*) capable of three states, any two of which are distinguishable. By erasing the output labels and making w_4 the final state, we get the anchored automaton $\langle R'', w_4, w_4 \rangle$ Figure 13*d*).

Thus the automaton $\langle R, w_1, w_4 \rangle$ of Figure 13*b* has been converted into an equivalent minimal automaton $\langle R'', w_4, w_4 \rangle$ (Figure 13*d*).

If no restrictions are imposed on the nonanticipatory operator T, any language $\mathfrak{A}$ is representable by a suitable operator T with a two-letter output alphabet, such as $Y = \{0, 1\}$. In fact, we need only define the operator T by setting $y(t) = 1$ if and only if $x(1)x(2)\ldots x(t) \in \mathfrak{A}$. Hence the class of all nonanticipatory operators with given alphabets X and Y, where Y contains at least two letters, has the cardinality of the continuum; it follows that the "overwhelming majority" of these operators are ineffective.

Nonetheless, it is clear that any language $\mathfrak{A}$ which is representable in a finite automaton is also representable in a finite automaton $\mathfrak{M}$ with output alphabet consisting of only two letters. We may assume that $\mathfrak{M}$ accepts words of $\mathfrak{A}$ by output 1 and rejects words of $\neg\mathfrak{A}$ by output 0. That this is possible follows from the fact that, in constructing the Moore automaton $\langle \mathfrak{M}', q_0 \rangle$ for an anchored automaton $\langle \mathfrak{M}, q_0, Q' \rangle$, we could have defined the output function $\Phi(q, x)$ by setting $\Phi(q, x) = 1$ if $\Psi(q, x) \in Q'$ and $\Phi(q, x) = 0$ otherwise.

Let T be a nonanticipatory operator with output alphabet $\{0, 1\}$ which represents a language $\mathfrak{A}$ by output 1. It is easy to see that residual indistinguishability of two words p_1, p_2 by T means that, for any nonempty word r,

$p_1 r \in \mathfrak{A}$ if and only if $p_2 r \in \mathfrak{A}$. Now this recalls the definition (Chapter I) of the left interchangeability of two words p_1 and p_2 with respect to a language $\mathfrak{A}$, except for one point: in Chapter I the word r may be empty, so that the words p_1 and p_2 themselves must both either belong or not belong to $\mathfrak{A}$. Thus, left interchangeability with respect to $\mathfrak{A}$ implies residual indistinguishability by the operator T. The following simple example will show that the converse is not necessarily true. Consider the language $\mathfrak{A}$ over the alphabet $\{a, b\}$ consisting of all words ending with the letter b, and a nonanticipatory operator T (which is even a truth-table operator) accepting these words. It is obvious that any two words are residually indistinguishable with respect to T, but the one-letter words a, b are not left interchangeable.

We have thus worked out a complete characterization of representability in terms of operators. We shall now attempt the converse: to characterize operators in terms of representability. Define the graph* of the operator T to be the set of all pairs $\langle x, y \rangle$ such that $y = Tx$. If x, y are ω-words or words of the same length, we can consider the coupling $\langle x(1)y(1) \rangle \cdot \cdot \langle x(2)y(2) \rangle \ldots$ The set of all these ω-words (words) forms an ω-language (language), which we shall also call the graph of the operator. The following assertion is evident:

If the operator T is realizable by a finite automaton, its graph is representable in a finite automaton.

To fix ideas, we shall consider only ω-word operators. Let $T = T(\mathfrak{M}, q_0)$. If we regard the diagram of $\mathfrak{M}$ as a source over the alphabet $X \times Y$, the graph of T will coincide with the ω-language carried by the flow consisting of all possible ω-paths through q_0. Thus the graph of T is representable in the source $\langle \mathfrak{M}, q_0, \mathfrak{C} \rangle$, where $\mathfrak{C}$ consists of all subsets of the set of vertices of $\mathfrak{M}$. It is natural to ask whether the converse is true:

A. *If the graph of an operator T is representable in a finite automaton, is the operator T realizable by a suitable finite automaton?*

The following simple example shows that, if no additional restrictions are imposed, the answer is negative. Consider the ω-language $\mathfrak{M}$ over the alphabet $\{0, 1\} \times \{0, 1\}$ represented by the source of Figure 15*c*, where q_0, q_1 are the initial vertices and $\{q_0\}$ and $\{q_2\}$ the limit macrostates. For every ω-word x over the alphabet $\{0, 1\}$, there exists a unique ω-word y over the alphabet $\{0, 1\}$ such that $\langle xy \rangle \in \mathfrak{M}$: for $00 \ldots 0 \ldots$ (all zeros)

* *Translator's note:* Not to be confused with the term from Graph Theory. The ambiguity is unavoidable in English; Russian has two different words.

the corresponding ω-word y is $00\ldots0\ldots$; for every other x the corresponding y is $111\ldots1\ldots$ (all ones). However, it is easy to see that the operator induced by this correspondence is an anticipatory operator (even with unbounded anticipation!), and so it cannot be realized by any (even infinite) automaton.

A natural modification of question **A** runs as follows:

A'. *If the graph of a nonanticipatory operator is representable in a finite automaton, is the operator realizable by a finite automaton?*

The answer to this question is positive, and a proof could be given here. However, we find it more convenient to treat the question in a more general framework: the problem of the uniformization of representable ω-languages.

Given an ω-language $\mathfrak{H}$ over an alphabet $X \times Y$. An operator T is said to *uniformize* $\mathfrak{H}$ if its graph is contained in $\mathfrak{H}$. The preceding example shows that, in general, existence of an operator (there may be several) which uniformizes a finite-state ω-language $\mathfrak{H}$ does not guarantee the existence of a *finite-state* operator (or even a nonanticipatory operator realizable by an infinite automaton) which uniformizes $\mathfrak{H}$.

The uniformization problem is as follows.

Given a macroanchored finite automaton $\langle \mathfrak{M}, q_0, \mathfrak{C} \rangle$,

(I) determine whether there exists a finite-state operator T uniformizing the ω-language $\Omega(\mathfrak{M}, q_0, \mathfrak{C})$,

and, if so,

(II) construct a finite automaton realizing this operator.

A solution of this problem would yield an algorithm which, given any automaton $\langle \mathfrak{M}, q_0, \mathfrak{C} \rangle$, provides an answer to the question (I) and, in case of a positive answer, constructs a suitable automaton. Construction of a finite-state uniformizing operator (when this is possible) is of fundamental importance in the synthesis theory of automata. The various questions arising in this connection will be treated using a proof of Büchi and Landweber which employs McNaughton's suggested game-theoretic interpretation.

II.6. More about decision problems of finite automata

In Section I.4 we touched upon this question with regard to outputless automata. Now, after our discussion of the connection between the behavior of outputless automata and automata with output, it should already be clear how one formulates the related concepts and results for the properties of finite automata with output. As an example, consider the following

problem. Given a finite automaton $\langle \mathfrak{M}, q_0 \rangle$, to determine whether it can generate (for suitable input) an output ω-word containing infinitely many occurrences of a letter z_0. Consider the ω-language L which $\mathfrak{M}$ represents, with initial state q_0, by the system $\mathfrak{C}$ of all output subalphabets containing the letter z_0. It is clear that our original question may be reformulated as follows: Is the ω-language L nonempty? To find the answer, we need only construct (effectively!) an outputless automaton representing the ω-language L (see Theorem 2.6, Section II.5) and apply the algorithm of Theorem 1.5 (Section I.4).

We consider another example, which involves decidability of the properties of a family of automata. Given automata $\langle \mathfrak{M}_1, q_0 \rangle$ and $\langle \mathfrak{M}_2, \pi_0 \rangle$ with common input alphabet, it is required to determine whether there exists an input word ξ for which both automata produce the same output word. That this property is effectively decidable for pairs of automata is quite obvious, even trivial, for if there exists such a word $\xi = x(1)x(2)\dots x(s)$ then the one-letter word $x(1)$ alone generates the same (one-letter!) output word at the output of both automata.

Now consider the following, more complicated property for an ordered triple of finite automata $\langle \mathfrak{N}, \mathfrak{M}_1, \mathfrak{M}_2 \rangle$: The language* $\omega(\mathfrak{N})$ represented by $\mathfrak{N}$ contains a word which both automata $\mathfrak{M}_1$ and $\mathfrak{M}_2$ transform into the same output word (assume, say, that $\mathfrak{N}$ is an anchored outputless automaton). The proof that this property is effectively decidable is also quite simple, though not as trivial as in the previous example. It suffices to verify that, given the pair of automata $\mathfrak{M}_1$ and $\mathfrak{M}_2$, one can effectively construct an automaton $\mathfrak{M}$ such that the language $\omega(\mathfrak{M})$ consists of precisely those words for which $\mathfrak{M}_1$ and $\mathfrak{M}_2$ produce the same output. One then constructs an automaton representing the intersection of the languages $\omega(\mathfrak{M})$ and $\omega(\mathfrak{N})$ and determines whether it is empty. The algorithm constructing $\mathfrak{M}$ from the pair $\mathfrak{M}_1, \mathfrak{M}_2$ involves no difficulties, but nevertheless we shall describe it in sufficient detail to emphasize its simplicity. First construct an automaton $\tilde{\mathfrak{M}}$ with the same input alphabet X as the automata $\mathfrak{M}_1, \mathfrak{M}_2$ and output alphabet $\{0, 1\}$, which functions as follows: let T_1x, T_2x, $\tilde{T}x$ be the words into which the automata $\mathfrak{M}_1, \mathfrak{M}_2, \tilde{\mathfrak{M}}$, respectively, transform the input word x; then $\tilde{T}x$ ends with 1 if and only if T_1x and T_2x end with the same letter. The states of $\tilde{\mathfrak{M}}$ are the pairs $\langle q, \pi \rangle$, where q is a state of $\mathfrak{M}_1$

* We use the abbreviated notation $\mathfrak{N}, \omega(\mathfrak{N})$, omitting the initial state, output subalphabet, etc.

and π a state of $\mathfrak{M}_2$. Suppose that $\Psi_1(q, a) = q'$, $\Phi_1(q, a) = y$ in the automaton $\mathfrak{M}_1$, and $\Psi_2(\pi, a) = \pi'$, $\Phi_2(\pi, a) = z$ in the automaton $\mathfrak{M}_2$; then the next-state and output functions of $\mathfrak{M}$ are defined by

$$\Psi[\langle q, \pi\rangle, a] = \langle q', \pi'\rangle,$$

$$\Phi[\langle q, \pi\rangle, a] = \begin{cases} 1 & \text{for} \quad y = z, \\ 0 & \text{for} \quad y \neq z. \end{cases}$$

Now, starting from the initialized automaton $\langle \tilde{\mathfrak{M}}, \langle q_0, \pi_0\rangle\rangle$, construct an automaton $\langle \mathfrak{M}, \rho_0\rangle$ which functions as follows: If $\tilde{\mathfrak{M}}$ transforms an input ω-word $x(1)x(2)\ldots x(t)\ldots$ into an output ω-word $y(1)y(2)\ldots y(t)\ldots$, and $y(t_0)$ is the first occurrence of the letter 0, then $\mathfrak{M}$ transforms the same input ω-word into the ω-word $y(1)y(2)\ldots y(t_0 - 1)00\ldots$ (zeros *ad infinitum*). It is clear that the finite automaton $\mathfrak{M}$ represents the required language by the output 1.

Thus, the above property of triples of automata is effectively decidable, and the algorithm is fairly simple. This property is nevertheless instructive, in that a slight (at first sight) modification thereof leads to a property of triples of automata which is no longer effectively decidable. Fix some letter from the output alphabet of the automata $\mathfrak{M}_1, \mathfrak{M}_2$; denote this letter by e. Suppose that $\mathfrak{M}_1$ and $\mathfrak{M}_2$ transform an input word x into words y_1 and y_2, respectively. It may happen that deletion of all occurrences of e from y_1 and y_2 yields identical nonempty words y_1', y_2'. We shall then say that $\mathfrak{M}_1$ and $\mathfrak{M}_2$ transform the word x into the same word "up to e." Now consider the following property of triples of automata $\mathfrak{N}, \mathfrak{M}_1, \mathfrak{M}_2$: There exists a word x in the language $\omega(\mathfrak{N})$ which $\mathfrak{M}_1, \mathfrak{M}_2$ transform into the same word up to e. Thus, the only modification made in the original property is that identity of output words has been replaced by identity up to e. Nevertheless:

THEOREM 2.7. *There is no algorithm which, given a triple of finite automata $\mathfrak{N}, \mathfrak{M}_1, \mathfrak{M}_2$, determines whether the language $\omega(\mathfrak{N})$ contains a word which the automata $\mathfrak{M}_1, \mathfrak{M}_2$ transform into the same word up to e.*

Our proof will be based on an important theorem of Post, which we recall. Let X be some alphabet, and

$$(a_1, b_1), (a_2, b_2), \ldots, (a_s, b_s) \qquad (\#)$$

a sequence of word pairs over this alphabet. Fixing some finite sequence of indices $i_1, i_2, \ldots, i_k$, where $1 \leqq i_j \leqq s$, we can form the word pair $a_{i_1}a_{i_2}\ldots a_{i_k}$ and $b_{i_1}b_{i_2}\ldots b_{i_k}$; we shall say that this pair corresponds to the

index sequence $i_1, i_2, \ldots, i_k$. The Post *correspondence problem* for the system of pairs (#) is to determine whether there exists a sequence of indices for which the corresponding words are identical. Post's Theorem (on the undecidability of the correspondence problem) states that there is no algorithm which, given the system (#) over an alphabet consisting of at least two letters, solves the correspondence problem.

The Post correspondence problem for a system (#) of pairs has a natural interpretation in terms of coding theory. Let the indices 1, 2, 3, ..., s be letters of an alphabet Σ, which are encoded in one system as words a_1, $a_2, \ldots a_s$ over X, in another as words $b_1, b_2, \ldots, b_s$ over X. As usual, the code of a word p over Σ is the concatenation of the codes of its letters. Then the system (#) has a solvable correspondence problem if and only if there exists a word over X which has the same codes in both systems.

Proof of Theorem 2.7. Given any system (#), one can effectively construct a triple of finite automata $\mathfrak{N}$, $\mathfrak{M}_1$, $\mathfrak{M}_2$ such that the system (#) has a solvable correspondence problem if and only if the triple has the property mentioned in the theorem. The existence of an algorithm recognizing this property would imply the decidability of the Post correspondence problem, contradicting Post's Theorem. Thus, let us see how to construct $\mathfrak{N}$, $\mathfrak{M}_1$, $\mathfrak{M}_2$ for a given system (#). The input alphabet for $\mathfrak{N}$, $\mathfrak{M}_1$ $\mathfrak{M}_2$ will be the set Σ' consisting of the indices 1, 2, ..., s and the letter e. Without loss of generality we may assume that the alphabet X in which the words $a_1, \ldots, a_s, b_1, \ldots, b_s$ are written does not contain e. The output alphabet of $\mathfrak{M}_1$ and $\mathfrak{M}_2$ will be $X \cup \{e\}$. Let r be the maximum length of the words $a_1, \ldots, a_s, b_1, \ldots, b_s$; let $\tilde{a}_i$ (or $\tilde{b}_i$) be the word obtained by adding as many e's to the right of the word a_i (or b_i) as are needed to get a word of length r (of course, there may be a_i, b_i such that $a_i = \tilde{a}_i$, $b_i = \tilde{b}_i$). We now define $\mathfrak{N}$, $\mathfrak{M}_1$, $\mathfrak{M}_2$ as follows.

1) The finite automaton $\mathfrak{N}$ represents the language consisting of all words

$$\underbrace{i_1 e \ldots e}_{r-1} \underbrace{i_2 e \ldots e}_{r-1} \ldots \underbrace{i_\mu e \ldots e}_{r-1},$$

where $1 \leqq i_j \leqq s$.

2) The finite automaton $\mathfrak{M}_1$ ($\mathfrak{M}_2$) transforms any word of the above form into the word $\tilde{a}_{i_1}\tilde{a}_{i_2} \ldots \tilde{a}_{i_\mu}$ ($\tilde{b}_{i_1}\tilde{b}_{i_2} \ldots \tilde{b}_{i_\mu}$) and is defined arbitrarily for all other words; for example, it might produce the output letter e when it finds that the input word is not of the required form. Since construction of automata satisfying conditions 1), 2) involves no difficulties, we shall omit a detailed description. It is also easy to see that the resulting finite automata

$\mathfrak{N}$, $\mathfrak{M}_1$, $\mathfrak{M}_2$ are the required automata for the system (#). This completes the proof.

In Chapter I we established several algorithms which, given finite automata, either solve some decision problem or construct other finite automata. In some cases, the actual description and justification of these algorithms was quite complicated. In other cases, though the basic idea underlying the algorithm is simple and obvious, actual application is extremely cumbersome and practically unfeasible, owing to the wealth of alternatives to choose from. Nevertheless, the very existence of these algorithms has immense theoretical value, since, as evinced by Theorem 2.7, there may be no decision algorithm for very simple properties of finite automata. This remark applies equally well to the algorithmic problems of finite automata theory which we shall consider both here and in later chapters.

II.7. Games, strategies and nonanticipatory operators

We shall consider a special class of games with perfect information, which we shall call simply *games*. These games have the following properties.

1. Each game involves two players, called black and white.

2. At each move, the player must choose (according to the rules of the game) one alternative from a set of alternatives $X = \{x_1, \ldots, x_m\}$ for black and $Y = \{y_1, \ldots, y_n\}$ for white. The choice depends on the player alone, no random choice mechanism being involved.

3. The first move is always made by black. Subsequent moves alternate between white and black; we shall call an entire sequence of moves a *game history*. There are two possibilities:

1) *Finite game history*: The game stops after a finite number of moves (in accordance with the rules). Without loss of generality, we shall henceforth assume that the last move in any finite game history is made by white. Thus any finite game history may be regarded as a word over the alphabet $X \times Y$

$$\langle x(1)y(1)\rangle \langle x(2)y(2)\rangle \ldots \langle x(t)y(t)\rangle,$$

or a pair of words

$$x(1)x(2)\ldots x(t), \quad y(1)y(2)\ldots y(t),$$

describing the moves of black and white, respectively.

2) *Infinite game history*: The moves alternate indefinitely. Thus an infinite game history may be regarded as an ω-word over $X \times Y$

$$\langle x(1)y(1)\rangle \langle x(2)y(2)\rangle \ldots \langle x(t)y(t)\rangle \ldots$$

or a pair of ω-words

$$x(1)x(2)\ldots x(t)\ldots, \quad y(1)y(2)\ldots y(t)\ldots$$

4. At each move, the player knows all the preceding moves in the game history (briefly, he has perfect information on the opening of the game).

5. There are only two possible, mutually exclusive outcomes for any game history (according to the rules of the game): black wins or white wins.

A game is said to be *finite* if there exists a constant μ such that no game history exceeds μ in length.

Every finite game may be described by a tree whose edges are labeled with letters from $X \times Y$, such that edges issuing from the same vertex are differently labeled. The final vertices fall into two classes: those labeled $\oplus$ (white wins) and those labeled $\ominus$ (black wins). At the beginning of the game, a counter is placed at the root of the tree. Black chooses a letter $x_i \in X$, which is the X-label of certain edges on the first level—suppose these edges are labeled

$$\langle x_i y_r\rangle, \langle x_i y_s\rangle, \ldots, \langle x_i y_p\rangle.$$

Then white chooses a letter which is the second component of one of these pairs, say y_s. Once these two moves have been made, the counter is moved to the vertex γ at the end of the edge labeled $\langle x_i y_s\rangle$. Now black chooses an X-label on an edge issuing from γ, white chooses a suitable Y-label, and the counter is moved again. The game ends when the counter first reaches a final vertex, whose label identifies the winner. Figure 16*a* illustrates the tree of a game with $X = \{a, b\}$, $Y = \{\alpha, \beta\}$. The game history $\begin{matrix} a & b & a \\ \beta & \alpha & \beta \end{matrix}$ is a winning history for black. It is clear that white could have won by choosing α instead of β on the last move.

Games in which each history is infinite are naturally called *infinite* games. We shall confine ourselves to infinite games in which, at each step, the player whose move it is may choose any alternative from one of the sets

$$X = \{x_1, x_2, \ldots, x_m\} \qquad \text{or} \qquad Y = \{y_1, y_2, \ldots, y_n\}.$$

A game of this type may be represented by an infinite tree in which there are mn edges issuing from each vertex, labeled with all possible pairs $\langle x_i y_j\rangle$. Thus, when X and Y are fixed, infinite games may differ only in the set of winning game histories for white;* this set is an ω-language over the alphabet

* Or, equivalently, the complementary set—black's winning game histories.

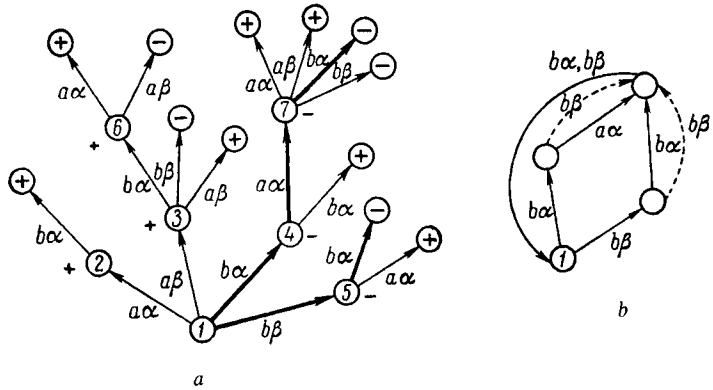


Figure 16

$X \times Y$. Conversely, any ω-language $\mathfrak{H}$ over $X \times Y$ defines an infinite game (denoted by $\tilde{\mathfrak{H}}$)—the game for which $\mathfrak{H}$ is the set of winning histories for white. In this sense one can speak of the game-theoretic interpretation of ω-languages.

The most important concept in the theory of games is the *strategy*. In our case, white's strategy indicates, for every possible opening, the choice of an admissible alternative. Since black makes the first move, his strategy, apart from indicating the choice made on his next move (and this choice depends on the opening extending up to the time in question), also indicates his first move. We first consider infinite games. White's strategy is then precisely a nonanticipatory operator, transforming "input" ω-words $x = x(1)x(2)\ldots$ (black's possible sequences of moves) into "output" words $y = y(1)y(2)\ldots$ (white's "answering" moves). Black's strategy is a nonanticipatory operator, transforming "input" words $y = y(1)y(2)\ldots$ into "output" ω-words $x = x(1)x(2)\ldots$ with the additional condition: for any $t > 1$, $x(t)$ depends only on $y(1)y(2)\ldots y(t-1)$, and $x(1)$ is a constant, independent of the input. Such an operator is known as an *operator with delay*. The converse assertions are also valid: any nonanticipatory operator $y = T''x$ is a strategy for white, and any operator $x = T'y$ with delay is a strategy for black.

In particular, the operator T'' (or T') may turn out to be finite-state. In this case we shall also call the strategy finite-state. Note that a finite (initialized) automaton realizing a strategy for black is an automaton with delay (see Section 0.2).

For finite games, strategies are again nonanticipatory operators, but they are defined only over a finite set of words. Any such operator may be

extended to a finite-state operator defined on the entire set of input words (therefore also on all input ω-words). For this reason, finite-state strategies for finite games are of little interest.

Once black and white have chosen their strategies T' and T'', respectively, the game history is uniquely determined; let us denote it by $\langle T', T'' \rangle$. It is easy to see that if T' and T'' are finite-state strategies of an infinite game, then $\langle T', T'' \rangle$ is a periodic ω-word over the alphabet $X \times Y$, whose period is at most the product of the weights of the operators T' and T''. Given automata defining T' and T'', this periodic ω-word (more precisely, an initial segment of it containing both phase and period) is effectively constructible. If white (black) wins in the game history $\langle T', T'' \rangle$ (of a finite or infinite game), we shall say that T'' beats T' (T' beats T''). It is easily seen that if a strategy T'' beats any strategy for black (T' beats any strategy for white), it will always lead to a win for white (black), however the opposite side plays. A strategy of this type is said to be a winning strategy for white (black). It is quite evident that in no game can both sides have a winning strategy. One can ask, does at least one of the players have a winning strategy? It is well known that for finite games the answer is positive. There is an essentially simple procedure (see the end of this section for an example) which, given the tree of a finite game, determines which side has a winning strategy and constructs one (there are generally several). According to foregoing remarks, we may assume that this strategy is finite-state. On the other hand, using set-theoretical arguments based on the Axiom of Choice, one can prove that there exist infinite games in which neither player has a winning strategy (even with infinite weight).

EXAMPLE. Let us find a winning strategy for the finite game of Figure 16*a*. Each nonfinal vertex of the tree is the root of a subtree, which defines a subgame. "Descending" from the upper vertices to the root, we shall determine winning strategies for all subgames step by step, finally finding a winning strategy for the entire game. With the vertex 6 we associate $+$, indicating that in the subgame with root 6 white has a winning strategy: white wins if he makes the move α for black's only possible move a. The vertex 7 has the sign $-$: black wins by choosing b. We can now consider the shorter game, in which the vertices 6 and 7 are regarded as final, with signs $+$, $-$, respectively. Thus we can continue our deliberations for the vertices 2, 3, 4, 5 and, finally, for 1.

It turns out that in this game black has a winning strategy T', which is a nonanticipatory operator with delay.

The operator (strategy) T' is defined by the subtree indicated in Figure 16*a* by bold arrows, on the assumption that the labels α, β are inputs, a, b outputs. This subtree is defined as follows: among the edges issuing from the root, retain only those labeled b; the letter b is black's first move, and all the edges chosen lead to "minus" vertices. Among the edges issuing from these vertices, retain those corresponding to black's next move; these also lead to "minus" vertices, etc. Figure 16*b* illustrates the diagram of a finite automaton (one of many possible) realizing this strategy. To fix ideas, the x-label of the dashed edges is b.

II.8. Game-theoretic interpretation of the uniformization problem

Consider an arbitrary ω-language $\mathfrak{H}$ over the alphabet $X \times Y$ and the corresponding infinite game $\tilde{\mathfrak{H}}$. Our discussion of strategies in the preceding section shows that a strategy T'' for white is precisely a nonanticipatory operator $y = T''x$ which uniformizes the ω-language $\mathfrak{H}$; a strategy T' for black is a nonanticipatory operator $x = T'y$ with delay, which uniformizes the complementary ω-language $\neg\mathfrak{H}$. The uniformization problem may thus be interpreted as the problem of existence and construction of a winning finite-state strategy for white in the game induced by the ω-language $\Omega(\mathfrak{M}, q_1, \mathfrak{C})$. We shall consider games of this type, which we shall call *finite-state games*, in greater detail. They may be described in the following way. At the beginning of the game, the counter is placed at the vertex q_1 of the diagram of $\mathfrak{M}$. Black announces his first move $x(1)$, white answers with $y(1)$, and the counter is moved to the next vertex along the edge labeled $\langle x(1)y(1)\rangle$. Now black chooses $x(2)$, and after white's answer $y(2)$ the counter is moved as required. The alternating moves continue *ad infinitum*:

$$x(1)x(2)\ldots x(t)\ldots,$$

$$y(1)y(2)\ldots y(t)\ldots,$$

and the counter runs through a sequence of vertices

$$q_1 = q(1), q(2), \ldots, q(t), q(t+1), \ldots$$

White wins if $\lim q(t) \in \mathfrak{C}$ and loses otherwise. White's strategy T'' is *winning* if, whatever black's choice of moves, the counter "homes" into a limit macrostate from $\mathfrak{C}$. The macrostates of $\mathfrak{C}$ will be called *favorable* for white, all others favorable for black.

EXAMPLE. Consider the diagram $\mathfrak{M}$ of Figure 17, where $X = \{a, b\}$,

$Y = \{c, d\}$. The initial state is q_1, the favorable macrostates for black are $\{q_1\}$, $\{q_2\}$, $\{q_1, q_2, q_3\}$, the other nonempty macrostates $\{q_3\}$, $\{q_1, q_2\}$, $\{q_1, q_3\}$, $\{q_2, q_3\}$ are favorable for white. In this game, black has a winning

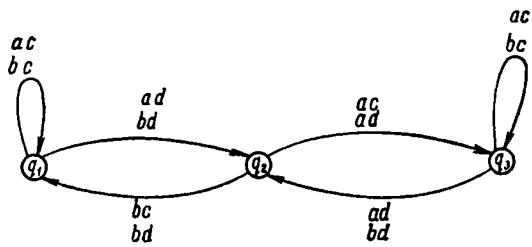


Figure 17

strategy. The first move is arbitrary (say a). Suppose that when black must choose his next move, the counter is at the vertex q_2. If the immediately preceding position of the counter was q_1, black plays a; this ensures that the next vertex will be q_3. If the preceding position of the counter was q_3, black plays b, thereby ensuring that the counter will move to q_1. In all other cases black's moves are immaterial; say he always plays a. It is easily seen that this strategy is finite-state. The operator $x = T'y$ is described by the relations

$$x(t) = \Phi(p(t)),$$

$$p(t+1) = \Psi(p(t), y(t)),$$

where $p(t) = \langle q(t), q(t-1)\rangle$, i.e., the states of a finite automaton realizing this strategy are pairs of states of the automaton describing the game. Here $\Phi[\langle q_2 q_3\rangle] = b$ and $\Phi[\langle q_i q_j\rangle] = a$ in all other cases; for $p(1)$ one can take $\langle q_1 q_2\rangle$. The next-state function Ψ is easily defined on the basis of the game; it is given in Table 8 below.

TABLE 8

y \ p	$\langle q_1q_1\rangle$	$\langle q_1q_2\rangle$	$\langle q_2q_3\rangle$	$\langle q_3q_3\rangle$	$\langle q_3q_2\rangle$	$\langle q_2q_1\rangle$
c	$\langle q_1q_1\rangle$	$\langle q_1q_1\rangle$	$\langle q_1q_2\rangle$	$\langle q_3q_3\rangle$	$\langle q_3q_3\rangle$	$\langle q_3q_2\rangle$
d	$\langle q_2q_1\rangle$	$\langle q_2q_1\rangle$	$\langle q_1q_2\rangle$	$\langle q_2q_3\rangle$	$\langle q_2q_3\rangle$	$\langle q_3q_2\rangle$

It was mentioned in Section II.7 that there exist infinite games in which neither side has a winning strategy. However, if we restrict ourselves to finite-state games, the situation is completely different:

THEOREM 2.8. (FUNDAMENTAL THEOREM ON FINITE-STATE GAMES). *In any finite-state game, one of the sides has a winning finite-state strategy. There is an algorithm which, given a finite-state game, (I) determines which side has a winning strategy, and (II) constructs a winning finite-state strategy.*

COROLLARY. *If one of the sides in a finite-state game has a winning strategy, it also has a finite-state winning strategy.*

In view of the significance of this theorem, we shall rephrase it, together with its corollary, in terms of the uniformization of finite-state ω-languages.

THEOREM 2.8′ (*rephrased*). *Given an arbitrary ω-language $\mathfrak{H}$ over an alphabet $X \times Y$, representable in a finite automaton. Then either $\mathfrak{H}$ is uniformizable by a finite-state operator $y = Tx$, or $\neg\, \mathfrak{H}$ is uniformizable by a finite-state operator $x = Ty$ with delay. There is an algorithm which, given an automaton representing $\mathfrak{H}$, determines which of these situations actually holds and constructs an automaton realizing the appropriate operator.*

COROLLARY (*rephrased*). *If there exists a nonanticipatory operator uniformizing a finite-state ω-language $\mathfrak{H}$, there also exists a finite-state operator uniformizing $\mathfrak{H}$. In particular, if the graph of a nonanticipatory operator is representable in a finite automaton, the operator itself is also definable in a finite automaton.*

The proof of the Fundamental Theorem 2.8 and the requisite preliminary concepts and propositions will be presented in Sections II.9 and II.10.

II.9. Proof of the Fundamental Theorem on finite-state games—intuitive outline *

Let $\langle \mathfrak{M}, q_0, \mathfrak{C} \rangle$ be a finite-state game with state set $Q = \{q_0, \ldots, q_n\}$, initial state q_0, next-state function $\Psi: X \times Y \times Q \to Q$, and limit macrostates $\mathfrak{C} = \{Q_1, \ldots, Q_m\}$. For any game history

$$x(1)x(2)\ldots, \quad y(1)y(2)\ldots,$$

let the corresponding sequence of states (vertices) be

$$q(1)\,q(2)\ldots$$

Recall that a winning white strategy must for any sequence of black moves $x(1)x(2)\ldots$ produce a sequence $y(1)y(2)\ldots$ such that the limit

* *Translator's note*: Sections II.9 and II.10 were written by L. Landweber for the English edition of this book, replacing the original version of the authors.

macrostate of the associated state sequence $q(1)q(2)\ldots$ is a member of $\mathfrak{C}$. A winning black strategy must prevent the limit macrostate from being in $\mathfrak{C}$, regardless of white's moves. White's strategy at time t chooses $y(t)$ based on knowledge of $x(1)y(1)\ldots x(t-1)y(t-1)x(t)$, while black's strategy may only use $x(1)y(1)\ldots x(t-1)y(t-1)$ in selecting $x(t)$. The main result of this section is that for any such game, either black or white has a winning strategy, and in fact a finite-state winning strategy.

To motivate the definitions and proofs which follow, we first consider the simple case $\mathfrak{C} = \{Q_1\}$. If white has a winning strategy, then it will have a winning strategy T'' which operates as follows (this will be proved in Section II.10). First T'' forces the game to some $q \in Q_1$ (i.e., for any moves by black, T'' produces moves which lead to the game entering a $q \in Q_1$). After selecting a *state goal* $q' \in Q_1$, T'' attempts to force the game to q' with all intermediate states being members of Q_1. The strategies to be considered will choose successive state goals by referring to a cyclic permutation of the members of Q_1. Hence, if the above is possible, Q_1 will be the limit macrostate. Otherwise black can either prevent the game from entering some state of Q_1 or infinitely often force the game out of Q_1. In either case Q_1 will not be the limit macrostate, so that black instead of white will have a winning strategy.

If $\mathfrak{C} = \{Q_1, \ldots, Q_n\}$, $Q_i \supset Q_{i+1}$, $i = 1, \ldots, n-1$ ($\supset$ will always mean proper inclusion), the situation will be somewhat more complicated. If white has a winning strategy, then it will be shown that white has a winning strategy T'' which operates as follows: T'' first forces the game to some q in some Q_i, say Q_1. A state goal q_1 for Q_1 is then selected and T'' attempts to force the game to q_1 (while remaining in states of Q_1). However, black may be able to prevent this, although in the process the game will reach a $q' \in Q_2$ (having stayed in states of Q_1, $Q_1 \supset Q_2$) from which T'' can force the game either to q_1, to a newly chosen Q_2 state goal $q_2 \in Q_2$, or to some $q'' \in Q_3$, which results in a Q_3 state goal $q_3 \in Q_3$ being chosen. If $n = 3$ and three state goals q_1, q_2, q_3 have been chosen, the winning strategy T'' will then be able to force the game to one of q_1, q_2, or q_3 while remaining in states of Q_1, Q_2, or Q_3, respectively. If q_1, then q_2, q_3 are forgotten and a new state goal for Q_1 is chosen. If q_2, then q_3 is forgotten and a new Q_2 state goal is selected. If q_3, then a new Q_3 state goal is selected. The state goals are determined by fixing a cyclic permutation of each Q_i and then, whenever a new state goal is required, choosing the next element in the permutation. If T'' is able to operate in the above manner, then the limit macrostate will be one of the Q_i.

The most complicated case occurs when $\mathfrak{C}$ is a collection of macrostate chains and the overlapping of macrostates in different chains is permitted (see Figure 17*a* for an example). If white has a winning strategy, then it will have a winning strategy T'' with the following properties. At each time t, T'' will be considering a chain of macrostates in $\mathfrak{C}$ (possibly an empty chain), $Q_1 \supset \ldots \supset Q_r, r \geqq 0$, and associated states $q_1, \ldots, q_r$, where $q_i \in Q_i$, $i = 1, \ldots, r$. Call Q_i and q_i the i-th level set and state goals, respectively. At time t, T'' will be attempting to force one of $Q_1, \ldots, Q_r$ to be the limit macrostate. T'' will either:

(1) force the game to one of the q_i (without leaving states of Q_i), in which case $Q_{i+1}, q_{i+1}, \ldots, Q_r, q_r$ are forgotten and a new i-th level state goal is selected;

(2) select an additional set and state goal $Q_{r+1} \subset Q_r, q_{r+1} \in Q_{r+1} \in \mathfrak{C}$; or

(3) forget all set and state goals after some level i (this includes the possibility of forgetting all state and set goals), while getting "closer" to one of the remaining state goals (or if all are forgotten, "closer" to a time after which all state and set goals will not be forgotten).

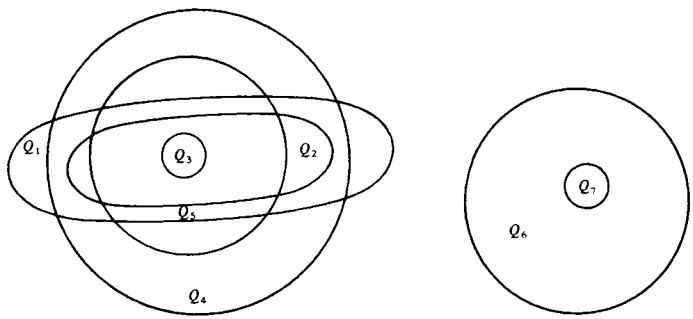


Figure 17*a*

Alternative (1) is omitted if at time t the chain is empty. Intuitively, this means that T'' is in the process of forcing the game to a state in some member of $\mathfrak{C}$ after which a state goal will be chosen. It is crucial in (1) that q_i be reached without leaving states of Q_i, since otherwise T'' will not be working towards its goal of forcing one of $Q_1, \ldots, Q_i$ to be the limit macrostate.

Alternative (2) results in the selection of an additional state and set goal. Note that because $\mathfrak{C}$ is finite, (2) can recur only a finite number of times in a row without (1) or (3) occurring. The strategy T'' will be designed so as to prevent alternative (3) from recurring indefinitely without (1) being applied

(i.e., "closer" will be well defined). The forgetting of all levels will only be allowed to occur finitely often in the course of the game.

It will be shown that the above considerations result in $Q_1, q_1, \ldots, Q_k, q_k$, for some $k > 0$, being eventually fixed (though at various times different $Q_{k+1}, q_{k+1}, \ldots$ may be added and removed). $Q_k \in \mathfrak{C}$ will then be the limit macroset.

The informal characteristics discussed above are incorporated into the definition of a white winning strategy. Such a strategy will make use of sets of predicates $\{\mathfrak{R}_k\}$, $\{\mathfrak{Q}_k\}$ and $\{\mathfrak{P}_k\}$ on Q, the set of states of the game. If $q_0 \in \mathfrak{R}_l[\ ]$ (l as defined below), then white will have a winning strategy as above. If $q_0 \notin \mathfrak{R}_l[\ ]$, then black will have a winning strategy.

In Section II.10.1 the predicates $\{\mathfrak{R}_k\}$, $\{\mathfrak{Q}_k\}$, $\{\mathfrak{P}_k\}$ are defined. The case $q_0 \in \mathfrak{R}_l[\ ]$ is treated in Section II.10.2, where a finite-state winning strategy for white is presented. Section II.10.3 deals with the definition of a finite-state winning strategy for black in case $q_0 \notin \mathfrak{R}_l[\ ]$.

II.10. Proof of the Fundamental Theorem on finite-state games

II.10.1. DEFINITION OF $\mathfrak{R}_l[\ ]$

For each $Q_i \in \mathfrak{C}$, fix a cyclic permutation of its members. For simplicity of notation, denote the value of this permutation at $q \in Q_i$ by $Q_i(q)$. The crucial construction is that of the subsets of Q,

$$\mathfrak{R}_k[Q_1, q_1, \ldots, Q_r, q_r], \quad \mathfrak{Q}_k[Q_1, q_1, \ldots, Q_r, q_r]$$

and

$$\mathfrak{P}_k[Q_1, q_1, \ldots, Q_r, q_r],$$

where $k, r \geqq 0$, $Q_1 \supset \ldots \supset Q_r$, $q_i \in Q_i \in \mathfrak{C}$ for $i = 1, \ldots, r$. Note that the statement $A \supset B$ will always mean "B is a proper subset of A". These sets are defined simultaneously by the following induction on $k = 0, 1, 2, \ldots$:

$$q \in \mathfrak{R}_0[Q_1, q_1, \ldots, Q_r, q_r] \equiv \text{false}$$

$$q \in \mathfrak{R}_{k+1}[Q_1, q_1, \ldots, Q_r, q_r] \equiv \bigwedge_{x \in X} \bigvee_{y \in Y} \Psi[x, y, q] \in \{q_1, \ldots, q_r\} \cup$$

$$\cup\, \mathfrak{Q}_k[Q_1, q_1, \ldots, Q_r, q_r] \cup \mathfrak{P}_k[Q_1, q_1, \ldots, Q_r, q_r]$$

$$q \in \mathfrak{Q}_k[Q_1, q_1, \ldots, Q_r, q_r] \equiv \bigvee_{B} B \in \mathfrak{C} \,\&\, q \in B \,\&\, B \subset Q_r \,\& \qquad (1)$$

$$\bigwedge_{u \in B} u \in \mathfrak{R}_k[Q_1, q_1, \ldots, Q_r, q_r, B, B(u)]$$

$$q \in \mathfrak{P}_k[Q_1, q_1, \ldots, Q_r q_r] \equiv q \in \mathfrak{R}_k[\ ] \vee q \in (Q_1 \cap \mathfrak{R}_k[Q_1, q_1]) \vee$$

$$\vee \ldots \vee\ q \in (Q_r \cap \mathfrak{R}_k[Q_1, q_1, \ldots, Q_r, q_r]).$$

Note that r is bounded by the length of maximal subset chains in $\mathfrak{C}$. If $r = 0$, the notation $\mathfrak{R}_k[\,]$, $\mathfrak{Q}_k[\,]$, and $\mathfrak{P}_k[\,]$ is used and the recursions reduce to:

$$q \in \mathfrak{R}_0[\,] \equiv \text{false}$$

$$q \in \mathfrak{R}_{k+1}[\,] \equiv \bigwedge_{x \in X} \bigvee_{y \in Y} \Psi[x,y,q] \in \mathfrak{P}_k[\,] \cup \mathfrak{Q}_k[\,]$$

$$q \in \mathfrak{Q}_k[\,] \equiv \bigvee_{B} B \in \mathfrak{C} \,\&\, q \in B \,\&\, \bigwedge_{u \in B} u \in \mathfrak{R}_k[B, B(u)] \tag{2}$$

$$q \in \mathfrak{P}_k[\,] \equiv q \in \mathfrak{R}_k[\,].$$

It can easily be shown that the above definition (1) is well founded. Furthermore, for all $k \geqq 0$ and $\alpha = Q_1, q_1, \ldots, Q_r, q_r$, $r \geqq 0$, $\mathfrak{R}_k[\alpha] \subseteq \subseteq \mathfrak{R}_{k+1}[\alpha]$, $\mathfrak{Q}_k[\alpha] \subseteq \mathfrak{Q}_{k+1}[\alpha]$, $\mathfrak{P}_k[\alpha] \subseteq \mathfrak{P}_{k+1}[\alpha]$. Since all of these are subsets of the finite set Q, and there are but a finite number of α's, there is a smallest number l such that for all α, $i > 0$, $\mathfrak{R}_l[\alpha] = \mathfrak{R}_{l+i}[\alpha]$, $\mathfrak{Q}_l[\alpha] = = \mathfrak{Q}_{l+i}[\alpha]$, $\mathfrak{P}_l[\alpha] = \mathfrak{P}_{l+i}[\alpha]$. Hence a finite-state strategy can store (1) in its internal memory.

II.10.2. THE CASE $q_0 \in \mathfrak{R}_l[\,]$

Definition (1) is used to define a winning white strategy T'' in case $q_0 \in \mathfrak{R}_l[\,]$. If at time t the game is in state $q(t)$, the strategy T'' will be using some $\mathfrak{R}_k[\alpha]$, $\alpha = Q_1, \ldots, q_r$, where $q(t) \in \mathfrak{R}_k[\alpha]$ (at time 1, $\mathfrak{R}_l[\,]$ is used). Moreover, for each $1 \leqq j \leqq r$, there will be h_j such that for all $u \in Q_j$

$$u \in \mathfrak{R}_{h_j}[Q_1, \ldots, q_{j-1}, Q_j, Q_j(u)]. \tag{3}$$

$\{Q_i\}$ are the set goals, i.e., the members of $\mathfrak{C}$ which T'' is attempting to force to be the limit macroset. $q_i \in Q_i$ is the current i-th level or Q_i state goal. $q(t) \in \mathfrak{R}_k[\alpha]$ means that for any black move $x(t)$, T'' has a move $y(t)$ such that the next state

$$q(t+1) = \Psi[x(t), y(t), q(t)]$$

is in one of $\{q_1, \ldots, q_r\}$, $\mathfrak{Q}_{k-1}[\alpha]$ or $\mathfrak{P}_{k-1}[\alpha]$.

In the first case (corresponding to (1) in the above discussion of white's winning strategy for the general case in Section II.9), if $q(t+1) = q_i$, T'' selects a new i-th level state goal $Q_i(q_i)$ and uses

$$\mathfrak{R}_{h_i}[Q_1, \ldots, q_{i-1}, Q_i, Q_i(q_i)]$$

at time $t+1$. The relation (3) ensures that $q(t+1)$ is in this set.

If $q(t+1) \in \mathfrak{Q}_{k-1}[\alpha]$, then an $(r+1)$-th level set goal Q_{r+1} satisfying (3) is chosen, where $h_{r+1} = k - 1$. The process is then repeated using

$\mathfrak{R}_{k-1}[\alpha, Q_{r+1}, Q_{r+1}(q(t+1))]$, i.e., (2) in Section II.9. It should be emphasized that the process described is defined when a future state goal is reached only because all added set goals must satisfy (3).

If $q(t+1) \in \mathfrak{P}_{k-1}[\alpha]$, then all levels after some i-th level are forgotten and the process is repeated with index $k-1$ and

$$\mathfrak{R}_{k-1}[Q_1, \ldots, q_i].$$

This corresponds to (3) in Section II.9, where the index $k-1$ means that within $k-1$ moves some state goal must be reached (since each time this does not occur the index is lowered and $\mathfrak{R}_0[\alpha]$ is empty for all α). An important feature of this case is that if level $j, j \leqq i$, is retained, then $q(t+1) \in \in Q_j$. Finally, α can be empty (all levels forgotten) only a finite number of times, since each such time a lower index is used, $\mathfrak{R}_0[\,]$ is empty and $\mathfrak{R}_l[\,]$ is the first predicate considered. When α is empty, the strategy will within k moves (if $\mathfrak{R}_k[\,]$ is the predicate) select a 1-st level set and state goal.

If white uses the above strategy and $q_0 \in \mathfrak{R}_l[\,]$, eventually some Q_1, $q_1, \ldots, Q_k, q_k, k \geqq 1$, will be fixed, with $Q_k \in \mathfrak{C}$ becoming the limit macrostate. Hence T'' will be a winning strategy for white.

Choose a fixed linear order of the members of Y and $\mathfrak{C}$. The expression $(\mu y)E(y)$ denotes the first member y of Y, in the chosen order, which satisfies $E(y)$, if such a y exists.

In the following, $x(1)x(2)\ldots$, $y(1)y(2)\ldots$, $q(1)q(2)\ldots$ and $k(1)k(2)\ldots$ will be sequences over X, Y, Q, and $\{0, \ldots, l\}$, respectively. $v(1)v(2)\ldots$ will be a sequence of elements of the form

$$[Q_1, q_1, h_1, \ldots, Q_r, q_r, h_r],$$

where $r \geqq 0$, $Q_1 \supset \ldots \supset Q_r$, $q_i \in Q_i \in \mathfrak{C}$ for $i = 1, \ldots, r$ and $l > h_1 > \ldots > > h_r \geqq 1$. (The notation $[\,]$ will be used in case $r = 0$.)

A strategy T'' for white is defined as follows:

At time 1, the memory of T'' contains

$$q(1) = q_0, \quad k(1) = l, \quad v(1) = [\,]. \tag{4}$$

Assume that at time t the memory of T'' contains

$$q(t), \quad k(t), \quad v(t) = [Q_1, q_1, h_1, \ldots, Q_r, q_r, h_r], r \geqq 0. \tag{5}$$

Given $x(t)$, black's move at time t, white's move by T'' at time t is

$$y(t) = (\mu y)\Psi[x(t), y, q(t)] \in \{q_1, \ldots, q_r\} \cup$$
$$\cup \mathfrak{Q}_{k(t)-1}[Q_1, \ldots, q_r] \cup \mathfrak{P}_{k(t)-1}[Q_1, \ldots, q_r]. \tag{6}$$

The state of the game at $t + 1$ is

$$q(t + 1) = \Psi[x(t), y(t), q(t)]. \tag{7}$$

Finally, $v(t + 1)$ and $k(t + 1)$ are given by:

I. If $q(t + 1) \in \{q_1, \ldots, q_r\}$, let i be the first such that $q(t + 1) = q_i$. Then

$$v(t + 1) = [Q_1, q_1, h_1, \ldots, Q_i, Q_i(q_i), h_i], \qquad k(t + 1) = h_i. \tag{8}$$

II. If $q(t + 1) \in \mathfrak{Q}_{k(t)-1}[Q_1, \ldots, q_r]$ but not I, let B be the first macrostate in the chosen order of $\mathfrak{C}$ such that $q(t + 1) \in B \subset Q_r$ and

$$\bigwedge_{u \in B} u \in \mathfrak{R}_{k(t)-1}[Q_1, \ldots, q_r, B, B(u)]. \tag{*}$$

Then

$$v(t + 1) = [Q_1, q_1, h_1, \ldots, Q_r, q_r, h_r, B, B(q(t + 1)), k(t) - 1], \tag{9}$$

$$k(t + 1) = k(t) - 1.$$

III. If $q(t + 1) \in \mathfrak{P}_{k(t)-1}[Q_1 \ldots, q_r]$ but neither I nor II, let j be the first such that $q(t + 1) \in Q_j \cap \mathfrak{R}_{k(t)-1}[Q_1, \ldots, q_j]$. Then

$$v(t + 1) = [Q_1, q_1, h_1, \ldots, Q_j, q_j, h_j], \quad k(t + 1) = k(t) - 1. \tag{10}$$

If $r = 0$, all occurrences of $q_1, \ldots, q_r, Q_1, \ldots, Q_r, h_1, \ldots, h_r$ in the above definitions are omitted. If $\alpha = Q_1, \ldots, h_r$, then level $(\alpha) = r$ and (Q_i, q_i, h_i) is the i-th level of α. Similarly, $\mathfrak{R}_k[Q_1, \ldots, q_r]$ has level r, i-th level (Q_i, q_i) and index k.

Though an explicit proof is not given, it should be clear that the strategy T'' can be realized by a finite automaton.

That (6) – (10) are well founded in case $q_0 \in \mathfrak{R}_l[\,]$ follows directly from (1), (4) and the following lemma.

LEMMA 1. *If white uses* T'', $q_0 \in \mathfrak{R}_l[\,]$, *and* $v(t) = [Q_1, \ldots, h_r]$, *then*

(*a*) $q(t) \in (Q_r \cap \mathfrak{R}_{k(t)}[Q_1, \ldots, q_r])$ $(\mathfrak{R}_{k(t)}[\,]$ if $r = 0)$,

(*b*) *for all* $1 \leqq i \leqq r$, $u \in Q_i$,

$$u \in \mathfrak{R}_{h_i}[Q_1, \ldots, q_{i-1}, Q_i, Q_i(u)].$$

Proof. For $t = 1$, $v(1) = [\,]$, $k(1) = l$ and $q_0 = q(1) \in \mathfrak{R}_l[\,]$. Assume the lemma is true at time t. Let $x(t)$ be black's move at t. Since $q(t) \in \mathfrak{R}_{k(t)}[Q_1, \ldots, q_r]$, it follows from (1) that there is a $y(t)$ such that

$$q(t + 1) = \Psi[x(t), y(t), q(t)] \in \{q_1, \ldots, q_r\} \cup \mathfrak{Q}_{k(t)-1}[Q_1, \ldots, q_r] \cup \cup \mathfrak{P}_{k(t)-1}[Q_1, \ldots, q_r],$$

so that $y(t)$ is well defined at $t + 1$.

If $v(t+1)$ is defined by (8) (Case I; $q(t+1) = q_i$), then (b) is true at $t+1$ since it is true at t. (a) is true at $t+1$ since $q(t+1) = q_i \in Q_i$, $k(t+1) = h_i$ and (b) is true at t.

The proofs for (9) and (10) (Case II or III) are straightforward and are left to the reader. Q. E. D.

THEOREM 2.9. *If white uses T'' and $q_0 \in \mathfrak{R}_l[\,]$, then* $\lim(q(1)q(2)\dots) \in \mathfrak{C}$.

Proof. First note that if $t' > t$ and $v(t) = [\,]$, then $k(t') < k(t)$. This implies that there is a t_1 such that $v(t) \neq [\,]$ for $t > t_1$ (since for all t it is true that $v(t) = [Q_1, \dots, h_r]$ implies $q(t) \in \mathfrak{R}_{k(t)}[Q_1, \dots, q_r]$ and $\mathfrak{R}_0[Q_1, \dots, q_r]$ is empty for all $Q_1, \dots, q_r$).

As level $(v(t))$ is bounded by the length of the maximal macrostate chain of $\mathfrak{C}$, there is a smallest $j \geqq 1$ such that level $(v(t)) = j$ infinitely often. Let $t_2 > t_1$ be such that level $(v(t)) \geqq j$ for $t > t_2$. But then by (8), (9) (10), $v(t)$ for $t > t_2$ is of the form

$$v(t) = [Q_1, q_1, h_1, \dots, Q_j, -, h_j, \dots]$$

for fixed $Q_1, q_1, h_1, \dots, Q_{j-1}, q_{j-1}, h_{j-1}, Q_j, h_j$. By Lemma 1, this implies that $q(t) \in Q_j$ for $t > t_2$ so that $\lim(q(1)q(2)\dots) \subseteq Q_j$.

$v(t)$ has level j infinitely often. For $t > t_2$ this can only occur as a result of (8) or (10) for level j. But (10) can only be applied to give a j-th level $v(t)$ a finite number of times without (8) being applied, because each time the former occurs (without the latter having occurred) $k(t)$ is lower and by Lemma 1(a) $k(t)$ can never be 0. Hence (8) must be used infinitely often to obtain a j-th level $v(t)$. Each time this occurs the state goal of Q_j is entered and a new state goal is selected. The method of choosing state goals ensures that each $q \in Q_j$ is entered infinitely often, so $\lim(q(1)q(2)\dots) \supseteq Q_j$. Q. E. D.

COROLLARY. *If $q_0 \in \mathfrak{R}_l[\,]$, T'' is a winning strategy for white.*

II.10.3. THE CASE $q_0 \notin \mathfrak{R}_l[\,]$

The following recursions follow directly from (1) and the definition of l:

$$q \notin \mathfrak{R}_l[Q_1, q_1, \dots, Q_r, q_r] \equiv \bigvee_{x \in X} \bigwedge_{y \in Y} \Psi[x, y, q] \notin \{q_1, \dots, q_r\} \cup \cup \mathfrak{Q}_l[Q_1, \dots, q_r] \cup \mathfrak{P}_l[Q_1, \dots, q_r]$$

$$q \notin \mathfrak{Q}_l[Q_1, q_1, \dots, Q_r, q_r] \equiv \bigwedge_{B} [B \in \mathfrak{C} \,\&\, q \in B \,\&\, B \subset Q_r \Rightarrow \Rightarrow \bigvee_{u \in B} u \notin \mathfrak{R}_l[Q_1, \dots, q_r, B, B(u)]] \tag{11}$$

$$q \notin \mathfrak{P}_l[Q_1, q_1, \dots, Q_r, q_r] \equiv q \notin \mathfrak{R}_l[\,] \,\&\, q \notin (Q_1 \cap \mathfrak{R}_l[Q_1, q_1]) \,\& \,\& \dots \& \, q \notin (Q_r \cap \mathfrak{R}_l[Q_1, \dots, q_r]).$$

Choose a fixed linear order for the members of X. The expression $(\mu x)F(x)$ denotes the first member x, in the chosen order of X, which satisfies F.

In the following $u(1)u(2)\ldots$ will be a sequence of elements of the form

$$[Q_1, q_1, \ldots Q_r, q_r],$$

where $r \geqq 0$, $Q_1 \supset \ldots \supset Q_r$, $q_i \in Q_i \in \mathfrak{C}$ for $1 \leqq i \leqq r$. $w(1)w(2)\ldots$ will be a sequence of macrostate chains: each $w(t)$ is of the form $\{E_1, \ldots, E_s\}$, $E_1 \supset \ldots \supset E_s$, $s \geqq 1$. $x(1)x(2)\ldots$, $y(1)y(2)\ldots$, and $q(1)q(2)\ldots$ are as in Section II.10.2.

If $q_0 \notin \mathfrak{R}_l[\,]$, then (11) can be used to obtain a winning strategy T' for black. At time t the internal memory of T' will contain $q(t)$, $u(t) = [Q_1, \ldots, q_r]$, and $w(t) = \{E_1, \ldots, E_s\}$. This means that, at time t, T' will be trying to prevent the game from reaching $\{q_1, \ldots, q_r\}$, thereby preventing any of $Q_1, \ldots, Q_r$ from being the limit macrostate. If at some future time $\bar{t}$ the game leaves Q_i, it will be removed from $u(\bar{t})$. $E_i \in w(t)$ if and only if there is some $t' < t$ such that all and only states of E_i were entered between t' and t. Sets in $w(t) \cap \mathfrak{C}$ are candidates for inclusion in some future $u(\bar{t})$. Because $q_0 \notin \mathfrak{R}_l$, it will always be possible to eventually add to some $u(\bar{t})$ any $E_i \in \in \mathfrak{C} \cap w(t)$ which threatens to be the limit macrostate.

A strategy T' for black is defined as follows:

At time 1, the internal memory of T' contains

$$q(1) = q_0, \qquad u(1) = [\,], \qquad \text{and} \qquad w(1) = \{\{q(1)\}\}. \tag{12}$$

Assume that at time t the internal memory of T' contains

$$q(t), \qquad u(t) = [Q_1, \ldots, q_r], r \geqq 0, \qquad w(t) = \{E_1, \ldots, E_s\}, s \geqq 1.$$

Then

$$x(t) = (\mu x) \bigwedge_{y \in Y} \Psi[x, y, q(t)] \notin \{q_1, \ldots, q_r\} \cup \mathfrak{Q}_l[Q_1, \ldots, q_r] \cup \cup \mathfrak{P}_l[Q_1, \ldots, q_r]. \tag{13}$$

Assume that $y(t)$ is white's move at time t. Then

$$q(t+1) = \Psi[x(t), y(t), q(t)] \tag{14}$$

and

$$w(t+1) = \{B \cup \{q(t+1)\} : B \in w(t) \quad \text{or} \quad B = \varnothing\}. \tag{15}$$

Finally, $u(t+1)$ is defined as follows:

Let $Q_0 \supset Q$, $Q_{r+1} = \varnothing$.

I. If there exist B, $B \in \mathfrak{C} \cap w(t+1)$ and i, $0 \leqq i \leqq r$, such that (a) $Q_i \supset B \supset Q_{i+1}$ and (b) $q(t+1) \notin \mathfrak{R}_l[Q_1, \ldots, q_i, B, B(q(t+1))]$, then let B be the largest of these (B is in $w(t+1)$ which is a macrostate chain) and

$$u(t+1) = [Q_1, \ldots, q_i, B, B(q(t+1))]. \tag{16}$$

II. If not I, let i be such that $q(t+1) \in Q_i - Q_{i+1}$, $0 \leqq i \leqq r$, and

$$u(t+1) = [Q_1, \ldots, q_i]. \tag{17}$$

Note that if $r = 0$, Case I reduces to: There exist $B \in \mathfrak{C} \cap w(t+1), q(t+1) \notin \notin \mathfrak{R}_l[B, B(q(t+1))]$, and $u(t+1) = [B, B(q(t+1))]$. Case II reduces to: If not I, then $u(t+1) = [\,]$. If $r > 0$ and $i = 0$, then $u(t+1)$ is $[B, B(q(t+1))]$ in Case I and $[\,]$ in Case II.

As in Section II.10.2, it should be clear that the above strategy can be realized by a finite automaton.

That (13) to (17) are well founded in case $q_0 \notin \mathfrak{R}_l[\,]$ follows directly from (11), (12) and the following lemma.

LEMMA 2. *If black uses* T', $q_0 \notin \mathfrak{R}_l[\,]$, *and* $u(t) = [Q_1, \ldots, q_r]$, $r \geqq 0$, *then*:

(*a*) $q(t) \notin \mathfrak{R}_l[Q_1, \ldots, q_r]$.
(*b*) *If* $r > 0$, $q(t) \in Q_r$.
(*c*) *If* $r > 0$, $Q_j \in \mathfrak{C} \cap w(t)$ *for* $1 \leqq j \leqq r$.

Proof. $u(1) = [\,]$ and $q(1) = q_0 \notin \mathfrak{R}_l[\,]$, so the lemma is true for $t = 1$.

Assume $u(t) = [Q_1, \ldots, q_r]$, $r \geqq 0$, and (*a*) through (*c*) are true at time t. Since $q(t) \notin \mathfrak{R}_l[Q_1, \ldots, q_r]$, relation (11) implies that an $x(t)$ as in (13) exists. Therefore, for any $y(t)$,

$$q(t+1) = \Psi[x(t), y(t), q(t)] \notin \{q_1, \ldots, q_r\} \cup \mathfrak{Q}_l[Q_1, \ldots, q_r] \cup \cup \mathfrak{P}_l[Q_1, \ldots, q_r].$$

If $u(t+1)$ is chosen using (16) (Case I), then (*a*) is true for $q(t+1)$ because of I(b); (*b*) is true because $B \in w(t+1)$. Finally (*c*) follows from (c) of the induction hypothesis, the fact that $q(t+1) \in B \subset Q_j$ for $j = 1, \ldots, i$, and the definition of B in I.

If $u(t+1) = [Q_1, \ldots, q_i]$ is chosen using (17) (Case II), then $q(t+1) \notin \notin \mathfrak{P}_l[Q_1, \ldots, q_r]$ and $q(t+1) \in Q_i$ implies by (11) that $q(t+1) \notin \mathfrak{R}_l[Q_1, \ldots, q_i]$; (*b*) is immediate from II. $q(t+1) \in Q_i$ implies $q(t+1) \in Q_j$ for $1 \leqq j \leqq i$. This together with the assumption that $Q_j \in w(t)$ implies $Q_j \in w(t+1)$ for $1 \leqq j \leqq i$, so that (*c*) holds for $t+1$. Q. E. D.

THEOREM 2.10. *If black uses T' and $q_0 \notin \mathfrak{R}_l[\,]$, then* $\lim(q(1)q(2)\dots) \notin \mathfrak{C}$.

Proof. Assume that black uses T' and $q_0 \notin \mathfrak{R}_l[\,]$. To prove that T' is a winning black strategy, we assume that $\lim(q(1)q(2)\dots) = Q' \in \mathfrak{C}$ and derive a contradiction.

Since $\lim(q(1)q(2)\dots) = Q'$, there is a t_1 such that

$$t \geqq t_1 \Rightarrow q(t) \in Q', \tag{18}$$
$$u \in Q' \Rightarrow (\forall a)(\exists t)[(t \geqq a) \,\&\, q(t) = u].$$

The relations (15), (18) and $Q' \in \mathfrak{C}$ imply that there is a $t_2 \geqq t_1$ such that

$$t \geqq t_2 \Rightarrow Q' \in (\mathfrak{C} \cap w(t)). \tag{19}$$

Define the quasi-order $\prec$ on chains $B_1 \supset \dots \supset B_p$, $A_1 \supset \dots \supset A_s$ ($p, s \geqq 0$) of members of $\mathfrak{C}$ by

$$[B_1, \dots, B_p] \prec [A_1, \dots, A_s] \equiv [p < s \,\&\, \bigwedge_{1 \leqq j \leqq p} (A_j = B_j)] \vee \bigvee_{1 \leqq i \leqq \min(p,s)} [(B_i \subset A_i) \,\&\, \bigwedge_{1 \leqq j < i} (A_j = B_j)]. \tag{20}$$

The *principal part* of $[Q_1, \dots, q_r]$, denoted by $\mathrm{PP}[Q_1, \dots, q_r]$, is $[Q_1, \dots, q_\phi]$, where $0 \leqq \phi \leqq r$ is the largest index such that $Q' \subseteq Q_\phi$. (Assume $Q_0 \supset Q'$. If $\phi = 0$, the principle part is $[\,]$.)

LEMMA 3. *If $t' > t > t_2$, then* $\mathrm{PP}(u(t)) \preccurlyeq \mathrm{PP}(u(t'))$.

Proof. Let $t > t_2$ and $\mathrm{PP}(u(t)) = [Q_1, \dots, q_p]$. If $p = 0$ or $r = 0$, then $\mathrm{PP}(u(t+1)) \succcurlyeq \mathrm{PP}(u(t)) = [\,]$. Assume that $r, p > 0$. $u(t+1)$ can be obtained from $u(t)$ by use of either (16) or (17). Since $t > t_2$, $q(t+1) \in Q'$, so that $q(t+1) \in Q_j$ for $1 \leqq j \leqq p$ (because $Q_j \supseteq Q'$). Hence if (17) is used, $\mathrm{PP}(u(t+1)) = \mathrm{PP}(u(t))$. If (16) is used, then the principal part of $u(t+1)$ differs from that of $u(t)$ only if there exist i, $0 \leqq i \leqq p$, and B, $B \supseteq Q'$, satisfying (a) and (b) of Case I. In this case $u(t+1) = [Q_1, \dots, q_i, B, B(q(t+1))]$, and so $\mathrm{PP}(u(t+1)) \succ \mathrm{PP}(u(t))$ $(Q_{i+1} \subset B \subset Q_i)$. Q.E.D.

Let $t_3 > t_2$ be such that, for $t > t_3$, $\mathrm{PP}(u(t))$ is fixed, say $[Q_1, \dots, q_p]$. There are two cases:

1. If $Q_p = Q'$, then by equality (13) for $t > t_3$, $q(t) \neq q_p \in Q'$, so that $\lim(q(1)q(2)\dots) \neq Q'$, which is a contradiction.

2. If $Q_p \supset Q'$, first note that for $t > t_3$, if $u(t) = [Q_1, \dots, q_p, Q_{p+1}, \dots]$, then $Q' \supset Q_{p+1}$. This is true because Q_{p+1} is not in $\mathrm{PP}(u(t))$ and both Q' and Q_{p+1} are in $w(t)$ (Q' by definition of t_3 and Q_{p+1} by Lemma 2(c)), which is a macrostate chain. Let $u(t) = [Q_1, \dots, q_p, Q_{p+1}, \dots]$, $t > t_3$. Because $Q_{p+1} \subset Q' = \lim(q(1)q(2)\dots)$, some $q(t') \notin Q_{p+1}$ is entered eventu-

ally. Then $u(t'+1) = [Q_1, \ldots, q_p]$ or $[Q_1, \ldots, q_p, Q'_{p+1}, q'_{p+1}]$, where $Q' \supset Q'_{p+1} \supset Q_{p+1}$. The second alternative can only occur finitely often without the first occurring, so that there is $t_4 > t_3$ such that $u(t_4) = [Q_1, \ldots, q_p]$. By Lemma 2(*a*), $q(t_4) \notin \mathfrak{R}_l[Q_1, \ldots, q_p]$, and so by (13) $q(t_4+1) \notin \mathfrak{Q}_l[Q_l, \ldots, q_p]$. But $Q' \subset Q_p$ and $Q' \in \mathfrak{C}$; therefore

$$\bigvee_{u \in Q'} u \notin \mathfrak{R}_l[Q_1, \ldots, q_p, Q', Q'(u)].$$

Since $Q' = \lim(q(1)\,q(2)\ldots)$, there is $t_5 > t_3$ such that $q(t_5) = u \in Q'$. But then (16) for $i = p$ will be applied, resulting in

$$\mathrm{PP}(u(t_5+1)) \succ [Q_1, \ldots, q_p],$$

which is a contradiction.

From Cases 1 and 2 it follows that $\lim(q(1)q(2)\ldots)$ cannot be in $\mathfrak{C}$. Q.E.D.

COROLLARY. *If $q_0 \notin \mathfrak{R}_l[\,]$, T' is a winning strategy for black.*

Theorem 2.8, the Fundamental Theorem on finite-state games, now follows immediately from the corollaries to Theorems 2.9 and 2.10.

II.11. Spectra of accessibility and distinguishability

The weight is the most important numerical parameter characterizing the memory of an operator; it is therefore natural to classify operators by weight (e.g., singling out operators with infinite weight). The analogous parameter in classification of automata is the number of states (which again may be infinite). However, in certain problems one needs a finer classification of operators (automata), which splits the set of operators (automata) of weight (number of states) μ into different levels of complexity (for every $\mu = 1,2,3,\ldots$ or ∞). This may be achieved by basing the classification not on a single numerical parameter, as before, but on a sequence of parameters; we shall call such sequences spectra. We shall define the "accessibility spectrum" and "distinguishability spectrum" first for operators, later for automata. They will then be used to prove a useful analogue of Theorem 2.5.

All operators and automata will be considered for fixed alphabets $X = \{x_1, \ldots, x_m\}$ and $Y = \{y_1, \ldots, y_n\}$; m and n will always denote the cardinalities of these alphabets. We shall confine ourselves to nonanticipatory operators, though suitable modifications adapt this approach to the general case, admitting operators with anticipation. The basic definitions will be given for nonanticipatory word operators; they are easily adapted

to (nonanticipatory) ω-word operators, since any nonanticipatory ω-word operator induces a nonanticipatory word operator (see Remark II in Section II.1).

Let T' and T'' be two distinguishable nonanticipatory operators (the adjective "nonanticipatory" will be omitted from now on). In other words, there exists an input word $x(1)x(2)\ldots x(t)$ which these operators transform into different output words. We shall say that this word distinguishes T' and T''. Call operators T' and T'' *k-distinguishable* ($k = 1,2,3,\ldots$) if they can be distinguished by a word of length at most k, *k-indistinguishable* otherwise. It is convenient to define k-distinguishability for $k = 0$ as well: any two operators T_1, T_2 are 0-indistinguishable (indistinguishable by the empty word). Obviously, if T' and T'' are k-distinguishable they are k'-distinguishable for any $k' > k$. k-indistinguishability will be denoted by $T' \approx_k T''$. Let $E(k)$ denote the maximal number (remember: for fixed alphabets X, Y!) of pairwise k-distinguishable operators. This number may be computed in the following trivial fashion. The transformation induced by T on the set of input words of length at most k into a set of output words (of the same length) is completely described by a finite tree v of height at most k, which is a truncation of the tree for T. Thus $E(k)$ cannot exceed the maximal number of pairwise k-distinguishable finite trees of height at most k. If two trees v_1, v_2 of this type are k-distinguishable, their extensions to complete trees v_1', v_2' of height k are also k-distinguishable. Thus $E(k)$ cannot exceed the number of complete trees of height k. Now a finite complete tree v contains exactly $m + m^2 + \ldots + m^k$ edges, which can be labeled by output letters in exactly $n^{m+m^2\ldots+m^k}$ ways. Thus the maximal number of pairwise k-distinguishable operators is exactly $n^{m+m^2+\ldots m^k}$.

The *distinguishability spectrum* of an operator T is the function $E_T(k)$ defined, for each k, as the maximal number of pairwise k-distinguishable residual operators of T.

The distinguishability spectrum is clearly a nondecreasing function such that

$$E_T(0) = 1,$$

$$E_T(k) \leqq n^{m+m^2+\ldots+m^k} \quad \text{for} \quad k > 0. \tag{*}$$

In particular, if the operator T has finite weight μ, the function $E_T(k)$ is bounded by the constant μ.

REMARKS. I. Recall that there is a one-to-one correspondence between the set of residual operators of T and the set of all input words (each word p corresponds to a unique operator T_p). One can thus define a (residual)

k-indistinguishability relation for input words:

$$p \approx_k r(T) \underset{df}{\equiv} T_p \approx_k T_r.$$

Here again it is instructive to compare these concepts with the previous concepts of indistinguishability and interchangeability (see pp. 36 and 104).

II. Let T be an everywhere defined operator with a two-letter output alphabet $\{0,1\}$ and let $\mathfrak{A}$ denote the language which T represents by one of its outputs, say 1. Then two words p_1 and p_2 are k-distinguishable if and only if there exists a word r of length k such that exactly one of the words $p_1 r, p_2 r$ belongs to $\mathfrak{A}$.

III. If T is an everywhere defined operator, k-indistinguishability is an equivalence relation over the set of all its residual operators, which induces a partition into k-indistinguishability classes, and $E_T(k)$ is precisely the index of this partition. But if T is a partial operator, then, in general, neither k-indistinguishability nor distinguishability is a transitive relation.

We now proceed to define the accessibility spectrum of an operator T. Fix some constant k ($k = 0, 1, 2, \ldots$) and consider all input words of length at most k and the corresponding residual operators. There are at most $1 + m + m^2 + \ldots + m^k < 2m^k$ such words (operators). The *accessibility spectrum* of an operator T is the function $D_T(k)$ defined as the maximal number of words of length at most k which are pairwise residually distinguishable (i.e., the corresponding residual operators are pairwise distinguishable). It is immediate that $D_T(k)$ is a nondecreasing function, and

$$D_T(k) \leqq \frac{m^{k+1} - 1}{m - 1}. \qquad (**)$$

In particular, if T has finite weight μ, the function $D_T(k)$ is bounded by the constant μ.

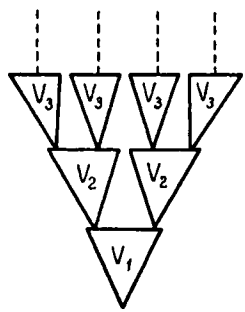


Figure 18

Let us consider the rate of growth of the functions E and D more closely.

1. The upper bounds (*) and (**) cannot be lowered. As already remarked, for the serial multiplication operator (T_7 of Section II.1) all input words are pairwise residually distinguishable. Therefore,

$$D_{T_7}(k) = \frac{m^{k+1} - 1}{m - 1} = \frac{4^{k+1} - 1}{3} \qquad (m = 4).$$

The simplest construction of an effective nonanticipatory operator T such that

$$E_T(k) = n^{m+m^2+\ldots+m^k}$$

is as follows. Let $v_1, v_2, v_3, \ldots$ be the sequence of all possible finite complete (m-branching) trees, labeled with the letters of the output alphabet. Construct the tree of the operator T as illustrated in Figure 18. Its root is the root of the tree v_1; at the final vertices of v_1 one grafts as many copies of v_2 as required; copies of v_3 are then grafted at the final vertices of the copies of v_2, and so on.

Note that the accessibility spectrum of this operator by no means assumes its maximal value $(m^{k+1} - 1)/(m - 1)$ (because, for any fixed s, all residual operators corresponding to final vertices of the copies of v_s are indistinguishable). Similarly, it is not hard to see that the distinguishability spectrum of the operator T_7 (which has the maximal accessibility spectrum) is much smaller than the upper bound $n^{m+m^2+\ldots+m^k}$; one can show that the rate of growth of $E_{T_7}(k)$ is only exponential.

2. *Truth-table operator.* Since $E(0) = D(0) = 1$ and the functions E and D are nondecreasing, the constant function 1 is the greatest lower bound for both functions. It is easy to see that this lower bound is attained (simultaneously, for both E and D) if and only if T is a truth-table operator.

3. *Operator with finite weight.* If the weight of the operator T is μ, then $\forall k[E_T(k) \leqq \mu]$, $\forall k[D_T(k) \leqq \mu]$.

For example, consider the constant operator producing the periodic ω-word

$$\underbrace{00\ldots0}_{\mu-1}\underbrace{100\ldots0}_{\mu-1}\underbrace{100\ldots0}_{\mu-1}1\ldots$$

with output alphabet $\{0, 1\}$ (since the operator is constant, the choice of input alphabet is immaterial in this situation). It is easily seen that this operator has weight μ, and

$$E(0) = 1, E(1) = 2, E(2) = 3, \ldots, E(\mu - 1) = E(\mu) = \ldots = \mu,$$

$$D(0) = 1, D(1) = 2, D(2) = 3, \ldots, D(\mu - 1) = D(\mu) = \ldots = \mu.$$

4. *Operator with infinite weight.* As already remarked, the operator T_2 of Section II.1 has infinite weight. Recall (Section II.2, p. 91) that two input words p_1 and p_2 are residually indistinguishable by T_2 (so that the residual operators T_{p_1}, T_{p_2} are k-indistinguishable for any k) when $n_1(p_1) - n_0(p_1) = n_1(p_2) - n_0(p_2)$. Suppose that $n_1(p) - n_0(p) > k$; then, for any word r of length at most k, the word pr contains more ones than zeros, i.e., the residual operator T_p transforms all such words into a word consisting solely of ones. Thus all residual operators T_p such that $n_1(p) - n_0(p) > k$ are pairwise k-indistinguishable. Similarly, one shows that all operators T_p such that $n_1(p) - n_0(p) \leq -k$ are pairwise indistinguishable. Hence it is clear that there are at most $2k + 2$ pairwise k-distinguishable residual operators, corresponding to the possible positions of $n_1(p) - n_0(p)$ in the interval $[-k, k+1]$. Now consider arbitrary numbers $i < j$ in this interval, and some word r such that $n_1(r) - n_0(r) = -i$. If $n_1(p_1) - n_0(p_1) = i$ and $n_1(p_2) - n_0(p_2) = j$, then $p_1 r$ contains zeros and ones in equal numbers, while there are more ones than zeros in $p_2 r$; therefore T_{p_1} and T_{p_2} are distinguishable by the word r. Thus

$$E_{T_2}(k) = 2k + 2.$$

On the other hand, for words of length at most k the difference $n_1(p) - n_0(p)$ can take exactly $2k + 1$ values, viz., $-k, -k+1, \ldots, 0, \ldots, k$; it follows that $D_{T_2}(k) = 2k + 1$.

II.12. Spectra of operators and of automata defining them

The definitions of distinguishability and accessibility spectra for automata are entirely analogous to their definitions for operators.

The *accessibility spectrum of an initialized automaton* $\langle \mathfrak{M}, q_0 \rangle$ is the function $D_{\langle \mathfrak{M}, q_0 \rangle}(k)$ defined as the number of states accessible from q_0 by words of length at most k (including the state q_0 itself, which is accessible via the empty word!). It is clear that the accessibility spectrum of an automaton $\mathfrak{M}$ is independent of the output function Φ, and it can be regarded as a characteristic of the corresponding outputless automaton $\langle Q, X, \Psi \rangle$. More precisely, $D_{\langle \mathfrak{M}, q_0 \rangle}(k)$ is a characteristic of the directed graph which remains when all labels (both input and output) are removed from the diagram of $\mathfrak{M}$ (retaining only the specification of the initial vertex). The vertices of this graph accessible from the initial vertex may be assigned ranks: rank 0—initial vertex; rank 1—all vertices different from the initial vertex accessible by a single edge from the initial vertex; in general,

rank $n+1$ is assigned to all vertices, not of rank $\leqq n$, which are endpoints of edges issuing from vertices of rank n. The set of all vertices of rank j constitutes the j-th *level*.

Let T be the operator realized by an initialized automaton $\langle \mathfrak{M}, q_0 \rangle$. We wish to determine the connection between the functions D_T and $D_{\langle \mathfrak{M}, q_0 \rangle}$. If $D_T(k) = \nu$, there are ν residually distinguishable words $p_1, \ldots, p_\nu$. Now these words take q_0 to ν pairwise distinguishable states of $\mathfrak{M}$; thus $D_T(k) \leqq \leqq D_{\langle \mathfrak{M}, q_0 \rangle}(k)$. Now suppose that $\langle \mathfrak{M}, q_0 \rangle$ is the *reduced* automaton realizing T. If $D_{\langle \mathfrak{M}, q_0 \rangle}(k) = \nu$, there exist ν states accessible from q_0 by words p_1, $p_2, \ldots, p^\nu$, respectively, of length at most k. These words are clearly pairwise residually distinguishable, and so $D_T(k) \geqq \nu$. But since always $D_T(k) \leqq \leqq D_{\langle \mathfrak{M}, q_0 \rangle}(k)$, it follows that in fact $D_T(k) = D_{\langle \mathfrak{M}, q_0 \rangle}(k)$. We have proved

THEOREM 2.11. *If the automaton $\langle \mathfrak{M}, q_0 \rangle$ realizes T, then $D_{\langle \mathfrak{M}, q_0 \rangle}(k) \geqq \geqq D_T(k)$. For every operator T, there is an automaton realizing it* (viz., the reduced initialized automaton) *such that $D_{\langle \mathfrak{M}, q_0 \rangle}(k) = D_T(k)$.*

By contrast, we shall define the distinguishability spectrum $E_{\mathfrak{M}}(k)$ for *non*initialized automata, since choice of an initial state has no effect on $E_{\mathfrak{M}}(k)$. Call states q_i, q_j of an automaton *k-indistinguishable* ($q_i \approx_k q_j(\mathfrak{M})$) if the operators $T(\mathfrak{M}, q_i)$ and $T(\mathfrak{M}, q_j)$ are k-indistinguishable; otherwise the states q_i, q_j are said to be *k-distinguishable*. The *distinguishability spectrum of a noninitialized automaton* $\mathfrak{M}$ is the function $E_{\mathfrak{M}}(k)$ defined for any natural k as the maximal number of pairwise k-distinguishable states of $\mathfrak{M}$. Clearly, the function $E_{\mathfrak{M}}(k)$ (like $E_T(k)$) is nondecreasing, and

$$E_{\mathfrak{M}}(k) \leqq n^{m+m^2+\ldots+m^k}.$$

Let $\langle \mathfrak{M}, q_0 \rangle$ be an automaton, realizing an operator T; let us compare $E_{\mathfrak{M}}(k)$ with $E_T(k)$. If $E_T(k) = \nu$, there are ν pairwise residually k-distinguishable words $p_1, \ldots, p_\nu$. But these words take q_0 to ν different pairwise k-distinguishable states. Consequently, $E_{\mathfrak{M}}(k) \geqq E_T(k)$. Now suppose that $\langle \mathfrak{N}, q_0 \rangle$ is the *reduced* automaton realizing the operator T, and $E_{\mathfrak{N}}(k) = \nu$. This automaton has ν pairwise k-distinguishable states. But then the ν words $p_1, \ldots, p_\nu$ that take q_0 to $q_1, \ldots, q_\nu$, respectively, are pairwise k-distinguishable relative to T. Therefore $E_T(k) \geqq E_{\mathfrak{M}}(k)$, and so in fact $E_T(k) = E_{\mathfrak{N}}(k)$. We have thus proved

THEOREM 2.12. *If the automaton $\mathfrak{M}$ realizes T, then $\forall k\,[E_{\mathfrak{M}}(k) \geqq E_T(k)]$. Every operator T is realizable by an automaton $\mathfrak{M}$ such that*

$$\forall k\,[E_{\mathfrak{M}}(k) = E_T(k)].$$

REMARK. Equivalent automata have the same distinguishability spectra but, in general, different accessibility spectra.

Theorems 2.11 and 2.12 are often used when one wishes to prove that some operator T is not realizable in automata of a certain type (usually described in structural terms). In so doing, one finds an upper bound for the distinguishability (accessibility) spectrum of automata of the required class H, and shows that this bound is definitely smaller than the corresponding spectrum of the operator T. We shall illustrate this approach for the class H of so-called *one-dimensional von Neumann automata*, which are infinite automata. This concept belongs to the structural theory and we shall describe it intuitively, without going into unnecessary formal detail.

An automaton of class H consists of an infinite set of cells, indexed by the natural numbers and arranged as illustrated in Figure 19. Each cell can be in one of the states of a set $\Pi = \{\pi_1, \pi_2, \ldots, \pi_\nu\}$, one of which is designated as the *passive state* (say π_ν). The first (left-most) cell has an input channel which receives letters from an input alphabet X, and an output channel, emitting letters of an output alphabet Y. Let $\pi(i, t)$ denote the state of cell i at time t. The first cell functions as follows:

$$\begin{aligned} \pi(1, t+1) &= \Psi_1[\pi(1,t), \pi(2,t), x(t)], \\ y(t) &= \Phi_1[\pi(1,t), \pi(2,t), x(t)], \end{aligned} \tag{*}$$

where Ψ_1, Φ_1 are suitable mappings of $\Pi \times \Pi \times X$ into Π and Y, respectively. The remaining cells have no input or output channels, and function as follows:

$$\pi(i, t+1) = \Psi_2[\pi(i-1, t), \pi(i, t), \pi(i+1, t)], \tag{**}$$

where Ψ_2 maps $\Pi \times \Pi \times \Pi$ into Π and satisfies the condition

$$\Psi_2[\pi_\nu, \pi_\nu, \pi_\nu] = \pi_\nu. \tag{***}$$

A sequence $\pi(1)\pi(2)\pi(3)\ldots\pi(t)\ldots$ of elements of Π is called an *admissible sequence* if, beginning from some t, all $\pi(t)$ coincide with π_ν, the passive state of the cell. The states of the automaton are defined to be all possible admissible sequences; the set of states is clearly denumerably infinite. Suppose that, at time t, the automaton is in state $\pi(i, t)$ $(i = 1, 2, 3, \ldots)$, and $x(t)$ is applied at the input. Then the next state $\pi(i, t+1)$ and the output $y(t)$ are determined by conditions (*) and (**). This definition is legitimate since, by condition (***), an admissible sequence $\pi(i, t)$ goes into an admissible sequence $\pi(i, t+1)$ $(i = 1, 2, 3, \ldots)$.

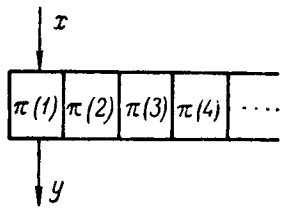


Figure 19

We shall now find an upper bound for the distinguishability spectrum of this automaton. Let two states $\pi = \pi(1)\pi(2)\ldots$ and $\pi' = \pi'(1)\pi'(2)\ldots$ be such that $\pi(1) = \pi'(1), \pi(2) = \pi'(2), \ldots, \pi(k) = \pi'(k)$. Then, as is easily seen, these states are k-indistinguishable. Indeed, while the automaton is receiving the input word $x(1)x(2)\ldots x(k)$, the cells to the right of the k-th cell have no effect; therefore, the output word will be the same, irrespective of whether the initial state was π or π'. Thus, the states π and π' are k-distinguishable only if the words $\pi(1)\pi(2)\ldots\pi(k)$ and $\pi'(1)\pi'(2)\ldots\pi'(k)$ are different. Since the number of different words of length k over the alphabet Π is v^k, our automaton $\mathfrak{M}$ satisfies the condition

$$\forall k[E_{\mathfrak{M}}(k) \leqq v^k],$$

that is to say, the distinguishability spectrum is dominated by an exponential function. Hence automata of this type cannot realize any operator whose distinguishability spectrum increases more rapidly than any exponential function, such as an operator whose spectrum is of order $n^{m+m^2+\ldots+m^k}$. The operator T_2 has a linear increasing distinguishability function $E_{T_2} = 2k + 2$ (see Section II.11), and so it *may* be realizable by an automaton of class H, though of course this argument does not guarantee this. It can indeed be shown that T_2 is realizable by an automaton of this type.

As mentioned above, the distinguishability and accessibility spectra are nondecreasing functions, i.e., the functions $\Delta E(k) = E(k+1) - E(k)$ and $\Delta D(k) = D(k+1) - D(k)$ are nonnegative. Our next goal is in some way to specify the rate of growth of the functions E and D for everywhere defined operators and automata. Since the spectra of an everywhere defined operator coincide with the spectra of the reduced automata realizing it, we may confine the discussion to automata. Call states q_i and q_j of an automaton $\mathfrak{M}$ *strictly k-distinguishable* ($k = 1,2,3,\ldots$) if they are k-distinguishable but $(k-1)$-indistinguishable (and therefore k'-indistinguishable for any $k' < k$). A state q is said to be *strictly k-accessible* ($k = 1,2,3,\ldots$) if it is k-accessible but k'-inaccessible for all $k' < k$. These definitions imply:

A. *If q is strictly $(k+1)$-accessible, then $\mathfrak{M}$ also has strictly k-accessible q'_i, q'_j which are strictly k-distinguishable.*

In fact, suppose that the word $x(1)x(2)\dots x(k+1)$ distinguishes q_i and q_j. Then q'_i and q'_j may be defined as the states to which the input letter $x(1)$ takes q_i and q_j, respectively. It is clear that the word $x(2)\dots x(k+1)$ distinguishes the states q'_i, q'_j. Were there a shorter word distinguishing them, we could add $x(1)$ in order to get a word of length less than $k+1$ distinguishing q_i and q_j, which contradicts their strict $(k+1)$-distinguishability.

B. *If q is strictly $(k+1)$-accessible, then $\mathfrak{M}$ also has a strictly k-accessible state; in other words, if an automaton graph contains a vertex of rank $k+1$, it also contains a vertex of rank k.*

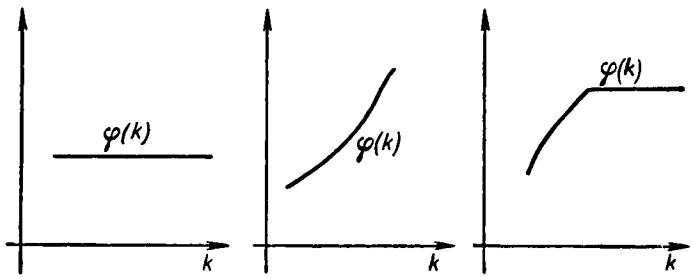


Figure 20

In fact, if $x(1)x(2)\dots x(k+1)$ is a word of minimal length taking q_0 to q, and $x(1)\dots x(k)$ takes q_0 to q', then q' is strictly k-accessible.

These assertions will be used to prove Theorem 2.13 below. Before stating the latter, let us consider those nondecreasing functions ϕ satisfying the condition

$$\forall k[\Delta\phi(k) = 0 \to \Delta\phi(k+1) = 0].$$

Any function of this type is either strictly increasing, a constant, or strictly increasing up to a maximum for some k_0, where it remains for all $k > k_0$ (Figure 20). In all cases considered hitherto the distinguishability and accessibility spectra have been nondecreasing functions of one of these three types. This is no accident, as evinced by the following

THEOREM 2.13. *For any automaton $\mathfrak{M}$ (everywhere defined operator T), the spectra satisfy the conditions*

$$\forall k[\Delta E(k) = 0 \to \Delta E(k+1) = 0],$$

$$\forall k[\Delta D(k) = 0 \to \Delta D(k+1) = 0].$$

Proof. It is obvious that if $\Delta E(k) \neq 0$ there exist strictly $(k+1)$-distinguishable states. Similarly, if $\Delta D(k) \neq 0$ there exists a strictly $(k+1)$-accessible state. The converse is also true. That the existence of a strictly $(k+1)$-accessible state implies that $\Delta D(k) \neq 0$ is obvious; it is however important to stress the reason why the existence of a pair of strictly $(k+1)$-distinguishable states q_i, q_j implies that $\Delta E(k) \neq 0$. The point is that for automata (and everywhere defined operators) k-indistinguishability is an equivalence of index $E(k)$, for each k. Thus the fact that q_i and q_j are strictly $(k+1)$-distinguishable implies that the k-indistinguishability class to which they belong splits into at least two $(k+1)$-indistinguishability classes, so that

$$E(k+1) \geqq E(k) + 1.$$

Thus, for any k, $\Delta E(k) = 0$ if and only if there are no strictly $(k+1)$-distinguishable states, while $\Delta D(k) = 0$ if and only if there are no strictly $(k+1)$-accessible states. But then propositions A and B formulated above show that

$$\Delta E(k) = 0 \rightarrow \Delta E(k+1) = 0 \qquad \text{and} \qquad \Delta D(k) = 0 \rightarrow \Delta D(k+1) = 0.$$

Q.E.D.

COROLLARY. *For any reduced automaton (everywhere defined operator) with infinite weight, the distinguishability and accessibility spectra are strictly increasing functions, and so their rate of growth is at least linear:*

$$E(k) \geqq k+1, \qquad D(k) \geqq k+1.$$

REMARK. In general, Theorem 2.13 is not valid for partial operators. For example, consider the distinguishability spectrum. The analogue of Proposition A (rephrased for input words) is valid for all operators, since its proof only uses the definition of k-indistinguishability. The trouble is that k-indistinguishability is no longer an equivalence relation, while this fact was essential for the proof of the theorem. However, it can be shown that the theorem remains valid for finite complete trees (multiple experiments). The proof is left to the reader.

II.13. Parameters of a finite automaton and its behavior

In this section we define several parameters which, together with the number of states of the automaton (weight of the operator), characterize the automaton and its behavior; we shall also establish upper and lower bounds for these parameters. These bounds will be found useful in solving the

synthesis and minimization problems and in other problems in the behavioral theory of automata.

Thus, let $\mathfrak{M}$ be a finite automaton (T an operator with finite weight). There certainly exists a number ν, depending on the automaton $\mathfrak{M}$ (operator T), such that any two distinguishable states (residual operators) must be ν-distinguishable.

The *degree of distinguishability* of an automaton $\mathfrak{M}$ (operator T) is the minimal number ν such that any two distinguishable states (residual operators) are ν-distinguishable. We denote the degree of distinguishability by $\rho(\mathfrak{M})$ or $\rho(T)$.

The *degree of accessibility* of an initialized automaton $\langle \mathfrak{M}, q_0 \rangle$ is the minimal ν such that any state accessible from q_0 is ν-accessible. In other words, the degree of accessibility $\delta(\mathfrak{M}, q_0)$ of an automaton $\langle \mathfrak{M}, q_0 \rangle$ is the highest rank of vertices in the corresponding graph, for fixed initial vertex q_0.

The *degree of accessibility of an operator* T (denoted by $\delta(T)$) is the maximal ν such that there exists an input word of length ν which is residually distinguishable from any shorter input word. Obviously, $\delta(T)$ coincides with the degree of accessibility of the reduced automaton realizing T.

These definitions and the proof of Theorem 2.13 directly imply that each of these parameters is precisely the value of the argument at which the corresponding spectrum levels off, i.e., the minimal k such that $\Delta E(k)$ or $\Delta D(k)$ vanishes. In view of this and the particular rate of growth of the spectra E and D, one can estimate the parameters comparing them with the weight of the operator or the number of states of the automaton.

THEOREM 2.14. *(I) Let $\mathfrak{M}$ be an automaton with reduced weight μ (T an operator with weight μ). Then its degree of distinguishability ρ satisfies the inequality*

$$\log_m \log_n \mu - 1 \leqq \rho \leqq \mu - 1.$$

(II) Let $\langle \mathfrak{M}, q_0 \rangle$ be an initialized automaton, with μ states accessible from q_0 (T an operator of weight μ). Then its degree of accessibility δ satisfies the inequality

$$\log_m \mu - 1 < \delta \leqq \mu - 1.$$

Proof. It follows directly from the definitions that $E(\rho) = \mu$, and moreover $E(k) = \mu$ for $k > \rho$. Similarly, the spectrum of accessibility levels off at δ: $D(\delta) = D(\delta + 1) = \ldots = \mu$.

Since the spectrum is a nondecreasing function, we have

$$1 = E(0) < E(1) < \ldots < E(\rho) = E(\rho + 1) = \ldots = \mu,$$

$$1 = D(0) < D(1) < \ldots < D(\delta) = D(\delta + 1) = \ldots = \mu.$$

Now, even for the slowest possible growth of $E(k)$, when $E(1) = 2, E(2) = 3, \ldots, E(\mu - 1) = \mu$, the value of ρ can never exceed $\mu - 1$. The same holds for the spectrum $D(k)$ and the degree δ. On the other hand, by previously established bounds on the spectra (see Section II.11), for the fastest possible growth we have

$$\mu = E(\rho) \leqq n^{m + m^2 + \ldots + m^\rho} < n^{2m^\rho} \qquad (*)$$

$$\mu = D(\delta) \leqq 1 + m + m^2 + \ldots + m^\delta < 2m^\delta. \qquad (**)$$

The lower bounds for ρ and δ now follow from (*) and (**) by taking logarithms. This completes the proof.

REMARK I. Clearly, the upper bounds in Theorem 2.14 are valid *a fortiori* when μ denotes the number of states of the automaton, instead of the reduced weight (see (I)) or number of accessible states (see (II)). However, it is evident that no meaningful lower bound can be based on the number of states of an automaton, since many states may turn out to be indistinguishable or inaccessible from the initial state.

We shall now show by means of examples that the bounds in Theorem 2.14 cannot be improved.

First, for any k there exists an operator of type T_5 (see Section II.1) with weight k, and then both the degree of distinguishability and the degree of accessibility are exactly $k - 1$.* An analogous statement holds for the reduced automata which realize these operators.

We shall now show how to construct reduced automata $\mathfrak{M}$, $\tilde{\mathfrak{M}}$ with arbitrarily large number of states μ, such that

$$\rho(\mathfrak{M}) \leqq \log_m \log_n(\mu + 1);$$

$$\delta(\tilde{\mathfrak{M}}) \leqq \log_m \mu.$$

This will show that the lower bounds are the best possible. For simplicity, take $m = n = 2$ (Figure 21). Fix an arbitrary natural number s and let $v = 2^s$. Construct a 2-branching tree v of height $s + v - 1$, labeling the edges with (output) letters 0 and 1 as follows. There are exactly 2^v words of length v over the alphabet $\{0, 1\}$; let these be $\pi_0, \pi_1, \ldots, \pi_{2^v - 1}$. They may be regarded as the binary expansions of the numbers from 0 to $2^v - 1$. The edges of the s lowest levels are labeled with zeros. Thus the level of rank s records the word π_0. The level of rank $s + 1$ has length $2v$; label its edges so that they generate the concatenation $\pi_1 \cdot \pi_2$ from left to right. The vertices q_0, q_1, q_2 will then be pairwise s-distinguishable. Now generate the

* The same may be said of the constant operator of Section II.11.

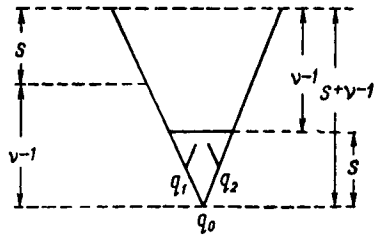


Figure 21

concatenation of $\pi_3, \pi_4, \pi_5, \pi_6$ in a similar way, along the level of rank $s + 2$; all seven vertices of rank at most 2 are then pairwise s-distinguishable. Continuing in this way, we fill up all the levels, forming 2^3 words, then $2^4, \ldots,$ and finally $2^{\nu-1}$ words. Thus we exhaust $1 + 2 + 2^2 + \ldots + 2^{\nu-1} = 2^\nu - 1$ words from the total number of 2^ν. The result is a finite comple tree v in which all vertices of rank at most $\nu - 1$ are pairwise s-distinguishable.

We shall estimate the weight μ and the degree of distinguishability ρ of the automaton $\mathfrak{M}$ obtained by contracting this tree v, and show that they satisfy the lower inequality.

Since the basis of the tree v contains all $2^\nu - 1$ vertices of rank $\leqq \nu - 1$, it follows that $\mu \leqq 2^\nu - 1$, i.e., $\nu \leqq \log_2(\mu + 1)$. The basis may contain vertices of higher rank, but then each such vertex α is surely $(s - 1)$-distinguishable from any other vertex of the basis, since α is the root of a subtree of v of height at most $s - 1$. That the vertices of rank at most $\nu - 1$ are s-distinguishable from other vertices in the basis follows, as shown above, from the way in which the edges were labeled. Thus $\rho = s$. Recalling that $s = \log_2 \nu$, $\nu \leqq \log_2(\mu + 1)$, we see that

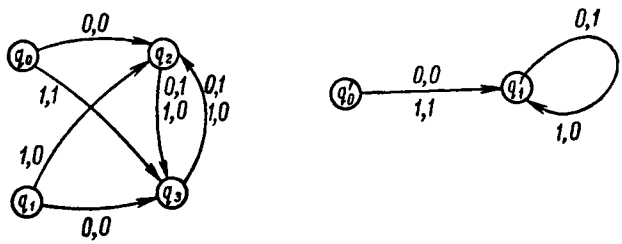
$$\rho \leqq \log_2 \log_2(\mu + 1).$$

If an initial state (vertex) q_0 is specified in this reduced automaton, then, obviously, $\delta \leqq s + \nu - 2$, i.e., $\delta \leqq \log_2(\mu + 1) + \log_2 \log_2(\mu + 1)$, so that asymptotically (as $\mu \to \infty$) δ does not exceed $\log_2 \mu$. However, this estimate may be improved (without using asymptotic arguments) by constructing another automaton $\tilde{\mathfrak{M}}$ on the basis of the same tree v. Let two edges issue from each vertex α of rank $s + \nu - 1$; assign them arbitrary output labels (say zeros throughout) and direct them to vertices of rank at most $\nu - 1$, $\psi_0(\alpha)$ and $\psi_1(\alpha)$, where the former is the endpoint of the edge with input label 0 and the latter that of the edge with input label 1. In so doing the following precautionary measure should be taken: if α and β are two

vertices of rank $v + s - 1$, the pair $\langle \psi_0(\alpha), \psi_1(\alpha) \rangle$ is different from the pair $\langle \psi_0(\beta), \psi_1(\beta) \rangle$. This can be done (and in more than one way), since the number 2^{v+s-1} of vertices of rank $v + s - 1$ is much smaller than the number $(2^v - 1)^2$ of pairs of vertices of rank at most $v - 1$ (recall that $s = \log_2 v$). For any diagram of an automaton $\tilde{\mathfrak{M}}$ constructed in this way, the vertices of the tree previously of rank $v + s - 1$ become pairwise distinguishable (to be precise, pairwise $(s + 1)$-distinguishable). We have thus constructed an automaton $\tilde{\mathfrak{M}}$ which has at least 2^{v+s-1} distinguishable states, and its degree of accessibility (from the initial vertex q_0) is at most $v + s - 1$; even the equivalent reduced automaton has weight $\mu \geqq 2^{v+s-1}$, while $\delta \leqq v + s - 1$, i.e., $\delta \leqq \log_2 \mu$.

REMARK II. In describing the minimization algorithm in Section II.3, we in fact based our considerations on the upper bound μ^2 for the degree of distinguishability ρ of an automaton with μ states. It should be clear from Remark I of this section that one can indeed employ the better estimate $\rho \leqq \mu - 1$. This somewhat simplifies the minimization algorithm.

For example, consider the initialized automaton with four states (so that $\rho \leqq 3$, $\delta \leqq 3$) defined by the diagram of Figure 22*a*. The states q_2 and q_3 are obviously 3-indistinguishable, but q_0 and q_2 are 3-distinguishable. After removing the state q_1, which is inaccessible from q_9, and merging indistinguishable states, we get the automaton of Figure 22*b*.



a *b*

Figure 22

In cases where additional information gives an even better upper bound for ρ, the minimization procedure may be substantially simplified (and also since, as the preceding examples show, the degrees of accessibility and distinguishability may actually be far smaller than the number of states, being of the order of the logarithm and repeated logarithm, respectively, of this number).* When a good upper bound for the degree of distinguish-

* In Chapter V we shall show that this is indeed so in "almost all" cases.

ability is lacking, the following procedure for determining the indistinguishable states of the automaton is sometimes convenient. Using multiple experiments of length 1, divide all the states into 1-indistinguishability classes. Assume that the set of states has already been divided into k-indistinguishability classes. Then states q_i, q_j are $(k+1)$-indistinguishable if and only if

1) q_i, q_j are k-indistinguishable;

2) for every x, the states $\Psi(q_i, x)$ and $\Psi(q_j, x)$ are k-indistinguishable.

The procedure ends when the i-indistinguishability classes first coincide with the $(i+1)$-indistinguishability classes. When this happens, the degree of indistinguishability is i and every i-indistinguishability class is an indistinguishability class.

Given an operator T and a multiple experiment v of length h.* We shall say that the operator T and the experiment v are *compatible* if T is an extension of the partial operator v to an everywhere defined operator. It is clear that every operator has an infinite set of compatible experiments (with increasing lengths), while every experiment has an infinite set of pairwise distinguishable operators of finite weight which are compatible with it. We shall say that an experiment v *reconstructs* an operator T if v is compatible with T and no operator T' distinguishable from T and of weight at most that of T is compatible with v. These definitions immediately imply that if there exists an experiment of length h reconstructing the operator T, then

(I) this experiment is unique;

(II) it reconstructs the operator T alone;

(III) for any $h' > h$, there exist experiments of length h' that reconstruct T.

Moreover, we have the following assertion: *For any operator T of finite weight μ there exists at least one experiment reconstructing it.* In fact, there is an infinite number of such experiments (of increasing lengths!). Indeed, the set $\tau = \{T_i\}$ of pairwise distinguishable operators of weight at most μ is finite. Let v_i be the length of the shortest word distinguishing T from $T_i \in \tau$; then T_i surely has a reconstructing experiment of length $v = \max v_i$. Moreover, this v is the minimal h such that T can be reconstructed by an experiment of length h. On the other hand, our definitions do not ensure that any experiment reconstructs some operator.

For brevity, we shall call an operator *h-reconstructible* if it has a reconstructing experiment of length h.

We now introduce an important parameter: the *degree of reconstructibility*

* For the definition, see p. 98.

of an operator T (notation: $B(T)$) is the minimal number h such that T is h-reconstructible.

Our terminology, phrased for operators, carries over in a natural way to automata. An experiment v is *compatible with an automaton* $\langle \mathfrak{M}, q_0 \rangle$ if it is compatible with the operator $T(\mathfrak{M}, q_0)$. It *reconstructs the behavior of the automaton* $\langle \mathfrak{M}, q_0 \rangle$ if it reconstructs the operator $T(\mathfrak{M}, q_0)$. Correspondingly, the *degree of reconstructibility of the automaton* $\langle \mathfrak{M}, q_0 \rangle$ (notation: $B(\mathfrak{M}, q_0)$) is the degree of reconstructibility of the operator $T(\mathfrak{M}, q_0)$.

These definitions have the following consequences.

A. *Any experiment reconstructing an operator T of weight μ has weight μ and admits a unique extension of the same weight* (this extension is precisely the operator T).

B. *If a complete finite tree of weight μ has a unique extension T of weight μ, it is a reconstructing experiment for T.*

We shall prove A. Assume that T has weight μ and is reconstructed by an experiment v_h of weight μ'. It is then obvious that $\mu' \leqq \mu$. However it is easy to prove that the strict inequality $\mu' < \mu$ leads to a contradiction. In fact, if $\mu' < \mu$, then v_h may be extended to an operator T' of the same weight μ' (see Section II.4). This operator has weight smaller than that of T (and so T' and T are distinguishable), and is also compatible with the experiment v_h. But this means that v_h is not a reconstructing experiment for T. Thus μ' must be equal to μ, so that the weight of v_h is μ. The fact that v_h has no extension other than T of the same weight also follows directly from the definition of a reconstructing experiment. For the definition requires that no operator T' distinguishable from T and having weight at most that of T can be compatible with v_h.

Assertions A and B enable us to rephrase Theorem 2.5 (Section II.4) as follows:

THEOREM 2.15. *There is an algorithm which, given any finite tree v, (a) determines whether v reconstructs some operator; if so, (b) it constructs the reduced automaton realizing this operator.*

We now wish to estimate the degree of reconstructibility of an operator T in terms of its other parameters (weight, degree of distinguishability, degree of accessibility). We have the following

THEOREM 2.16. *The degree of reconstructibility $B(T)$ of an operator T satisfies the inequalities*

$$B(T) \leqq \rho(T) + \delta(T) + 1;$$
$$B(T) \leqq 2\mu(T) - 1.$$

Proof. The second inequality is a direct consequence of the first (see Theorem 2.14). It will therefore suffice to prove that an experiment v of length $\rho(T) + \delta(T) + 1$ admits a unique extension of the same weight (this extension coincides with the operator T). Consider the infinite tree defined by T, with weight μ; consider the initial finite tree v, of height $\rho(T) + \delta(T) + 1$. The definition of $\delta(T)$ implies that the vertices on levels $0, 1, \ldots, \delta$ already include μ pairwise distinguishable vertices. It follows from the definition of $\rho(T)$ that the distinguishability of these μ vertices is already determined by the finite tree v.

Now apply the contraction procedure of Theorem 2.5 (Section II.4), with D consisting of μ vertices (in the tree defining the operator T) on levels not exceeding the δ-th. Then any vertex of $\mathfrak{V}$ has rank at most $\delta + 1$. Thus, every vertex of $\mathfrak{V}$ is the root of a branch in v whose height is at least $\rho(T)$, and is therefore indistinguishable only from a single vertex of D. As already indicated in Theorem 2.5, this shows that there is exactly one extension of the same weight, proving the theorem.

COROLLARY (EXTRAPOLATION). *If two operators whose weights are at most μ are $(2\mu - 1)$-indistinguishable, they are indistinguishable.*

The definition of degree of reconstructibility for an initialized automaton is analogous: The *degree of reconstructibility of an initialized automaton* $\langle \mathfrak{M}, q_0 \rangle$ (notation: $B(\mathfrak{M}, q_0)$) is the degree of reconstructibility of the operator $T(\mathfrak{M}, q_0)$. It follows that Theorem 2.16 may be rephrased as follows:

THEOREM 2.16′. *The degree of reconstructibility of an automaton $\langle \mathfrak{M}, q_0 \rangle$ satisfies the inequalities*

$$B(\mathfrak{M}, q_0) \leqq \rho(\mathfrak{M}) + \delta(\mathfrak{M}, q_0) + 1; \qquad B(\mathfrak{M}, q_0) \leqq 2\mu - 1,$$

where μ is the weight (number of states) of the automaton $\langle \mathfrak{M}, q \rangle$.

Supplementary material, problems

I. Let T be an operator of weight k applied to an input ω-word $x(1)x(2)x(3)\ldots$ with period of length v, and let $v_0, v_1, \ldots, v_n$ be all the divisors of v. Then the corresponding output sequence has a period whose length may only be one of the numbers $v_0, 2v_0, \ldots, kv_0, v_1, 2v_1, \ldots, kv_1, \ldots, v_n, 2v_n, \ldots, kv_n$ (Trakhtenbrot [7]).

II. In studying ω-languages and ω-word operators it is sometimes convenient to employ a geometrical approach: each ω-word is interpreted as

a point in Baire space, each operator as a mapping defined in this space. Fix an alphabet X; regard the ω-words over X as points of a metric space, defining the distance between two ω-words $x' = x'(1)x'(2)\ldots, x'' = x''(1)x''(2)\ldots$ by

$$\rho(x', x'') = \frac{1}{r},$$

where r is the smallest t such that $x'(t) \neq x''(t)$.

It is easy to see that the space thus defined is homeomorphic to the Cantor set (perfect and nowhere dense) and is therefore compact. In this sense an ω-word operator is simply a mapping of the Cantor set into itself.

Using this interpretation, one can speak of continuous operators, closed ω-languages, and so on. We shall employ the usual terminology and notation for the standard classes of sets: open sets (G), closed sets (F), denumerable intersections of open sets (G_δ), denumerable unions of closed sets (F_σ), denumerable unions of G_δ-sets—$G_{\delta\sigma}$-sets, denumerable intersections of F_σ-sets—$F_{\sigma\delta}$-sets, and so on.

Show that any ω-language representable in a finite automaton is both a $G_{\delta\sigma}$-set and a $F_{\sigma\delta}$-set. Give an example of an ω-language, representable in a finite automaton, which is neither an F_σ-set nor a G_δ-set. Show that an operator has a finite anticipation if and only if it is continuous.

Show that any continuous operator with finite weight also has bounded anticipation (Trakhtenbrot [60]).*

III. In investigations relating to coding theory (especially with regard to devices for nonuniform coding [41]), it is often desirable to extend the concept of an automaton, relaxing the condition that input and output words have the same length. One then assumes that for any pair $q \in Q, a \in X$ the value $\Phi(q,a)$ of the output function Φ is a certain (possibly empty) word over the output alphabet. If we specify an initial state q_0 in a generalized automaton $\mathfrak{M} = \langle Q, X, Y, \Phi, \Psi\rangle$ of this kind, the initialized automaton $\langle \mathfrak{M}, q_0\rangle$ will define a word operator, as usual; however, the length of the output word need not be equal to that of the input word (it may even be empty). If no value of the function Φ is the empty word, the length of the output word is at least that of the input word; one can then associate, in a natural way, a nonanticipatory ω-word operator with the initialized automaton $\langle \mathfrak{M}, q_0\rangle$, though the weight of this operator may be infinite even if the automaton $\mathfrak{M}$ is finite. Give examples of this situation.

* *Translator's note*: This problem was first solved in a stronger form (Boolean algebra over G_δ) by Büchi and Landweber [129].

IV. Let $v(m,n,h)$ denote the set of all finite trees of height h over a fixed input alphabet of $m \geqq 2$ letters and a fixed output alphabet of $n \geqq 2$ letters. Any tree of this type defines a partial nonanticipatory operator, defined over all input words of length at most h. Let $k(m,n,h)$ be the maximal weight of the trees in $v(m,n,h)$ (see Section II.4). The asymptotic behavior of the function $k(m,n,h)$ for fixed (though arbitrary) m, n, as $h \to \infty$, may be described in terms of the parameter

$$c = \frac{\log_m n}{(m-1)^2}.$$

1) As $h \to \infty$,

$$k(m,n,h) \leqq \frac{c \cdot m^{h+2}}{h}[1 + o(1)].$$

On the other hand, there exists a sequence $h_1 < h_2 < h_3 < \ldots$ such that

$$k(m,n,h_i) \geqq \frac{c \cdot m^{h_i+2}}{h_i}[1 - o(1)].$$

2) As $h \to \infty$,

$$k(m,n,h) \geqq \frac{c \cdot m^{h+1}}{h}[1 - o(1)].$$

On the other hand, there exists a sequence $h_1 < h_2 < h_3 < \ldots$ such that

$$k(m,n,h_i) \leqq \frac{c \cdot m^{h_i+1}}{h_i}[1 + o(1)].$$

(Trakhtenbrot [62]).

Inequalities 1) and 2) give upper and lower bounds, respectively, for $k(m,n,h)$, and describe their accuracy. More precise results have been obtained by Korshunov and Grinberg [25], who establish the asymptotic behavior of $k(m,n,h)$ for any h.

V. Let us say that a word operator T(automaton $\mathfrak{M}$) *generates a language* $\mathfrak{A}$ *at its output* (or, briefly, *enumerates* $\mathfrak{A}$), if $\mathfrak{A}$ is the set of words over the output alphabet into which the operator T(automaton $\mathfrak{M}$) transforms the set of all input words (Trakhtenbrot [58], Korpelevich [34]).

Any language which is enumerable by a finite automaton is also representable in a suitable finite automaton. Under what conditions is a language represented by a finite automaton also enumerable by a finite automaton?

If the enumerating automaton has n states, the corresponding accepting [representing] automaton has at most 2^n states, and this bound cannot be improved (Lupanov [42], Ershov [29]).

If a language $\mathfrak{A}$ is enumerated by a Moore automaton with n states, it is representable in a suitable automaton with $2^{\mu(n)}$ states, where $\mu(n) \sim n/2$ (this bound is best possible—Grinberg [26]).

VI. Consider the following alternative definition for the acceptance of an ω-language $\mathfrak{A}$ over an alphabet X by a nonanticipatory operator T with input alphabet $X \times \{0,1\}$ and output alphabet Y. T accepts $\mathfrak{A}$ if a suitable subalphabet $Y' \subset Y$ satisfies the following condition: $x(1)x(2)\ldots \in \mathfrak{A}$ if and only if there exists an ω-word $\sigma(1)\sigma(2)\ldots$ over the alphabet $\{0, 1\}$ such that T transforms $\langle x(1)\sigma(1)\rangle\ \langle x(2)\sigma(2)\rangle\ \langle x(3)\sigma(3)\rangle\ldots$ into an ω-word containing infinitely many occurrences of letters from Y'. Show that an ω-language is representable in a finite automaton if and only if it is acceptable in this new sense by a nonanticipatory operator with finite weight.

VII. Show that if the graph of an operator T is representable in a finite automaton (see Section II.5), the operator T has finite memory (T may also be an anticipatory operator; see Section II.5) (Trakhtenbrot [60]).

VIII. Given an ω-language $\mathfrak{A}$ over the alphabet $X \times Y$ and a finite-state operator T_0 transforming ω-words over an alphabet $X \times U$ into ω-words over Y (i.e., T_0 is an operator $y = T_0(x,u)$ mapping pairs of ω-words x, u into an ω-word y). Call T_0 a *general A-solution* of the ω-language $\mathfrak{A}$ if:

(I) for any finite-state operator $y = T(x)$ which uniformizes the ω-language $\mathfrak{A}$, there exists a finite-state operator $u = \tilde{T}(x)$ such that $T(x) = T_0(x, \tilde{T}(x))$;

(II) for any finite-state operator $u = \tilde{T}(x)$, the operator $T_0(x, \tilde{T}(x))$ (which is obviously finite-state) uniformizes $\mathfrak{A}$.

The following assertion is true (Trakhtenbrot [60]):

Let $\mathfrak{A}$ be a closed (in the sense of Problem II) ω-language which is representable in a finite automaton. Then one of the following alternatives must hold: a) $\mathfrak{A}$ is not uniformizable by a nonanticipatory operator, b) $\mathfrak{A}$ has a general A-solution.

Given an automaton representing $\mathfrak{A}$, one can effectively determine whether $\mathfrak{A}$ is closed, and, if so, whether it has a general A-solution; if it has, a general A-solution is effectively constructible.*

IX. The concept of a language enumerated (generated) by a generalized automaton (in the sense of Problem III) is analogous to the usual automaton concept: this language consists of all nonempty words into which the

* *Translator's note*: In fact, it is shown in Landweber [130] that it can be decided whether $\mathfrak{A}$ is closed, G_δ or $G_{\delta\sigma} \cap F_{\sigma\delta}$.

generalized automaton transforms all possible input words. Show that any language enumerated by a finite generalized automaton is also enumerable by an ordinary finite automaton, which can be effectively constructed on the basis of the given automaton. Estimate the number of states of this ordinary automaton in terms of the number of states of the generalized automaton.

X. Following Büchi [73], let us introduce the concepts of a special ω-word, special automaton, and its behavior. We consider only the so-called *special* alphabets, which contain (possibly together with other letters) special designated symbols 0 and 1. An ω-word is said to be *special* if it contains at most finitely many occurrences of letters different from 0. An initialized Moor automaton with input alphabet $\{0,1\}$ is said to be *special* if, for any state q, $\lambda(q) = \lambda[\Psi(q, 0)]$ (i.e., output letters may change only on application of an input symbol other than 0). Thus, a special automaton transforms a special input ω-word ξ either into a special ω-word, or into an ω-word containing at most finitely many occurrences of letters other than 1. In the first case we say that the automaton *accepts* ξ, in the second — that it *rejects* ξ. The *behavior* of a special automaton is the ω-language consisting of all ω-words accepted by the automaton. Every word $x(1)x(2)\ldots x(t)$ may be encoded as a special ω-word $x(1)x(2)\ldots x(t)1000\ldots$ Show that, for any finite-state language A, there exists a finite special automaton $\mathfrak{M}_A$ whose behavior coincides with the set of codes of the words of A. Describe an algorithm which, given an outputless automaton $\langle \mathfrak{N}, q_0, Q' \rangle$, representing the language A, constructs a suitable special automaton $\mathfrak{M}_A$.

XI. The cartesian product of m special alphabets becomes a special alphabet if one allots the roles of 0 and 1 to the m-tuple of zeros and the m-tuple of ones, respectively. If the special ω-words $\xi_1, \xi_2, \ldots, \xi_m$ are codes (in the sense of the preceding paragraph) of words $a_1, a_2, \ldots, a_m$, then the coupling of these ω-words (which is obviously a special ω-word) is regarded as the code of the m-tuple $a_1, a_2, \ldots, a_m$. Following Büchi, let us call a set $\mathfrak{A}$ of m-tuples of words $a_1, a_2, \ldots, a_m$ a *finite-state set* if there exists a finite special automaton whose behavior is precisely the set of couplings of the codes of the m-tuples of $\mathfrak{A}$ (note that the lengths of the words $a_1, a_2, \ldots, a_m$ may be different). Since any set of m-tuples of words over an alphabet induces an m-ary predicate over the alphabet, we can also define finite-state predicates whose arguments are words. Show that, for any finite outputless automaton $\mathfrak{M} = \langle Q, X, \Psi \rangle$, the binary relation (binary predicate) of indistinguishability relative to $\mathfrak{M}$ is a finite-state predicate.

Show that the ternary predicate $P(x, y, z)$, defined to hold if and only if z

is the binary expansion of the sum of the numbers with binary expansions x, y, is finite-state.

Give other examples of finite-state m-ary predicates.

Notes

A rudimentary classification of operators in terms of anticipation and weight (Sections II.1, II.2) is already implicit in the paper of Burks and Wright [79]. A more thorough examination of the problem, based on a clear differentiation between properties related to anticipation and properties related to weight, was given in [56], and later in [60]. To this end, wide use was made in [7] of description of nonanticipatory operators by trees. The concepts and theorems relating to minimization of operators and comparison between the weight of an operator and the weight of an automaton realizing it (Sections II.3, II.4) have been introduced independently by many authors. A rigorous and detailed exposition, including the case of partial operators and automata, may be found in Glushkov's monograph [6].

The problem of the conditions under which a set of words is the graph of a finite-state operator was first formulated and solved in [58], and later in [54]. A more general formulation of the uniformization problem (Section II.5) first appeared, apparently, in Church [84], who also solved the problem for various special cases.

Numerous examples of properties of finite automata for which there exist no decision algorithms may be found in Rabin–Scott [114] and Burks [78]. The game-theoretic interpretation of the uniformization problem is due to McNaughton [105]. In Landweber [127] and Büchi and Landweber [128] this suggested interpretation is used to prove the fundamental theorem on strategies for these games. [McNaughton's original proof in [105] was erroneous—*Trans.*] In essence, special cases of the uniformization problem had been attacked in the earlier papers of Church [84] and Trakhtenbrot [59], but, in the absence of a convenient interpretation like McNaughton's, the formulation and proofs of their results were considerably less lucid.

Parameters for the behavior of automata and operators, as considered in Sections II.11 to II.13, were introduced independently by Moore [108] and Trakhtenbrot [56], who established bounds for these parameters. Moore's proof of the theorem was published first. Then Trakhtenbrot advanced the conjecture that, in some suitable sense, the degree of reconstructibility of "almost all" automata is far smaller than the maximum $(2k - 1)$ and is of

the same order as $\log k$. A rigorous proof of this conjecture was given by Barzdin' and Korshunov [17, 37].

The concepts of the distinguishability and accessibility spectra were in fact clear and even used (implicitly) in the latter papers, but the present book contains their first explicit definition. Even earlier, in the theory of growing automata, Barzdin' [16] defined a related spectrum concept, which he called capacity. Barzdin' was the first to prove, by comparison of the spectra of operators and automata, that operators of certain classes cannot be realized by automata of a given class (see Section II.12). Later, the same approach was utilized (independently) by Cole [86].

CHAPTER III

METALANGUAGES

III.1. Preliminary examples and problems

The preceding chapters have rigorously defined the intuitive concept of the behavior of automata: for outputless automata, in terms of representation of languages and ω-languages, for automata with output—in terms of realization of operators. Throughout the sequel we shall use the term "synthesis of automata" to mean the functional synthesis of finite automata. The unqualified term "automaton" will be used for automaton with output, anchored or macroanchored outputless automaton, etc., whenever there is no danger of confusion.

The situation to be studied in this chapter resembles the dialogue between a "client," who presents his requirements of the behavior of the automaton, and a "designer," whose task it is to construct a suitable automaton. We shall illustrate this by a few examples. We first consider several "requirements" phrased in terms of "input/output," assuming throughout that the output alphabet is $Y = \{0,1\}$ and the input alphabet $X = X_1 \times X_2$, where $X_1 = X_2 = \{0,1\}$.

1. *Client.* The output ω-word $y = y(1)y(2)\ldots y(t)\ldots$ is the sum of the input ω-words $x_1 = x_1(1)\ldots x_1(t)\ldots$ and $x_2 = x_2(1)\ldots x_2(t)\ldots$ (i.e., the projections of the input ω-word x onto X_1 and X_2, respectively), in the following sense. For any t, the output symbol $y(t)$ is the t-th binary digit from the right in the sum of the t-digit natural numbers with binary expansions $x_1(t)x_1(t-1)\ldots x_1(1)$ and $x_2(t)x_2(t-1)\ldots x_2(1)$.

Designer. The client has described a nonanticipatory operator with finite weight (the operator T_6 in Section II.1). The appropriate reduced initialized automaton (serial binary adder) has two states, which store the carry of zeros and ones, respectively. The automaton is described in Table 4 and Figure 23.

2. *Client.* For odd t, the t-th symbol of the output ω-word is $y(t) = x_1(t+1) \oplus x_2(t+1)$ (where $\oplus$ denotes addition mod 2); for even t, $y(t)$ is arbitrary.

Designer. There are infinitely many operators satisfying the client's

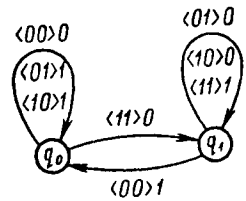


Figure 23

specifications, but they are all anticipatory; such operators are generally not realizable by (even infinite!) automata.

3. *Client.* The output ω-word $y = y(1)y(2)\ldots y(t)\ldots$ is the product of the input ω-words $x_1 = x_1(1)x_1(2)\ldots x_1(t)\ldots$ and $x_2 = x_2(1)x_2(2)\ldots x_2(t)\ldots$ in the following sense: $y(t)$ is the t-th binary digit from the right in the product of the t-digit natural numbers with binary expansion $x_1(t)\ldots x_1(1)$ and $x_2(t)\ldots x_2(1)$.

Designer. As in Case 1, the client has described a nonanticipatory operator, but here with infinite weight (the operator T_7 in Section II.1). Consequently, it is not realizable by a finite automaton.*

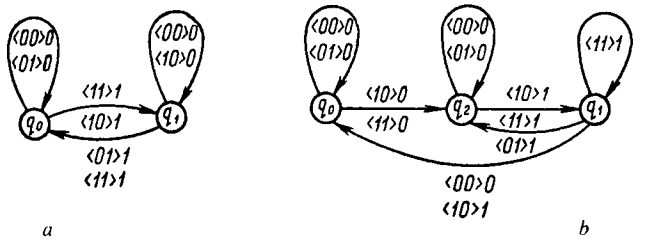
4. *Client.* The output ω-word y is to contain infinitely many ones if and only if each of the input words x_1 and x_2 contains infinitely many ones.

Designer. Here there are infinitely many nonanticipatory operators of finite weight satisfying the client's requirements. Figure 24*a* and Table 9 define an automaton with two states (binary-input flip-flop) which realizes one of these operators. The main feature of this automaton is that state q_1 can change to state q_0 only when $x_2(t) = 1$, and q_0 to q_1 only when $x_1(t) = 1$ (these conditions are also sufficient). Since ones are generated at the output precisely when these changes of state occur, the automaton meets the client's requirements. Another suitable automaton is illustrated in Figure 24*b* and Table 10. It does not reduce to the preceding operator by minimization but realizes an essentially different operator.

In the following examples the "requirements" are formulated solely in terms of input. The client specifies a certain language (ω-language) and requires an anchored (macroanchored) finite automaton representing it.

5. *Client.* The language consists of all words over the alphabet $\{0,1\}$ with an even number of ones.

* If necessary, the designer could have provided additional information: the required operator T is realizable by a one-dimensional von Neumann automaton (see Section II.12), which may be called a serial binary multiplier.



a *b*

Figure 24

TABLE 9

x_1x_2 \ q	q_0	q_1
00	$0q_0$	$0q_1$
01	$0q_0$	$1q_0$
10	$1q_1$	$0q_1$
11	$1q_1$	$1q_0$

TABLE 10

x_1x_2 \ q	q_0	q_1	q_2
00	$0q_0$	$0q_0$	$0q_2$
01	$0q_0$	$1q_2$	$0q_2$
10	$0q_2$	$1q_0$	$1q_1$
11	$0q_2$	$1q_1$	$1q_1$

Designer. The client has described a language with finite interchangeability index; a representing automaton is illustrated in Figure 8.

6. *Client.* The language consists of all words over $\{0,1\}$ containing more ones than zeros.

Designer. This language is not representable in a finite automaton (see the operator T_2 in Section II.1).

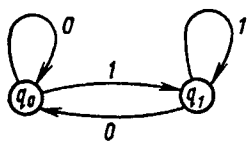


Figure 25

7. *Client.* The ω-language consists of all ω-words over the alphabet $\{0,1\}$ such that $0.x(1)x(2)\ldots$ is the (infinite) binary expansion of an irrational number.

Designer. This language cannot be represented in a finite automaton, since it contains no periodic ω-word (see Remark to Theorem 1.5, Section I.4).

8. *Client.* The ω-language consists of all ω-words over the alphabet $\{0, 1\}$ containing infinitely many ones.

Designer. This language is represented in the automaton $\mathfrak{M}$ of Figure 25, with initial state q_0 and limit macrostates $\{q_0,q_1\}$ and $\{q_1\}$.

III.2. Discussion of the examples. Statement of the problem

It will be useful to accompany our detailed formulation of the various components of the synthesis problem with a renewed discussion of the above examples. Note first that the "requirement" is not always met (Examples 2, 3, 6, 7). Thus, in solving the general synthesis problem, the first question to be settled is:

A. (EXISTENCE PROBLEM). *Does there exist a finite-state operator (language, ω-language) which satisfies the customer's conditions?*

Now, Examples 1, 3, 5, 6, 7, 8 differ from Examples 2 and 4 in that the former involve a description of a specific operator (language, ω-language); thus Problem A becomes:

A'. *Is the operator (language, ω-language) realizable (representable) by a finite automaton?*

If the answer to Problem A is negative, one has a (negative!) solution to the synthesis problem itself. Otherwise, investigation of the problem proceeds. For example, the following problem arises naturally:

B. (UNIQUENESS PROBLEM). *Is the operator (language, ω-language) satisfying the given conditions unique?*

Clearly, if Problem A' is relevant this question does not arise. Whatever the case, the central problem when the answer to Problem A is positive is as follows:

C. *Construct an automaton realizing (representing) an operator (language, ω-language) which satisfies the given conditions.*

That the solution to this problem need not be unique is reflected in our use of the indefinite article "an" for both operator (language, ω-language) and automaton. The non-uniqueness of the automaton may usually be avoided by requiring, in addition, that the number of its states be minimal. Using a minimization algorithm (Sections II.3 to II.13), one can always devise a unique (up to isomorphism!) reduced automaton equivalent to the original automaton. Essentially, then, the non-uniqueness in Problem C applies only to cases in which the answer to Problem B is negative. In any case, this does not affect the case of A', in which the uniqueness problem is irrelevant (Examples 1, 3, 5, 6, 7, 8) and Problem C may be rephrased as follows:

C′. *Construct an automaton realizing (representing) the operator (language, ω-language) defined by the given conditions.*

The situation is analogous when Problem B has a positive solution. However, when the solution to B is negative, one usually demands of the selected operator (language, ω-language) that it be realizable in an automaton with minimal number of states (that the operator have minimal weight). This is the case in Example 4, since it is clear that no truth-table operator satisfies the client's demands; the automaton constructed by the designer has only two states. Nevertheless, one must remember that there may be several operators with the same minimal weight.

In the sequel, we shall concentrate on Problems A and C for operators, and Problems A′ and C′ for languages (ω-languages). A solution to these problems will consist, first, of an algorithm which, given the client's conditions, will always provide an answer to Problem A (or A′), and in case of a positive answer, carry out the construction C (or C′). Clearly, our statement of the problem of automaton construction assumes prior knowledge of two classes of objects: the class of initial data, to which the algorithm is applied, and the class of resultant data, which the algorithm generates. We call these classes *metalanguages*,* and the structural objects of which they consist *formulas*. In the case at hand, we have already discussed the second class (the *designer's metalanguage*)—this is the class of finite-automaton diagrams or the class of transition–output tables. However, as yet nothing has been said of the second class—the *client's metalanguage*; in other words, we have not explained what is meant by the phrase "the client's requirements." In Examples 1 to 8 we disregarded this question and the conditions were formulated in natural language, employing whatever terminology and tools seemed suitable. An exact statement, *a fortiori* solution, of the synthesis problem is impossible without a preliminary definition of the client's metalanguage, and it depends essentially on the specific choice of a metalanguage.

In this chapter we shall examine the synthesis problem for several special metalanguages; this will lead to a better understanding of the conditions to be imposed on the metalanguage.

To conclude this section, we direct our attention to the problem of automaton analysis, which is a natural inverse of the synthesis problem:

* *Translator's note*: The author uses this term (Russian: *metayazyk*) in a sense rather different from that customary in mathematical logic. However, in this context it is quite natural and unambiguous.

given a finite automaton, we wish to characterize its behavior in terms of a given metalanguage. A solution to the problem is an algorithm which, given any automaton, produces a suitable "formula" of the metalanguage.

As we shall see later, for most interesting metalanguages the construction of an analysis algorithm is incomparably simpler than that of a synthesis algorithm. Our principal attention will therefore be directed toward the synthesis problem, and only occasionally, in passing, shall we mention the analysis of automata.

III.3. The metalanguages of sources (macrosources), trees and grammars

A description of certain useful metalanguages is in effect implicit in the previous chapters. One of these is the set of all sources (macrosources). Here the structural objects, in terms of which the client describes the projected language, are the sources (macrosources) themselves. As we already know, the designer's answer to question A is positive, while the construction C is effected by means of determinization; this is a complete solution to the synthesis problem. The analysis problem is trivial: any automaton may be regarded as a source (macrosource).

In engineering practice, the functioning of the automaton is often specified by stipulating the output word $y = \phi(x)$ to be generated for each word x in a certain finite set $\mathfrak{A}$ of input words. Since the correspondence ϕ is (at least, potentially) given by a table, one can speak of a metalanguage whose formulas are finite tables. Consider Problems A, B, C as applied to this language. There exists a finite automaton $\langle \mathfrak{M}, q_0 \rangle$ whose behavior coincides with ϕ on the set $\mathfrak{A}$ only if ϕ is nonanticipatory: (I) the words x and $\phi(x)$ are equal in length; (II) if x and x' have identical initial segments of length μ, then so have $\phi(x)$ and $\phi(x')$. This property is obviously effectively decidable. If it holds, the mapping ϕ is a partial nonanticipatory operator, and, consequently, can be represented by a finite tree v. It is also clear that conditions (I) and (II) are not only necessary, but also sufficient, since any finite tree possesses infinitely many different extensions which are infinite trees of finite-state operators. One variant of the automaton diagram may be constructed by using the contraction procedure of Section II.4. Moreover, this procedure yields an automaton of minimal weight whose behavior is an extension of the finite tree (partial operator). The analysis problem is again trivial: finite trees of arbitrary height can be constructed given a finite automaton.

We thus have examples of metalanguages for the description of languages

and ω-languages (sources and macrosources), and also operators (finite trees). In all cases there are synthesis and analysis algorithms which in fact have already been presented in previous chapters. However, as we shall soon see, the situation is quite different when we wish to use context-free (CF) grammars as our metalanguage, although this approach is quite natural: the client presents the designer with a certain CF-grammar, and asks the following question:

A'. *Can the language L generated by the grammar G be represented in some finite automaton*? (In other words, are the grammatically well-formed strings accepted by a suitable finite automaton?)

If the answer is positive, the client advances an additional demand:

C. *Construct a finite automaton representing the language L.*

In this case, however, the designer has no recourse to an algorithm enabling him to meet the customer's specifications:

THEOREM 3.1. *There is no algorithm which, given an arbitrary CF-grammar G, decides whether the language L generated by G is finite-state or not.*

Proof. Consider a fixed alphabet $X = \{0, 1, c, *\}$ and an arbitrary system of pairs of words over the subalphabet $\{0,1\}$:

$$(a_1, b_1), (a_2, b_2), \ldots, (a_s, b_s). \qquad (\#)$$

We are going to show that any system (#) can be associated with a language $L_\#$ over the alphabet X with the following properties:

I. $L_\#$ is a CF-language (even linear!) and a CF-grammar generating it is effectively constructible from the system (#).

II. $L_\#$ is a finite-state language if and only if the system (#) has a solvable Post correspondence problem.*

It is clear that properties I and II directly imply the theorem, since they show that solution of the correspondence problem for (#) is effectively reducible to deciding whether $L_\#$ is finite-state or not. By Post's Theorem there is no algorithm solving the correspondence problem for arbitrary (#), and so there is no algorithm which decides whether the languages $L_\#$ are finite-state, given CF-grammars generating them. We now construct a language with properties I and II.

I. *Description of* $L_\#$; *its linearity.* Given a system (#). In order to define the languate $L_\#$, we need three other languages L, L_s and L_p, the first two of which are independent of (#), while the third is not. We shall use the

* See Section II.6 (p. 107).

letters $w_1, w_2, \ldots$ for words (possibly empty) over the subalphabet $\{0, 1\}$.

The language L consists of all words of the form

$$cw_1cw_2 \ldots cw_nc*cw_{n+1} \ldots cw_{n+k}c \qquad (n, k = 0, 1, \ldots). \tag{1}$$

L is obviously a finite-state language. L_s and L_p are sublanguages of L.

L_s consists of all words (1) which are *c-symmetric*, that is, the word $w_{n+1}w_{n+2} \ldots w_{n+k}$ is the reflection of $w_1w_2 \ldots w_n$. In other words, deletion of all occurrences of the letter c yields a word which is symmetric in the usual sense.

L_p consists of all #-symmetric words, defined as follows: 1) $n = k$; 2) each of the words $w_1, w_2, \ldots, w_n$ is a "left-hand" word of the system (#), i.e., one of the words $a_1, a_2, \ldots, a_s$; 3) each of the words $w_{n+1}, w_{n+2}, \ldots, w_{n+k}$ is the reflection of a right-hand word of (#): $b_1, b_2, \ldots, b_s$; 4) w_1 and the reflection of w_{n+k} form a single pair of the system (#), the same holds for w_2 and w_{n+k-1}, and so on.

We now define $L_\#$ to be the language $(L - L_s) \cup (L - L_p)$. The first term $L - L_s$ is precisely the language L_4 in Section I.12; it was shown there that this language is linear. The second term $L - L_p$ is the language L_6 described in that section, for the case in which the system of pairs is $(a_1, \hat{b}_1), \ldots, (a_s, \hat{b}_s)$, where $\hat{b}_i$ is the reflection of b_i. It was proved there that L_6 is also linear. Thus $L_\#$ is the union of two linear languages, and so a linear grammar for it is effectively constructible from linear grammars for its two "component" languages.

II. *The connection between representability of $L_\#$ in a finite automaton and solution of the correspondence problem for* (#). Let L_{sp} denote the intersection of L_s and L_p. Since $L_\# = L - L_{sp}$ and L is a finite-state language, it follows easily that $L_\#$ and L_{sp} are either both finite-state or both not finite-state. It will therefore suffice to prove that L_{sp} is a finite-state language if and only if the system (#) has an unsolvable correspondence problem. To verify this, note first that solvability of the correspondence problem for (#) is equivalent to the existence of a word

$$cw_1cw_2 \ldots cw_nc * cw_{n+1}c \ldots cw_{n+n-1}cw_{n+n}c,$$

which is both c-symmetric and #-symmetric. Now, by definition, the set of all these words is precisely the language L_{sp}. Let (#) have an unsolvable correspondence problem; then no such words exist, the language L_{sp} is empty and so is certainly finite-state (in this case $L_\#$ simply coincides with the finite-state language L).

Now assume that (#) has a solvable correspondence problem, and so L_{sp} is nonempty and contains a word $c\Omega_1 * \Omega_2c$, where $\Omega_1 = w_1cw_2c \ldots w_nc$,

$\Omega_2 = cw_{n+1}cw_{n+2}c \dots cw_{2n}$. Then the language L_{sp} must also contain all words of the form $c\Omega_1\Omega_1 * \Omega_2\Omega_2 c$, $c\Omega_1\Omega_1\Omega_1 * \Omega_2\Omega_2\Omega_2 c$, ..., $c\Omega_1^m * \Omega_2^m c$, ... ($m = 1,2,\dots$), and no words of the form $c\Omega_1^m * \Omega_2^n c$ for $m \neq n$. Thus, the language L_{sp} separates* the languages $\{c\Omega_1^m * \Omega_2^m c\}$ ($m = 1,2,\dots$) and $\{c\Omega_1^n * \Omega_2^n c\}$ ($n \neq m$). Now an argument as at the end of Section I.2 shows that L_{sp} cannot be finite-state. This completes the proof.

REMARK. The class of CF-languages is not closed under complementation and intersection; therefore, we cannot state that the language L_{sp} is context-free. This is the reason why, despite the fact that the proof of the theorem essentially involves the language L_{sp}, we worked with the language $L_{\#}$, which is surely context-free.

III.4. The metalanguage of regular expressions

A *regular expression* over the alphabet $X = \{a, b, c, \dots\}$ is defined inductively as follows. Every letter in X is a regular expression; if $\mathfrak{A}_1$ and $\mathfrak{A}_2$ are regular expressions, so are $(\mathfrak{A}_1) \vee (\mathfrak{A}_2)$, $(\mathfrak{A}_1)\cdot(\mathfrak{A}_2)$, $(\mathfrak{A}_1)^*$. Every regular expression over X defines a unique language over X, via the following interpretation:

(I) a (or b, or $c, \dots$) is the language consisting of the single one-letter word a;

(II) $\vee, \cdot, *$ denote union, concatenation and iteration closure of languages.

The synthesis problem for the metalanguage of regular expressions involves solution of Problems A′ and C′; the complete solution is in fact already implicit in the theorems of Sections I.6, I.7. The procedures underlying these theorems construct a source (more precisely, a two-terminal) representing the language defined by a regular expression. To see this it suffices to make the following observations:

1) The language consisting of a single one-letter word a is represented in a two-terminal with a single edge, labeled a.

2) The operations on languages specified in the definition of a regular expression are effectively definable by suitable constructions on two-terminals representing these languages: parallel connection realizes union, serial connection—concatenation product, and cycling—iteration closure.

Thus, using induction on the number of operations involved in a regular expression, one can construct the corresponding two-terminal source, and determinization then yields the required automaton. We have thus proved

* For the definition of separability, see Section I.2.

THEOREM 3.2. *Every regular expression defines a language representable in a finite automaton; there exists an algorithm which, given a regular expression, constructs the corresponding automation.*

REMARK. The same theorems of Sections I.6, I.7 imply that Theorem 3.2 remains valid for a more comprehensive metalanguage, which, besides the operations $\vee, \cdot, *$, also admits the operation symbols &, $\neg$.

EXAMPLE 1. Given the regular expression $(a \vee b)^* ba \vee b^* \vee (ab)^*$ (to simplify notation we have omitted some parentheses and all concatenation symbols; we adopt the convention that $\cdot$ dominates $\vee$, and use the associativity of the operations $\cdot$ and $\vee$). In Figure 26 we construct a two-terminal representing the language in question, proceeding step by step according to the inductive definition of the regular expression. The determinization step is omitted.

REMARK I. In synthesizing an automaton on the basis of a given regular expression $\mathfrak{A}$, it is natural, wherever possible, first to simplify this expression using various identities (for example, every subexpression $(\mathfrak{R}^*)^*$ may be replaced by $\mathfrak{R}^*$, and so on).

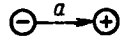
II. It is easy to indicate an upper bound for the number of states of an automaton obtained by synthesis for a regular expression, in terms of the number of occurrences of various symbols in the expression. Suppose that a regular expression $\mathfrak{A}$ contains $n(X)$ occurrences of letters from the alphabet X, $n(\vee)$ occurrences of $\vee$, $n(\cdot)$ occurrences of $\cdot$, $n(*)$ occurrences of $*$. Then the above method constructs a source $\mathfrak{B}$ in which the number of vertices is

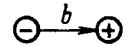
$$n(\mathfrak{B}) = 2n(X) - 2n(\vee) - n(\cdot) + n(*).$$

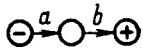
Since $n(\vee) + n(\cdot) \leqq n(X) - 1$, we have $n(\mathfrak{B}) \leqq n(X) + n(*) - 1$. If we first delete all "superfluous" occurrences of the symbol $*$ from $\mathfrak{A}$ (see the previous Remark), the number of occurrences of $*$ is at most $2n(X) - 1$; thus $n(\mathfrak{B}) \leqq 3n(X) - 2$. Therefore, the number of states in an automaton obtained by synthesis is at most $2^{3n(X)-2}$.

Consideration of the analysis problem in the metalanguage of regular expressions necessitates the following observation. Our definition of regular expressions always yields a nonempty language, while there do exist algorithms producing the empty language. Therefore, in all future quests for a regular expression for a language $\mathfrak{A}$, we shall assume that we have already ascertained whether $\mathfrak{A}$ is empty or not (another alternative

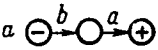
CONSTRUCTION OF A TWO-TERMINAL SOURCE FOR THE REGULAR EXPRESSION $(a \vee b)^* ba \vee b^* \vee (ab)^*$

(1) a 

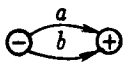
(2) b 

(3) ab 

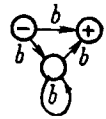
From (1) and (2) by serial connection

(4) ba 

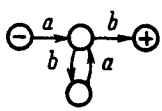
From (2) and (1) by serial connection

(5) $(a \vee b)$ 

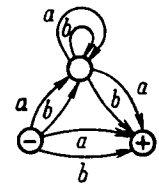
From (1) and (2) by parallel connection

(6) b^* 

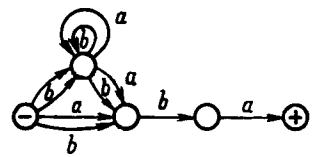
From (2) by cycling

(7) $(ab)^*$ 

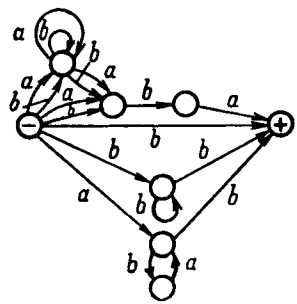
From (3) by cycling

(8) $(a \vee b)^*$ 

From (5) by cycling

(9) $(a \vee b)^* ba$ 

From (8) and (4) by serial connection

(10) $(a \vee b)^* ba \vee b^* \vee (ab)^*$ 

From (9), (6), (7) by parallel connection

Figure 26

would be to introduce a special symbol, $\varnothing$, as the regular expression describing the empty language). In all cases of interest, Theorem 1.4 and Remark III in Section I.5 guarantee that one can effectively decide whether the language is empty.

THEOREM 3.3. (ANALYSIS IN THE METALANGUAGE OF REGULAR EXPRESSIONS). *Every finite-state language can be defined by a regular expression (briefly, the language is regular). There exists an algorithm which, given a source (in particular, an automaton), constructs a regular expression defining the language.*

Proof. Given a source $\langle B,q_0,Q'\rangle$. Since $\omega(B,q_0,Q') = \bigvee_{q_i\in Q'} \omega(B,q_0,q_i)$, it will suffice to prove the theorem for a source $\langle B,q_0,q_i\rangle$. Let $\tilde{\omega}(B,q_0,q_i)$ denote the sublanguage of $\omega(B,q_0,q_i)$ generated by all paths beginning at q_0 and containing no intermediate passage through q_i. It is obvious that $\omega(B, q_0, q_i)$ is nonempty if and only if $\tilde{\omega}(B, q_0, q_i)$ is nonempty.† Now

$$\omega(B,q_0,q_i) = \begin{cases} \tilde{\omega}(B,q_0,q_i)\,(\tilde{\omega}(B,q_i,q_i))^*, & \text{if } \tilde{\omega}(B, q_0, q_i) \text{ is nonempty;} \\ \tilde{\omega}(B,q_0,q_i) & \text{otherwise.} \end{cases}$$

Hence it will suffice to prove the theorem for all languages of type $\tilde{\omega}(B,q_0,q_i)$; this we proceed to do by induction on the number k of states of the source. The regular expression which is the disjunction of all letters labeling the edges going from q_i to q_j will be denoted by $[q_i,q_j]$ (if there are no such edges we write $[q_i, q_j] = \varnothing$). The notation $[q_i, q_j]$ will also be used for the language defined by this regular formula; it should always be clear from the context in what sense the notation is being used.

Base. When $k = 1$ the language $\tilde{\omega}(B,q_0,q_0)$ is defined by the formula $[q_0,q_0]$.

Induction step. First assume that $q_i \neq q_0$. If there are no edges going to q_i, then $\tilde{\omega}(B,q_0,q_i)$ is empty. Otherwise, let Q' be the set of all edges other than q_i from which edges go to q_i. Denote by B' the diagram obtained from B by deleting the vertex q_i and all edges incident upon it. By the induction hypothesis, one can determine, for all $q' \in Q'$, whether the language $\tilde{\omega}(B',q_0,q')$, hence also $\omega(B',q_0,q')$, is empty or not, and if it is nonempty we can effectively find a regular expression for it. If $\omega(B',q_0,q')$ is not empty, then $\omega(B',q_0,q')\cdot[q',q_i] \subseteq \tilde{\omega}(B,q_0,q_i)$; if $\omega(B',q_0,q')$ is empty and $q_0 = q'$, then $[q',q_i] \subseteq \tilde{\omega}(B,q_0,q_i)$. Obviously, the union of all these sublanguages

† Recall that one can effectively decide whether $\omega(B,q_0,q_i)$ is empty or not.

is precisely $\tilde{\omega}(B,q_0,q_i)$. We thus obtain a regular expression for $\tilde{\omega}(B,q_0,q_i)$, as the sum (disjunction) over q' of the regular expressions for the languages $\omega(B',q_0,q')\cdot[q',q_i]$ and $[q',q_i]$.

Now let $q_i = q_0$. Then, by deleting q_0 and all edges incident upon it, we get a subdiagram B'; let Q' be the set of all vertices in B' to which edges go from q_0, Q'' the set of vertices from which edges go to q_0. As before, the language $\tilde{\omega}(B,q_0,q_0)$ is the union of languages of three types:

(I) $[q_0,q_0]$;

(II) for each pair $q'\in Q'$, $q''\in Q''$ such that $\omega(B',q',q'')$ is nonempty, a language $[q_0,q']\cdot\omega(B',q',q'')\cdot[q'',q_0]$;

(III) a language $[q_0,q']\cdot[q',q_0]$ for each $q'\in Q'$ such that $q'\in Q''$.

Thus, analysis of the original source reduces to that of sources with fewer vertices, and the theorem is proved.

EXAMPLE 2. Consider the behavior of the automaton B defined by the diagram of Figure 27*a*. The initial vertex is q_0, the final vertex q_3. Then

$$\omega(B,q_0,q_3) = \tilde{\omega}(B,q_0,q_3)\,(\tilde{\omega}(B,q_3,q_3))^*,$$

$$\tilde{\omega}(B,q_0,q_3) = \omega(B',q_0,q_1)\cdot[q_1,q_3] \vee \omega(B',q_0,q_2)\cdot[q_2,q_3],$$

$$\tilde{\omega}(B,q_3,q_3) = [q_3,q_0]\cdot\omega(B',q_0,q_1)\cdot[q_1,q_3] \vee$$
$$\vee\,[q_3,q_0]\cdot\omega(B',q_0,q_2)\cdot[q_2,q_3] \vee [q_3,q_2]\cdot[q_2,q_3],$$

where the diagram B' is that illustrated in Figure 27*b*. Since B' is very simple, regular expressions for $\omega(B',q_0,q_1)$ and $\omega(B',q_0,q_2)$ are easily derived: $\omega(B',q_0,q_1) = 0$, $\omega(B',q_0,q_2) = 0\cdot 0 \vee 1$.

Consequently,

$$\omega(B,q_0,q_3) = \{0\cdot(1 \vee 00 \vee 01) \vee (00 \vee 1)\cdot(0 \vee 1)\}\cdot$$
$$\cdot\{1\cdot 0\cdot(1 \vee 00 \vee 01) \vee 1\cdot(00 \vee 1)\cdot(0 \vee 1) \vee 0\cdot(0 \vee 1)\}^* =$$
$$= (01 \vee 000 \vee 001 \vee 10 \vee 11)\cdot(101 \vee 1000 \vee 1001 \vee 110 \vee 111 \vee 00 \vee 01)^*.$$

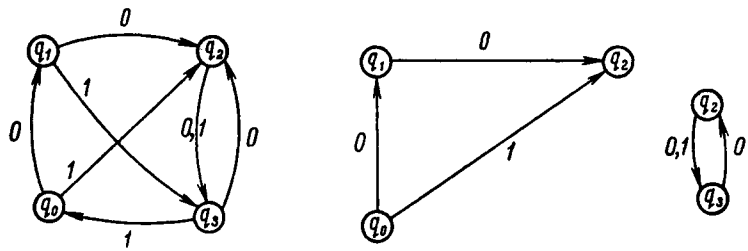
REMARK. If the languages under consideration may contain the empty word, it is convenient to generalize the concept of a regular expression over the alphabet $X = \{a,b,c,\dots\}$ by adding a special symbol $\wedge$ (not an element of X) for empty words. When this is done, the star operation is usually interpreted as closure with addition of the empty word:

$$\mathfrak{A}^* = \wedge \vee \mathfrak{A} \vee \mathfrak{A}\cdot\mathfrak{A} \vee \mathfrak{A}\mathfrak{A}\mathfrak{A} \vee \dots$$

The synthesis and analysis algorithms described above carry over almost unchanged to this case; we need only take the following precautionary measures:

(I) *Analysis.* The following changes should be made in the expression $\mathfrak{A}$ obtained from the above algorithm: 1) if the initial state of the automaton is also a final state, replace $\mathfrak{A}$ by $\wedge \vee \mathfrak{A}$; 2) replace every subexpression of the form $\mathfrak{B}^*$ by $\mathfrak{B} \cdot \mathfrak{B}^*$.

(II) *Synthesis.* 1) First replace every subexpression $\mathfrak{B}^*$ by $\wedge \vee \mathfrak{B}^*$; 2) a two-terminal for the expression $\wedge$ consists of one empty edge.



a *b* *c*

Figure 27

III.5. The metalanguage of ω-regular expressions

ω-regular expressions over an alphabet X are defined inductively as follows.

1) If $\mathfrak{A}$ is a regular expression, then $(\mathfrak{A})^\infty$ is an ω-regular expression; the ω-language defined by this expression is the strong iteration closure of the language defined by $\mathfrak{A}$.

2) If $\mathfrak{A}$ is a regular expression and $\mathfrak{B}$ an ω-regular expression, then $\mathfrak{A} \cdot \mathfrak{B}$ is an ω-regular expression; it defines the concatenation product of the language defined by $\mathfrak{A}$ and the ω-language defined by $\mathfrak{B}$.

3) If $\mathfrak{B}_1$ and $\mathfrak{B}_2$ are ω-regular expressions, so is $\mathfrak{B}_1 \vee \mathfrak{B}_2$; it defines the union of the corresponding ω-languages.

For example, consider the ω-regular expression $(1 \vee 0^*1)^\infty$. It defines the ω-language consisting of all ω-words with infinitely many ones. In Example 8 of Section III.1 we gave a verbal description of this ω-language, and also solved Problems A′, C′.

We are now in a position to describe a synthesis algorithm (i.e., to solve Problems A′, C′ for arbitrary ω-regular expressions.

Given an arbitrary ω-regular expression $\mathfrak{B}$, let $\Omega(\mathfrak{B})$ denote the ω-language that it defines. According to Theorem 3.2, there exists an algorithm which, given any regular subexpression $\mathfrak{A}$ of $\mathfrak{B}$, will construct a suitable finite

automaton. Now, with the aid of Theorems 1.11 and 1.12, this algorithm can be extended to an algorithm which, given the ω-regular expression $\mathfrak{B}$, constructs an automaton representing the ω-language $\Omega(\mathfrak{B})$. On the other hand, examination of the proof of Theorem 1.13 shows that for any finite macrosource $\langle B, q_0, \mathfrak{C}\rangle$ one can effectively construct an ω-regular expression defining the ω-language $\Omega(B, q_0, \mathfrak{C})$. At this point, as in the case of regular expressions, one should introduce certain stipulations as to the notation (a special symbol $\varnothing$) and acceptance of empty languages. The construction of an ω-regular expression from a finite macrosource proceeds as follows. With each limit macrostate $\Gamma_i = \{q_1, \ldots, q_s\} \in \mathfrak{C}^\dagger$ and initial state $q_0 \in Q$, associate two finite-state languages $\mathfrak{A}_{i_1}, \mathfrak{A}_{i_2}$; in the notation of Theorem 1.13, these are $\omega(B, q_0, q_1)$ and the concatenation product $\omega(B', q_1, q_2) \cdot \omega(B', q_2, q_3) \cdot \ldots \cdot \omega(B', q_s, q_1)$. Regular expressions for these languages are effectively constructible. Since the ω-language $\Omega(B, q_0, \mathfrak{C})$ is the union of ω-languages of type $\mathfrak{A}_{i_1} \cdot \mathfrak{A}_{i_2}^\infty$, a suitable ω-regular expression can be obtained in a natural way from the regular expressions for $\mathfrak{A}_{i_1}, \mathfrak{A}_{i_2} (i = 1, 2, \ldots)$. We have thus proved

THEOREM 3.4. (*I*) *Every ω-regular expression defines an ω-language representable in a finite automaton; there exists a (synthesis!) algorithm which, given an ω-regular expression, constructs a finite automaton representing the corresponding ω-language.*

(*II*) *Every ω-language representable in a finite automaton is definable by an ω-regular expression; there exists an (analysis!) algorithm which, given a finite macrosource, constructs an ω-regular expression defining the corresponding ω-language.*

EXAMPLE 1. (ANALYSIS). Find an ω-regular expression for the ω-language $\Omega(B, q_0, \{q_2, q_3\})$ (see Figure 27*a*). We have

$$\Omega(B, q_0, \{q_2, q_3\}) = \omega(B, q_0, q_3) \cdot \Omega(B'', q_3, \{q_2, q_3\}),$$

where B'' is the diagram of Figure 27*c*. We have already found a regular expression for the language $\omega(B, q_0, q_3)$ (Example 2, Section III.4):

$$(01 \vee 000 \vee 001 \vee 10 \vee 11) \cdot (101 \vee 1000 \vee 1001 \vee 110 \vee 111 \vee 00 \vee 01)^*.$$

An ω-regular expression for $\Omega(B'', q_3, \{q_2, q_3\})$ may be based on the representation

$$\Omega(B'', q_3, \{q_2, q_3\}) = (\omega(B'', q_3, q_2) \cdot \omega(B'', q_2, q_3))^\infty.$$

† We have simplified the notation for states by ignoring the dependence on i.

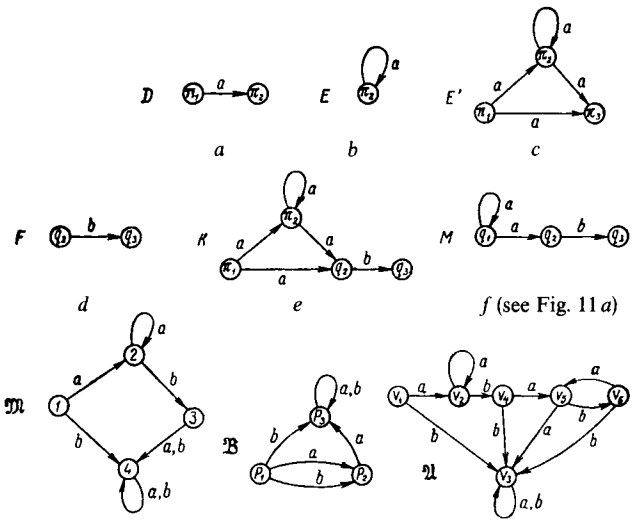


Figure 28

However, direct examination of the simple diagram B'' yields the expression $(0 \cdot (0 \vee 1))^{\infty}$. The final ω-regular expression for our ω-language is

$$(01 \vee 000 \vee 001 \vee 10 \vee 11) \cdot \cdot (101 \vee 1000 \vee 1001 \vee 110 \vee 111 \vee 00 \vee 01)^* \cdot (0 \cdot (0 \vee 1))^{\infty}.$$

EXAMPLE 2. (SYNTHESIS). Construct a macroanchored automaton representing the ω-language defined by the ω-regular expression $a^*b(ab)^{\infty}$.

Figure 28 illustrates the steps in the construction of the automaton $\mathfrak{A}$. We first construct an automaton $\mathfrak{M}$ representing the language defined by the regular expression a^*b (Figure 28, *a–g*), and then an automaton $\mathfrak{B}$ representing the ω-language defined by the strong closure $(ab)^{\infty}$ (Figure 28*h*). Finally, applying the algorithm from the Concatenation Theorem, we get the required automaton $\mathfrak{A}$ (Figure 28*i*).

Construction of the automaton $\mathfrak{M}$ for the subexpression a^*b proceeds in steps:

a) a source $\langle D, \pi_1, \pi_2 \rangle$ (Figure 28*a*) for the subexpression a;

b) a source $\langle E, \pi_2, \pi_2 \rangle$ (Figure 28*b*) for the subexpression a^*;

c) a two-terminal $\langle E', \pi_1, \pi_3 \rangle$ (Figure 28*c*), obtained from $\langle E, \pi_2, \pi_2 \rangle$ by introducing an input terminal π_1 and output terminal π_3;

d) a source $\langle F, q_2, q_3 \rangle$ (Figure 28*d*) for the subexpression b;

e) a source $\langle K, \pi_1, q_3 \rangle$ (Figure 28*e*), obtained by identifying the output

terminal π_3 of the two-terminal $\langle E', \pi_1, \pi_3 \rangle$ with the input terminal q_2 of the two terminal $\langle F, q_2, q_3 \rangle$; this source corresponds to the subexpression a^*b;

f) a source $\langle M, q_1, q_3 \rangle$ (Figure 28*f*), obtained from $\langle K, \pi_1, q_3 \rangle$ by merging the equivalent states π_1, π_2 (call the resulting vertex q_1), representing the language defined by the expression a^*b (comparing Figure 28*f* with Figure 11*a*, we see that they illustrate the same diagram);

g) an automaton $\langle \mathfrak{M}, 1, 3 \rangle$ (Figures 28*g* and 11*c*) obtained by determinization of the source $\langle M, q_1, q_3 \rangle$ (see Section I.7, Example 1).

Construction of an automaton $\langle \mathfrak{B}, p_1, \{p_1, p_2\} \rangle$ (Figure 28*h*) representing $(ab)^\infty$ was given in the example of Section I.10.

Construction of an automaton $\langle \mathfrak{A}, v_1, \{v_4, v_5\} \rangle$ (Figures 28*i* and 12*c*) representing the concatenation product of the language $\omega(\mathfrak{M}, 1, 3)$ and the ω-language $\Omega(\mathfrak{B}, p_1, \{p_1, p_2\})$ was described in the example following the Concatenation Theorem (Section I.9).

Thus the automaton $\langle \mathfrak{A}, v_1, \{v_4, v_5\} \rangle$ (Figure 28*i*) represents the ω-language defined by the ω-regular expression $a^*b(ab)^\infty$.

III.6. The logical metalanguage I

The statement of the synthesis problem appeals to various metalanguages, some of which have been considered in the preceding sections. Various arguments can be given in favor of choosing one metalanguage or another, or developing a new metalanguage. Nevertheless, one can indicate two requirements which it is quite natural to impose on any existing or projected metalanguage.

The first requirement represents the interests of the client, so to speak, and relates to the expressive power of the metalanguage. Intuitively, a metalanguage K_1 is at least as expressive as a metalanguage K_2 if any statement expressible in K_2 admits a clear reformulation in K_1. It is clear that the more expressive a metalanguage, the more appropriate it is for preliminary formulation of problems.

The second requirement reflects the viewpoint of the designer; there must be a fairly simple algorithm for solution of the synthesis problem in the metalanguage.

In a certain sense these two requirements are contradictory. The more comprehensive and expressive the metalanguage, the more universal and so less simple the algorithm. Moreover, it is not difficult to see that if the metalanguage is too comprehensive the required algorithm may not exist

at all. We shall "take sides" in this conflict, supporting the client in his demand for a metalanguage as expressive as possible, provided a synthesis algorithm still exists. In our quest, we shall appeal to the apparatus of mathematical logic, which, thanks to the suitable formalization of logical connectives and operations (conjunction, negation, quantifiers), achieves a close resemblance to natural language and usual modes of thought. For instance, the verbal conditions imposed by the client in Example 4 (Section III.1) on the ω-words

$$y = y(1)y(2)\dots y(t)\dots,$$
$$x_1 = x_1(1)x_1(2)\dots x_1(t)\dots,$$
$$x_2 = x_2(1)x_2(2)\dots x_2(t)\dots,$$

may be expressed as follows:

$$\exists^\infty t[y(t) = 1] \leftrightarrow \exists^\infty t[x_1(t) = 1] \,\&\, \exists^\infty t[x_2(t) = 1],$$

where $\exists^\infty t$ is the quantifier "there exist infinitely many t." The corresponding formula for Example 2 is

$$\forall t[y(2t - 1) = 1] \leftrightarrow \neg(x_1(2t) = 1 \leftrightarrow x_2(2t) = 1).$$

It is more difficult to set up an analogous formal notation for Examples 1 and 3. A natural suggestion here would be to employ symbols for addition and multiplication. However, it is well known that there is no algorithm that decides the truth of formulas built up from equalities of the type $t + \tau = u$, $t \cdot \tau = u$ (t, u, τ are variables ranging over the natural numbers) using the operations of the predicate calculus (&, $\vee$, $\neg$, $\rightarrow$, $\forall$, $\exists$). This circumstance, as well as other similar ones (there exists no algorithm deciding whether formulas of the predicate calculus with one binary predicate are tautologies), obliges us to exercise more caution in our choice of tools. We are going to describe a logical metalanguage **I** which, as far as we know at present, involves no "embarrassing" situations. We shall endeavor to present convincing arguments for the thesis that the metalanguage **I** is in fact considerably more expressive than those considered hitherto. It is far more difficult to prove that the synthesis problem in this language is algorithmically solvable—this will be done in the following sections. The metalanguage **I** is intended to provide formulation of statements about ω-words over different finite alphabets. The letters, which may occur in various alphabets, form a denumerable list of constants:

$$a, b, c, \dots \text{(possibly indexed)}.$$

Consider the set of all ω-words over an alphabet $\{a_i, a_j, \ldots, a_s\}$. Variables ranging over this set will be denoted by symbols $x^{a_i,a_j,\ldots,a_s}$, $y^{a_i,a_j,\ldots,a_s}$, etc., where the superscripts indicate the selected alphabet $\{a_i, a_j, \ldots, a_s\}$. Apart from these variables, which we shall call *predicate variables*,* the language also contains *individual variables*, t, τ, ρ, σ, ... ranging over the natural numbers. There is a constant 1 and a function symbol ϕ, denoting the successor function on natural numbers (i.e., $\phi(t) = t + 1$). Finally, the metalanguage **I** contains symbols for the logical connectives ($\&$, $\vee$, $\neg$, $\rightarrow$), quantifiers ($\forall$, $\exists$), and also parentheses and the equality sign: (, = ,).

We now define the terms, atomic formulas and formulas of **I** (**I**-formulas) and explain their semantics. As is customary, we shall frequently employ metanotation.

Expressions of the form t, $\phi(t)$, $\phi\phi(t)$, $\phi\phi\phi(1)$, etc., are *terms*; we shall replace them by the more familiar notation

$$t, t + 1, t + 2, 4, \text{etc.}$$

Let q be a predicate variable, a a letter from the corresponding alphabet,** ξ a term. A formula of the form $q(\xi) = a$ is called an *atomic formula* of **I**; it expresses the statement: the ξ-th entry in the ω-word q is the letter a. In accordance with this interpretation, the formula $q(\xi) = a$ assumes a well-defined truth value (i.e., true or false) as soon as the values of the variables occurring therein (the predicate variable q and any individual variable appearing in ξ) are prescribed. All other formulas of **I** (and their semantics) are derived from atomic formulas by the usual application of logical connectives and quantifiers, over both predicate variables (predicate quantifiers) and individual variables (individual quantifiers). We shall employ notation such as $\mathfrak{A}(x^{a,b}, q^{a,b,c}, t, \tau)$, $\mathfrak{B}(x^{a,b}, \tau)$, etc., to indicate the variables which have free occurrences in a formula; for example, $\mathfrak{B}(x^{a,b}, \tau)$ denotes a formula in which there is only one free predicate variable $x^{a,b}$ and one free individual variable τ.

A formula which contains no free variables is called a *sentence*. A sentence has a well-defined truth value, induced by the interpretation of the symbols indicated above. However, if the formula contains free occurrences of certain variables (predicate or individual) its truth or falsity in any situation

* *Translator's note*: This is Büchi's terminology [73]; the motivation follows from the remark at the end of this section.

** Henceforth, whenever no confusion can arise, we shall omit the superscripts indicating the alphabet associated with a predicate variable. In this case, therefore, q is an abbreviation for $q^{\cdots,a,\cdots}$.

depends on the actual objects (ω-words or natural numbers) substituted for the variables.

EXAMPLE. The sentence

$$\forall t \forall x^{a,b}[x^{a,b}(t) = a \rightarrow x^{a,b}(t+1) = a]$$

states that, in every ω-word (over the alphabet $\{a,b\}$), each occurrence of the letter a is followed by another occurrence of a; it is clearly false. The formula $\forall t[x^{a,b}(t) = a \rightarrow x^{a,b}(t+1) = a]$ contains the free variable $x^{a,b}$; it is true if and only if the ω-word $aaaa\ldots$ (infinite block of a's) or an ω-word of the form $bb\ldots baaa\ldots$ (block of b's followed by infinite block of a's) is substituted for $x^{a,b}$. The formula

$$\mathfrak{A}(\tau, t) = \forall x^{a,b}[\forall \rho(x^{a,b}(\rho) = \\ = a \rightarrow x^{a,b}(\rho+1) = a) \,\&\, x^{a,b}(\tau) = a \rightarrow x^{a,b}(t) = a] \qquad (\#)$$

contains the free (individual) variables τ, t. It is easily verified that it is true if and only if τ, t are natural numbers such that $\tau \leqq t$.

Let $\mathfrak{A}(\tau, t, \ldots, \rho)$ be a formula containing m free individual variables and no predicate variables; it can be associated with the set of all m-tuples of natural numbers for which it is true. We denote this set by $\hat{\tau}\hat{t}, \ldots \hat{\rho}\mathfrak{A}(\tau,t,\ldots,\rho)$, and say that it is *defined* by the formula $\mathfrak{A}(\tau,t,\ldots,\rho)$. Since there is one-to-one correspondence between sets of m-tuples of natural numbers and m-ary relations over the natural numbers (m-ary predicates), we speak of relations (predicates) defined by I-formulas. For example, formula (#) defines the relation $\leqq$ over the natural numbers. We shall now explain in what sense **I** can be employed as a metalanguage in the theory of automata.

A formula $\mathfrak{A}(x^{a,b,\ldots,d})$ with a single free variable (the predicate variable $x^{a,b,\ldots,d}$) is said to *define* the set of all ω-words over the alphabet $\{a,b,\ldots,d\}$ for which it is true. We denote this set by $\hat{x}^{a,b,\ldots,d}\mathfrak{A}(x^{a,b,\ldots,d})$. For example, the formula $\forall t[x^{a,b}(t) = a \rightarrow x^{a,b}(t+1) = a]$ considered above defines the set of ω-words (i.e., ω-language) consisting of $aaaa\ldots$ and all ω-words of the type $bb\ldots baaa\ldots$ Similarly, a formula $\mathfrak{A}(x_1, \ldots, x_m)$ with m free predicate variables defines a set of m-tuples of ω-words, which is denoted by $\hat{x}_1 \ldots \hat{x}_m \mathfrak{A}(x_1, \ldots, x_m)$.

For example, consider the formula $\Pi(x^{a,b}, y^{c,d}, z^{ac,ad,bc,bd})$, in which the alphabet of the predicate variable z is the product of the alphabets of the variables x, y:

$$\Pi(x^{a,b}, y^{c,d}, z^{ac,ad,bc,bd}) \underset{df}{=} \forall t[(z(t) = ac \leftrightarrow$$

$$\leftrightarrow x(t) = a \& y(t) = c) \& (z(t) = ad \leftrightarrow x(t) = a \& y(t) = d) \&$$

$$\& (z(t) = bc \leftrightarrow x(t) = b \& y(t) = c) \& (z(t) = bd \leftrightarrow x(t) = b \& y(t) = d)].$$

It obviously defines the set of triples $\langle x, y, z \rangle$, where the ω-word z is the coupling of the ω-words x and y (equivalently, x and y are the projections of z onto the corresponding alphabets). Therefore, if a formula $\mathfrak{A}(x, y)$ defines a set of pairs of ω-words $\hat{x}\hat{y}\mathfrak{A}(x, y)$, then the formula $\mathfrak{B}(z) \overline{\underset{df}{=}} \exists xy[\mathfrak{A}(x, y) \& \Pi(x, y, z)]$ defines the related set of ω-word couplings. Similarly, given a formula $\mathfrak{A}(z)$ one constructs a formula $\exists z[\mathfrak{B}(z) \& \Pi(x, y, z)]$ which defines the set of pairs of projections of the ω-words in $z\mathfrak{B}(z)$. In view of this very simple relation between a formula defining a set of pairs of ω-words and the formula defining the corresponding set of couplings we shall henceforth associate with the formula $\mathfrak{A}(\widehat{x_1, \ldots, x_m})$ not only the set $\hat{x}_1 \ldots \hat{x}_m \mathfrak{A}(x_1, \ldots, x_m)$ but also the ω-language consisting of the corresponding couplings; the latter will be denoted by $\widehat{x_1 \ldots x_m}\mathfrak{A}(x_1, \ldots, x_m)$, and we shall say, speaking rather loosely, that it too is defined by the formula $\mathfrak{A}(x_1, \ldots, x_m)$. Similarly, for any partition of the set of variables into μ disjoint subsets, we shall say that the formula $\mathfrak{A}(x_1, \ldots, x_m)$ defines the set of μ-tuples of ω-words over the corresponding alphabets. We shall be especially concerned with partitions into two groups; for example, in the formula $\mathfrak{A}(x_1, \ldots, x_s, y_1, \ldots, y_k)$ we might call $x_1, \ldots, x_s$ "input" variables, and $y_1, \ldots, y_k$ "output" variables, and consider the set of pairs of ω-words that it defines; this set will be denoted by

$$\widehat{x_1 \ldots x_s}\widehat{y_1 \ldots y_k}\mathfrak{A}(x_1, \ldots, x_s, y_1, \ldots, y_k).$$

We shall now explain in what sense the language **I** (or certain fragments thereof) can be employed as a metalanguage for synthesis theory.

I. Since any formula $\mathfrak{A}(x_1, \ldots, x_m)$ defines an ω-language, in the above described sense, the class of all these formulas ($m = 1, 2, \ldots$) may be regarded as a metalanguage for the description of ω-languages; one can thus formulate Problems A′, C′ of the general synthesis problem.

II. Consider an operator T that transforms ω-words over an input alphabet $\{a, b, \ldots, d\}$ into ω-words over an output alphabet $\{e, f, \ldots, h\}$, and also a formula $\mathfrak{A}(x^{a,b,\ldots,d}, y^{e,f,\ldots,h})$. We shall say that T *satisfies* this formula if the latter is true whenever $y = Tx$; in other words, the graph of the operator T is contained in the set $\hat{x}\hat{y}\mathfrak{A}(x, y)$. The restriction on the number of variables is clearly not essential, since for more than two variables the free variables may always be appropriately divided into "input" and "output" variables. Consequently, the class of formulas containing at least two free (predicate!)

variables may serve as a metalanguage for the description of operators; Problems A, B, C of the general synthesis problem may be formulated in this metalanguage.

III. It remains to see how I-formulas can describe languages. To this end, we first need a few new concepts.

Given a formula $\mathfrak{A}(x)$ whose only free variable is a predicate variable x, and natural numbers $\tau_0, t_0, \tau_0 \leqq t_0$. The formula $\mathfrak{A}(x)$ is said to be *fictitious outside* $[\tau_0, t_0]$ if it has the following property: for any two ω-words

$$x' = x'(1)x'(2)\dots x'(\tau_0)\dots x'(t_0 - 1)x'(t_0)\dots,$$

$$x'' = x''(1)x''(2)\dots x''(\tau_0)\dots x''(t_0 - 1)x''(t_0)\dots,$$

with identical segments

$$x'^{t_0}_{\tau_0 - 1} = x'(\tau_0)\dots x'(t_0) \quad \text{and} \quad x''^{t_0}_{\tau_0 - 1} = x''(\tau_0)\dots x''(t_0)$$

of length $t_0 - \tau_0 + 1$, the formula $\mathfrak{A}(x)$ is either true for both or false for both. In other words, whether an ω-word x belongs to the ω-language $\hat{x}\mathfrak{A}(x)$ depends only on the word $x(\tau_0)\dots x(t_0)$. In this sense, a formula which is fictitious outside $[\tau_0, t_0]$ defines a certain set of words of length $t_0 - \tau_0 + 1$. Now let $\mathfrak{A}(x,t)$ be a formula, containing both a predicate variable x and an individual variable t, having the property: for any fixed t_0, the formula $\mathfrak{A}(x,t_0)$ is fictitious outside $[1,t_0]$. We shall then say that the formula $\mathfrak{A}(x,t)$ is fictitious outside $[1,t]$ and defines the language $\mathfrak{A}$ which is the union of the languages corresponding to the formulas $\mathfrak{A}(x,1)$, $\mathfrak{A}(x,2)$, $\mathfrak{A}(x,3),\dots$ (the formula $\mathfrak{A}(x,t)$ expresses the statement that the word $x(1)\dots x(t)$ belongs to the language $\mathfrak{A}$).

By analogy with the definition of ω-languages by formulas containing several predicate variables, one can also consider definition of languages by formulas $\mathfrak{A}(x,y,\dots,z,t)$ with several predicate variables and one individual variable.

EXAMPLE. The formula

$$\forall\tau\{\forall x^{a,b}[\forall\rho(x^{a,b}(\rho) = a \to x^{a,b}(\rho + 1) = a)\&x^{a,b}(\tau) = a \to x^{a,b}(t) = a] \to$$

$$\to (x^{a,b}(\tau) = a\&y^{a,b}(\tau) = b) \vee (x^{a,b}(\tau) = b\&y^{a,b}(\tau) = a)\}$$

is fictitious outside $[1,t]$ and defines the language (over the alphabet $\{a,b\} \times \{a,b\}$) consisting of all words which contain only the letters $\langle ab\rangle$ and $\langle ba\rangle$. That this is true follows easily from the observation that the underlined subformula defines the inequality $\tau \leqq t$. Thus the condition

expressed by the above formula is

$$\forall\tau[\tau \leqq t \rightarrow x(\tau) = a \& y(\tau) = b \vee x(\tau) = b \& y(\tau) = a].$$

REMARK. Particularly noteworthy is the case of a formula in which all predicate variables (both free and bound) have the same alphabet $\{a,b\}$. Identifying the symbol a with "true" and b with "false," one can regard every ω-word $x^{a,b}$ over the alphabet $\{a,b\}$ as a monadic predicate whose argument ranges over the natural numbers. We can then replace the notation $x^{a,b}(t) = a$, $x^{a,b}(t) = b$ for atomic formulas by the usual predicate notation $x(t)$ (the predicate x is true for the argument t) and $\neg x(t)$ (x is false for the argument t). More complicated formulas also assume their more familiar predicate calculus form. For example, the formula (#) becomes

$$\forall x[\forall\rho(x(\rho) \rightarrow x(\rho + 1)) \& x(\tau) \rightarrow x(t)].$$

Since alphabets of arbitrary cardinality can be represented by binary codes, it follows that there is no loss of generality in confining the discussion of the metalanguage **I** to this narrower version (which might be called the monadic predicate calculus over the natural numbers). We nevertheless prefer to treat the general case, admitting ω-words over arbitrary alphabets (which may be regarded as predicates with several, not necessarily two, truth values).

III.7. Expressive power of the logical metalanguage I

In this section we shall present various illustrations of the expressive power of **I**. Suppose that the conditions imposed on the operator (language, ω-language) are specified by a formula $\mathfrak{A}$ in one of the metalanguages considered previously. Then, as we shall soon see, it is very easy to go from $\mathfrak{A}$ to an **I**-formula expressing the same condition. In essence, this procedure is a natural embedding of the original metalanguage in our logical language. However, this process is irreversible: even when a description is expressible in logical terms, its formulation in other languages (such as regular expressions) may be quite difficult. This is precisely the motivation for our claim that the logical language **I** is more comprehensive and expressive than the previous metalanguages. We shall first show that **I** may be used to define many useful relations and operations which are not primitive concepts of **I**. This makes it possible to extend **I** by introducing various

secondary concepts, which may be effectively eliminated, if so desired, by replacing them with the appropriate (defining) expressions of **I**.

1. The relation of equality of terms ξ and η is defined by the formula $\forall x^{a,b}[x^{a,b}(\xi) = a \leftrightarrow x^{a,b}(\eta) = a]$, which asserts that every ω-word has the same letter at its ξ-th and η-th positions. We express this as follows:

$$\xi = \eta \underset{df}{=} \forall x^{a,b}[x^{a,b}(\xi) = a \leftrightarrow x^{a,b}(\eta) = a]. \tag{1}$$

Similar notation will be employed subsequently to define secondary concepts.

2. As we have already seen, the relation $\leqq$ between terms is defined as follows:

$$\xi \leqq \eta \underset{df}{=} \forall x^{a,b}[\{\forall t(x^{a,b}(t) = a \rightarrow \rightarrow x^{a,b}(t+1) = a) \& x^{a,b}(\xi) = a\} \rightarrow x^{a,b}(\eta) = a]. \tag{2}$$

Since $\xi < \eta$ is equivalent to $\xi \leqq \eta \,\&\, \neg(\xi = \eta)$, it is clear that the relation $<$ is also definable in **I**.

3. Individual quantifiers $\exists^{\infty} t$ (there exist infinitely many t), $\forall^{\infty} t$ (for all t except possibly a finite set), $\exists t]_a^b$ (there exists t in the interval $[a,b]$), $\forall t]_a^b$ (for all t in the interval a, b) may all be defined in **I**:

$$\begin{aligned}
\exists^{\infty} t\, \mathfrak{A}(t,\dots) &\underset{df}{=} \forall \tau \exists t[t > \tau \,\&\, \mathfrak{A}(t,\dots)],\\
\forall^{\infty} t\, \mathfrak{A}(t,\dots) &\underset{df}{=} \exists \tau \forall t[t > \tau \rightarrow \mathfrak{A}(t,\dots)],\\
\exists t]_a^b \mathfrak{A}(t,\dots) &\underset{df}{=} \exists t[a \leqq t \leqq b \,\&\, \mathfrak{A}(t,\dots)],\\
\forall t]_a^b \mathfrak{A}(t,\dots) &\underset{df}{=} \forall t[a \leqq t \leqq b \rightarrow \mathfrak{A}(t,\dots)].
\end{aligned} \tag{3}$$

4. Limits* are defined in **I** as follows:

$$\begin{aligned}
a \in \lim q^{a,b,\dots,d} &\underset{df}{=} \exists^{\infty} t[q(t) = a],\\
\lim q^{a,b,c,d} = \{a,b\} &\underset{df}{=} a \in \lim q^{a,b,c,d} \,\&\, b \in \lim q^{a,b,c,d} \,\&\\
&\&\, \neg(c \in \lim q^{a,b,c,d}) \,\&\, \neg(d \in \lim q^{a,b,c,d}).
\end{aligned} \tag{4}$$

5. Equality of the δ-th letter in an ω-word x and the t-th letter in an ω-word y is defined as follows:

$$\begin{aligned}
x^{a,b,\dots,d}(\delta) = y^{a,b,\dots,d}(t) \underset{df}{=} [&x^{a,\dots,d}(\delta) = a \,\&\\
&\& y^{a,b,\dots,d}(t) = a \vee x^{a,b,\dots d}(\delta) = b \,\& y^{a,b,\dots,d}(t) = b \vee \dots\\
&\dots \vee x^{a,b,\dots,d}(\delta) = d \,\& y^{a,b,\dots,d}(t) = d].
\end{aligned} \tag{5}$$

* In the sense of Section 0.2 (page 8).

Henceforth we shall make use of these secondary concepts and their symbols without reservations; additional concepts will be introduced as the need arises.

We shall now indicate how to "embed" the metalanguages of finite trees (tables), sources (macrosources), regular and ω-regular expressions in the metalanguage **I**. An **I**-formula defining (in the appropriate sense) the same operators (language, ω-language) as a given formula $\mathfrak{A}$ of a metalanguage under consideration will be called *equivalent* to $\mathfrak{A}$.

I. *Finite trees.* As an example, consider the finite tree representing the transformation of the word *aab* into the word *abb*:

$$x(1) = a \& x(2) = a \& x(3) = b \rightarrow y(1) = a \& y(2) = b \& y(3) = b.$$

Clearly, the conjunction of implications of this type over all rows of the table (or paths of the tree) is a formula $\mathfrak{A}(x,y)$ which is satisfied by the very operator that satisfies the conditions described by the table. In simpler terms, one might say that this formula is a natural code for the table.

II. *Sources.* We first describe the procedure for "encoding" diagrams by **I**-formulas, which is analogous to that just presented for finite trees. Let B be a diagram labeled by letters $\{a,b,\ldots,d\}$ and having vertices (states) $\{q_1,q_2,\ldots,q_k\}$. The encoding formula will involve predicate variables $x^{a,b,\ldots,d}$ and $q^{q_1,q_2,\ldots,q_k}$ (denoted below by x and q) and an individual variable t. If B contains an edge going from q_i to q_j and labeled a (i.e., there is an instruction $q_i a \rightarrow q_j$), this is encoded by the formula

$$q(t) = q_i \& x(t) = a \& q(t+1) = q_j.$$

Denote the disjunction of all formulas of this type over all edges (instructions) by $\mathfrak{B}(q,x,t)$; this is the code of the diagram. Now consider a source $\langle B,Q',Q''\rangle$. Denote the disjunction $q(\tau) = q_\lambda \vee q(\tau) = q_\mu \vee \ldots$ over all q_λ, $q_\mu,\ldots$ in Q' by $q(\tau) \in Q'$; similarly for $q(\tau) \in Q''$. It is now easy to see that the formula

$$\mathfrak{A}(x,t) \underset{df}{=} \exists q \{q(1) \in Q' \& \forall \sigma]_1^t \mathfrak{B}(q,x,\sigma) \& q(t+1) \in Q''\}$$

is fictitious outside $[1,t]$; it expresses the statement: there is a path in the diagram B leading from Q' to Q'' and carrying the word $x(1)x(2)\ldots x(t)$. Consequently, this formula is equivalent to the source $\langle B,Q',Q''\rangle$, i.e., defines the language $\omega(B,Q',Q'')$, in the sense indicated in the preceding section.

III. *Macrosources.* After what has been said of sources, it should be quite clear that the ω-language $\Omega(B,Q_0,\mathfrak{C})$, where $\mathfrak{C} = \{\Gamma_1,\ldots,\Gamma_s\}$, is

defined by the I-formula

$$\mathfrak{A}(x) \underset{df}{=} \exists q\{q(1) \in Q_0 \& \forall t]_1^t \mathfrak{B}(q,x,t) \& \\ \& (\lim q = \Gamma_1 \vee \lim q = \Gamma_2 \vee \ldots \vee \lim q = \Gamma_s)\}$$

(in this sense, we shall say that the I-formula $\mathfrak{A}(x)$ is equivalent to the macrosource $\langle B, Q_0, \mathfrak{C}\rangle$).

IV. *Regular expressions.* A formula $\mathfrak{A}'$ equivalent to a regular expression $\mathfrak{A}$ is constructed inductively, simulating the construction of regular expressions. In order to carry through the induction, we shall have to prove an even stronger assertion, whose formulation entails some preliminary clarifications.

Consider a formula $\mathfrak{A}(x,\tau,t)$, containing a free predicate variable x and two free individual variables τ, t; assume that it satisfies the following two conditions:

(I) For any fixed τ_0 and t_0 such that $\tau_0 \leqq t_0$, the formula $\mathfrak{A}(x,\tau_0,t_0)$ is fictitious outside $[\tau_0,t_0]$.

(II) For any two pairs τ_0,t_0 and τ_1,t_1 such that $t_0 - \tau_0 = t_1 - \tau_1$, the formulas $\mathfrak{A}(x,\tau_0,t_0)$ and $\mathfrak{A}(x,\tau_1,t_1)$ define the same set of words of length $t_0 - \tau_0 + 1 = t_1 - \tau_1 + 1$ (therefore, the same set of words as the formula $\mathfrak{A}(x,1,t_0 - \tau_0)$). Obviously, for any fixed τ_0, the formula $\mathfrak{A}(x,1,t - \tau_0)$ with a single individual variable t defines the same language $\tilde{\mathfrak{A}}$, while the formula $\mathfrak{A}(x,\tau,t)$ itself, with two individual variables τ,t, expresses the statement: the word $x(\tau)x(\tau + 1)\ldots x(t)$ belongs to the language $\tilde{\mathfrak{A}}$.

We shall now use induction on the number of operations in the given regular expression $\mathfrak{A}$ to construct the corresponding formula $\mathfrak{A}'(x,\tau,t)$; a formula representing the required language is obtained by substituting the constant 1 for τ.

Basis. Let $\mathfrak{A} = a$; then

$$\mathfrak{A}'(x,\tau,t) \underset{df}{=} (x(\tau) = a) \& (\tau = t).$$

Induction step. Let $\mathfrak{A}_1'(x,\tau,t)$ and $\mathfrak{A}_2'(x,\tau,t)$ be the formulas corresponding to $\mathfrak{A}_1$ and $\mathfrak{A}_2$.

a) If $\mathfrak{A} = \mathfrak{A}_1 \vee \mathfrak{A}_2$, then

$$\mathfrak{A}' \underset{df}{=} \mathfrak{A}_1' \vee \mathfrak{A}_2'.$$

b) If $\mathfrak{A} = \mathfrak{A}_1 \cdot \mathfrak{A}_2$, then

$$\mathfrak{A}'(x,\tau,t) \underset{df}{=} \exists \rho[\tau \leqq \rho < t \& \mathfrak{A}_1'(x,\tau,\rho) \& \mathfrak{A}_2'(x,\rho + 1,t)].$$

The induction step for iteration closure is more complicated, though the

main idea is quite simple; this is natural, since one must use the apparatus of **I** to define the operation *. Let (q, ρ, σ) be an abbreviation for the formula

$$q(\rho) = a \& q(\sigma) = a \& \forall\, v[\rho < v < \sigma \rightarrow \neg (q(v) = a)],$$

which states that the letter a occurs at positions ρ and σ of the ω-word q but does not occur between these positions. Then:

c) If $\mathfrak{A} \underset{df}{=} \mathfrak{A}_1^*$, then

$$\mathfrak{A}'(x,\tau,t) \underset{df}{=} \exists q^{a,b}\{q^{a,b}(\tau) = a \& q^{a,b}(t+1) = a \,\&$$
$$\& \forall \rho\sigma[\tau \leqq \rho < \sigma \leqq t+1 \& (q,\rho,\sigma) \rightarrow \mathfrak{A}_1'(x,\rho,\sigma-1)]\}.$$

Explanation. The ω-word q contains occurrences of the letter a at those positions of the interval $[\tau, t+1]$ which split the word $x(\tau)x(\tau+1)\ldots x(t)$ into segments belonging to the language being iterated.

V. *ω-regular expressions.* The construction of equivalent **I**-formulas in this case is merely an extension of the procedure for regular expressions. We shall describe a procedure which yields somewhat more: for every ω-regular expression $\mathfrak{B}$ we shall inductively construct an **I**-formula $\mathfrak{B}'(x,\tau)$ stating that the "tail" $x(\tau)x(\tau+1)\ldots$ belongs to the ω-language represented by $\mathfrak{B}$. In particular, substituting 1 for τ we get the required **I**-formula without free individual variables.

Let $\mathfrak{A}$ be a regular expression and $\mathfrak{B}$ an ω-regular expression, and let $\mathfrak{A}'(x,\tau,t)$, $\mathfrak{B}'(x,\tau)$ denote the corresponding **I**-formulas. Then:

a) If $\mathfrak{C} = \mathfrak{A}^\infty$, then

$$\mathfrak{C}'(x,\tau) \underset{df}{=} \exists q^{a,b}\{q(\tau) = a \,\&\, \exists^\infty t(q(t) = a) \,\&$$
$$\& \forall \rho\sigma[\tau \leqq \rho < \sigma \& (q,\rho,\sigma) \rightarrow \mathfrak{A}'(x,\rho,\sigma-1)]\}$$

(compare with the formula for $\mathfrak{A}^*$).

b) If $\mathfrak{C} = \mathfrak{A} \cdot \mathfrak{B}$, then

$$\mathfrak{C}'(x,\tau) \underset{df}{=} \exists \rho[\tau \leqq \rho \,\&\, \mathfrak{A}'(x,\tau,\rho) \,\&\, \mathfrak{B}'(x,\rho+1)].$$

The main contents of this section may be summarized in the following theorem.

THEOREM 3.5. *There is an algorithm which, given any source (macro-source, regular expression, ω-regular expression), constructs an equivalent* **I***-formula.*

The essential factor, as far as we are concerned, does not figure in this formulation, though it is clear from the proof: the procedure involves an

extremely simple and natural encoding algorithm. This is convincing evidence for our claim that the metalanguage **I** is comprehensive and expressive.

III.8. Normal form

Two formulas with the same free variables are said to be equivalent if, for any substitution of appropriate objects (ω-words, natural numbers) for the free variables, they are either both true or both false. This definition extends in an obvious way to formulas whose free variables are not the same, by requiring that suitable formulas obtained from the former by adding the missing variables as "dummies" be equivalent in the narrow sense. For example, the formulas $\mathfrak{B}(x^{a,b})$ and $\mathfrak{A}(y^{b,c})$ are equivalent if the formulas

$$\mathfrak{B}(x^{a,b}) \& \forall t [y^{b,c}(t) = a \vee \neg (y^{b,c}(t) = a)]$$

and

$$\mathfrak{A}(y^{b,c}) \& \forall t [x^{a,b}(t) = a \vee \neg (x^{a,b}(t) = a)],$$

which are conjunctions of these formulas with suitable true formulas, are equivalent in the narrow sense.

For **I**-formulas one can define, besides the standard logical equivalences,* specific equivalences which follow from the interpretation adopted for the symbols of the language. For example, the formula $\forall t [x^{a,b}(t) = a]$ is equivalent (induction over the natural numbers!) to

$$x^{a,b}(1) = a \& \forall t [x^{a,b}(t) = a \rightarrow x^{a,b}(t+1) = a].$$

We cite one more specific equivalence in **I**, which will be used later. Let $\mathfrak{N}(\ldots t \ldots)$ be a formula with a free individual variable t (and perhaps other free variables). Assume, moreover, that $\mathfrak{N}(\ldots t \ldots)$ does not contain the predicate variable $p^{a,b}$ or the individual variable λ. Then it is not hard to see that $\exists t \mathfrak{N}(\ldots t \ldots)$ is equivalent to

$$\exists p^{a,b} \{\forall t [p(t) = a \rightarrow \mathfrak{N}(\ldots t \ldots)] \& \exists \lambda (p(\lambda) = a)\}. \tag{1}$$

Let $\mathfrak{A}$ and $\mathfrak{B}$ be equivalent **I**-formulas. It is evident that if $\mathfrak{A}$ defines a certain language (ω-language, operator), then $\mathfrak{B}$ defines the same language (ω-language, operator), and any operator satisfying $\mathfrak{A}$ also satisfies $\mathfrak{B}$. Conse-

* For example, $\forall t [x^{a,b}(t) = a]$ is equivalent to $\neg \exists t (\neg (x^{a,b}(t) = a))$.

quently, in solving the synthesis problem it is legitimate to replace formulas by equivalent formulas where necessary.

Our immediate goal is to give an algorithm for conversion of any **I**-formula into an equivalent **I**-formula of a special type (the so-called normal form), which is more convenient for further investigation of the synthesis problem.

Formulas of the following two types will be called *elementary*:

1) Elementary formulas of the first type: atomic formulas, such as $x(s) = a$, $x(\tau + 1) = b$, etc.

2) Elementary formulas of the second type: formulas of the form $\exists t\,\mathfrak{A}(t)$, where $\mathfrak{A}(t)$ is a formula built up from atomic formulas, each containing the free individual variable t, by means of the logical connectives. Examples:

$$\exists t[x(t) = a \,\&\, \neg(y(t) = b)],$$

$$\exists t[x(t) = a_2 \& y(t) = b_1 \& y(t + 3) = b_2].$$

Let $\Gamma_1, \Gamma_2, \ldots, \Gamma_s$ be elementary formulas, and $\Phi(\Gamma_1, \Gamma_2, \ldots, \Gamma_s)$ a formula built up from $\Gamma_1, \Gamma_2, \ldots, \Gamma_s$ by means of logical connectives alone. We shall call $\Phi(\Gamma_1, \Gamma_2, \ldots, \Gamma_s)$ a *simple formula* with components $\Gamma_1, \Gamma_2, \ldots, \Gamma_s$. For example,

$$x(3) = a_1 \to \exists t[x(t) = a_2 \& y(t) = \\ = b_1 \& y(t + 3) = b_2] \vee \exists t[x(t + 5) = a_1] \qquad (2)$$

is a simple formula with three components:

$$x(3) = a_1;$$

$$\exists t[x(t) = a_2 \& y(t) = b_1 \& y(t + 3) = b_2];$$

$$\exists t[x(t + 5) = a_1].$$

It is clear from the definition that, in any simple formula (in particular, in an elementary formula), all occurrences of predicate variables are free.

Finally, a formula is said to be in *normal form* if it is either simple or is derived from a simple formula by quantification over some of its predicate variables. Thus, the general appearance of a normal form is

$$\{x, y, \ldots\}\, \Phi(\Gamma_1, \Gamma_2, \ldots, \Gamma_s), \qquad (3)$$

where $\{x, y, \ldots\}$ denotes the quantifier prefix before the simple formula $\Phi(\Gamma_1, \Gamma_2, \ldots, \Gamma_s)$. For example, quantifying over y in (2), we get the normal form

$$\exists y[x(3) = a_1 \to \exists t(x(t) = a_2 \,\&\, y(t) = \\ = b_1 \,\&\, y(t+3) = b_2) \vee \exists t(x(t+5) = a_1)], \qquad (4)$$

which is no longer a simple formula.

THEOREM 3.6 (REDUCTION TO NORMAL FORM). *There is an algorithm which, given any* **I***-formula* $\mathfrak{A}$, *constructs an equivalent normal form* $\mathfrak{A}'$.

Proof. The reduction algorithm falls into the following steps:

A. *Reduction to prenex form*, i.e., the process bringing the quantifiers forward to the head of the formula, for both predicate and individual variables (this is a standard procedure).

The result of step A is a formula $[w]\mathfrak{B}$, where

(I) the prefix $[w]$ generally contains quantifiers over variables of both types;

(II) the quantifier-free part (*matrix*) is a simple formula all of whose components are atomic formulas.

B. Suppose that to the right of some individual quantifier (say $\exists t$) in the prefix $[w]$ there are m predicate quantifiers. We shall call this individual quantifier a *trespasser* generating m *inversions*. In this step of the algorithm we wish to arrange the quantifiers in such a way that there are no trespassers, while the matrix, as before, has property (II). It will suffice to prove that the number of trespassers can be reduced by one. Consider the right-most trespasser; to fix ideas, let us assume that it is the individual existential quantifier $\exists t$ (the procedure for universal quantifiers is analogous).

Suppose that the prefix $[w]$ contains in all μ trespassers, and the trespasser $\exists t$ generates m inversions. We shall convert this formula into an equivalent prenex formula whose prefix contains at most the same number of trespassers but the right-most one generates only $m - 1$ inversions. m-fold repetition of this procedure reduces the number of trespassers by one. Thus, step by step, one can reduce the number of trespassers and finally eliminate them altogether.

The right-most trespasser $\exists t$ is immediately followed by a predicate quantifier over some variable q. If this is $\exists q$, then $\exists q$ and $\exists t$ may be interchanged since they are quantifiers of the same type, and this reduces the number of inversions by one. Now suppose that the quantifier following $\exists t$ is the universal quantifier $\forall q$, i.e., the prefix has the form $[\xi]\exists t\forall q[\eta]$. Let $p^{a,b}$ be a predicate variable not occurring in our formula. We now use the equivalence (1), with the subformula $\exists t\forall q[\eta]\mathfrak{B}$ taking the role of $\exists t\mathfrak{N}(\ldots t\ldots)$. This means that we can replace $[\xi]\exists t\forall q[\eta]\mathfrak{B}$ by the equi-

valent formula

$$[\xi]\exists p(\forall t[p(t) = a \rightarrow \forall q[\eta]\mathfrak{B}]\,\&\,\exists\lambda(p(\lambda) = a)), \tag{5}$$

which can be put in prenex form:

$$[\xi]\exists p\forall t\forall q[\eta]\exists\lambda\{p(t) = a \rightarrow \mathfrak{B}\,\&(p(\lambda) = a)\}. \tag{6}$$

The prefix in (6) contains two more quantifiers than that of the original formula, but the number of trespassers is exactly the same. However, now the last inversion is generated by the *universal* quantifier $\forall t$, which is also followed by a universal quantifier. Interchanging these quantifiers, we get the prefix $[\xi]\exists p\forall q\forall t\,[\eta]\exists\lambda$, where $\forall t$ generates one less inversion than before.

C. The last step is the so-called *Behmann procedure*, which is applied to a formula without inversions $[\xi]\,[\alpha]\mathfrak{B}$ and reduces it to normal form. $[\xi]$ denotes the "predicate" part of the prefix, $[\alpha]$ the "individual" part, and $\mathfrak{B}$ a simple formula. Without loss of generality, let the last quantifier in $[\alpha]$ be an existential quantifier, i.e., $[\alpha]$ has the form $[\alpha']\exists\tau$; we shall convert $[\xi]\,[\alpha']\exists\tau\mathfrak{B}$ into an equivalent formula $[\xi]\,[\alpha']\mathfrak{B}'$, where $\mathfrak{B}'$ is again a simple formula (if the last quantifier in $[\alpha]$ is a universal quantifier $\forall\tau$, we need only replace $\forall\tau\mathfrak{B}$ by $\neg\exists\tau\,\neg\mathfrak{B}$). Repeating this process as many times as there are quantifiers in $[\alpha]$, we reduce $[\xi]\,[\alpha]\mathfrak{B}$ to a normal form $[\xi]\mathfrak{B}^*$.

Thus, consider the formula $\exists\tau\mathfrak{B}$. Express $\mathfrak{B}$ as a disjunction $\mathfrak{B}_1 \vee \mathfrak{B}_2 \vee \ldots \vee \mathfrak{B}_k$, where each $\mathfrak{B}_i$ is a conjunction of atomic formulas or their negations and elementary formulas of the second type, i.e., $\mathfrak{B}_i$ is $\mathfrak{B}_i^1\,\&\,\mathfrak{B}_i^2\,\&\ldots\&\,\mathfrak{B}_i^\nu\,\&\ldots\mathfrak{B}_i^\mu$. Then $\exists\tau\mathfrak{B}$ is equivalent to $\exists\tau\mathfrak{B}_1 \vee \exists\tau\mathfrak{B}_2 \vee \ldots. \vee \exists\tau\mathfrak{B}_k$. Now let $\mathfrak{B}_i^1, \ldots, \mathfrak{B}_i^\nu$ be the conjunctive terms which contain the variable τ (the variable in the quantifier $\exists\tau$), and $\mathfrak{B}_i^{\nu+1}, \ldots, \mathfrak{B}_i^\mu$ all other terms. It is clear that $\mathfrak{B}_i^1, \ldots, \mathfrak{B}_i^\nu$ must be atomic formulas, while $\mathfrak{B}_i^{\nu+1}, \ldots, \mathfrak{B}_i^\mu$ may be arbitrary elementary formulas or their negations. Then $\exists\tau\mathfrak{B}_i$ is equivalent to $\exists\tau[\mathfrak{B}_i^1\,\&\ldots\&\,\mathfrak{B}_i^\nu]\,\&\,\mathfrak{B}_i^{\nu+1}\,\&\ldots\&\,\mathfrak{B}_i^\mu$, where $\exists\tau[\mathfrak{B}_i^1\,\&\ldots\&\,\mathfrak{B}_i^\nu]$ is a new elementary formula. As a result, we have converted $\exists\tau\mathfrak{B}$ into the required equivalent formula $\mathfrak{B}'$. Consequently, the formula $[\xi]\,[\alpha']\,\exists\tau\mathfrak{B}$ is equivalent to the formula $[\xi]\,[\alpha']\mathfrak{B}'$. This completes the proof.

III.9. Synthesis of an automaton representing the ω-language defined by an I-formula

Given an arbitrary I-formula

$$\mathfrak{A}(x_1^{a,b,\ldots,f}, \ldots, x_m^{k,h,\ldots,r}),$$

with m free predicate variables and no free individual variables. To simplify the exposition, we shall omit the superscripts indicating the alphabets $A_1, A_2, \ldots, A_m$ associated with the predicate variables, and henceforth write $\mathfrak{A}(x_1, \ldots, x_m)$. As usual, $x_1 \ldots x_m \mathfrak{A}(x_1, \ldots, x_m)$ will denote the ω-language over the cartesian product $A_1 \times A_2 \times \ldots \times A_m$ defined by the formula $\mathfrak{A}(x_1, \ldots, x_m)$. In accordance with the general formulation of the synthesis problem, our aims are as follows:

A′. *To determine whether the ω-language*

$$\widehat{x_1 \ldots x_m} \mathfrak{A}(x_1, \ldots, x_m)$$

is finite-state.

C′. *If so, to construct a finite macroanchored automaton representing the ω-language.*

Using the known properties of the metalanguage **I** presented in the preceding sections, together with Theorems 1.1, 1.11, 1.12 on the closure properties of the class of finite-state ω-languages (Sections I.1 and I.8), we are now in a position to state the following important theorem, which contains a full solution to Problems A′ and C′.

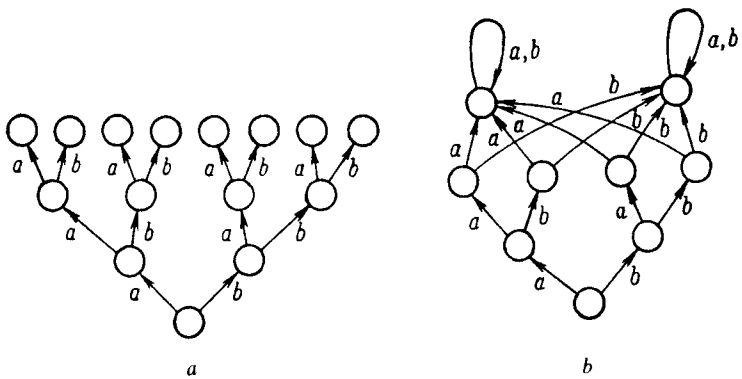
THEOREM 3.7. *For any* **I***-formula $\mathfrak{A}(x_1, \ldots, x_m)$ the ω-language $\widehat{x_1 \ldots x_m}$ $\mathfrak{A}(\widehat{x_1, \ldots, x_m})$ is representable in a finite automaton. There is an algorithm (synthesis algorithm) which, given any such formula, constructs a suitable automaton.*

Proof. We shall describe the algorithm, which falls into two parts. In the first part, the **I**-formula is reduced to normal form (Theorem 3.6). Suppose that $\mathfrak{A}(x_1, \ldots, x_m)$ is actually a formula in the extended metalanguage, in which the basic symbols and atomic formulas are supplemented by secondary (auxiliary) expressions, such as equality between terms, the quantifiers $\exists^\infty$, $\forall^\infty$, etc. One first eliminates these secondary expressions, replacing them by their defining expressions, and then carries out the reduction to normal form.

The second part of the algorithm is the construction of an automaton $\langle \mathfrak{M}, q_0, \mathfrak{C} \rangle$ corresponding to a given normal form $\mathfrak{A}$. The construction proceeds in steps. One first constructs automata for the elementary subformulas of the normal form; then, by induction on the structure of the normal form $\mathfrak{A}$, more involved automata are constructed for larger subformulas until the required automaton for the entire formula is obtained.

We first consider an elementary subformula of $\mathfrak{A}$. There are two cases, depending on whether the elementary subformula is of the first or the second type (Cases I and II, respectively).

Case I. The elementary subformula is an atomic formula. Since $\mathfrak{A}(x_1, \ldots, x_m)$ contains no free occurrences of individual variables, this subformula is necessarily $x(\gamma) = a$ for some natural number γ. The ω-language defined by this subformula is clearly the set of all words $x = x(1)x(2)\ldots x(\gamma)x(\gamma+1)\ldots$ in which the letter a appears at the γ-th position. A representing automaton may be constructed as follows. Construct a tree of height γ with root q_0 over the input alphabet X (Figure 29*a* illustrates the construction for $\gamma = 3, X = \{a,b\}$). All vertices of the highest rank γ which are endpoints of edges labeled a are merged into a single absorbing vertex q_a;* all other vertices of rank γ are merged into another absorbing vertex q'. The resulting automaton diagram (Figure 29*b*) is "macroanchored" by making q_0 the initial state and the singleton $\{q_a\}$ the only limit macrostate. This completes the construction for Case I.



a *b*

Figure 29

Case II. The elementary subformula is $\exists\tau\mathfrak{B}$, where $\mathfrak{B}$ is a quantifier-free formula in which every component contains the individual variable τ. To simplify the exposition we shall assume that $\mathfrak{B}$ contains only two predicate variables x and y, associated with two-letter alphabets $\{a_1,a_2\}$ and $\{b_1,b_2\}$, respectively; the detailed notation $\mathfrak{B}(x,y,\tau)$ indicates all free variables of $\mathfrak{B}$. Let γ be the maximal constant γ_i such that $\exists\tau\mathfrak{B}$ contains the term $\tau + \gamma_i$. It is immediate that, for any fixed natural number τ_0, the quantifier-free formula $\mathfrak{B}(x,y,\tau_0)$ contains atomic subformulas of only two types: $x(\tau_0 + \gamma_i) = a_1$, $x(\tau_0 + \gamma_i) = a_2$, $y(\tau_0 + \gamma_i) = b_1$ or $y(\tau_0 + \gamma_i) = b_2$, where

* Recall that a state q in an automaton (vertex in a diagram) is an absorbing state if any input letter takes q to q (all edges issuing from the vertex lead back to it).

$0 \leqq \gamma_i \leqq \gamma$. It follows directly that $\mathfrak{B}(x, y, \tau_0)$ is fictitious outside $[\tau_0, \tau_0 + \gamma]$; thus, the question as to whether a word

$$\xi = \langle x(1)y(1)\rangle \ \langle x(2)y(2)\rangle \ldots \langle x(\tau_0)y(\tau_0)\rangle \ldots \langle x(\tau_0 + \gamma)y(\tau_0 + \gamma)\rangle \ldots$$

belongs to the language $\hat{x}\hat{y}\mathfrak{B}(x, y, \tau_0)$ depends entirely on the segment of length $\gamma + 1$ from position τ_0 to position $\tau_0 + \gamma$.

Clearly, the set of all words p of length $\gamma + 1$ such that $\xi_{\tau_0}^{\tau_0+\gamma}{}_1 = p$ implies $\xi \in \hat{x}\hat{y}\mathfrak{B}(x, y, \tau_0)$ is independent of the choice of τ_0; denote this set by K. It is also clear that, for any word p of length $\gamma + 1$, one can effectively determine whether it belongs to K: it suffices to compute the truth value of the formula $\mathfrak{B}(x, y, 1)$ on the assumption that any atomic subformula $x(1 + \gamma_i) = a_j$ or $y(1 + \gamma_i) = b_j$ is true if and only if the $(1 + \gamma_i)$-th symbol in p has first component a_j or second component b_j, respectively. For example, the set K for the formula

$$\exists\tau[x(\tau) = a_1 \& (y(\tau) = b_1 \leftrightarrow x(\tau + 1) = a_2)]$$

consists of four words: $\langle a_1b_1\rangle \ \langle a_2b_1\rangle$, $\langle a_1b_1\rangle \ \langle a_2b_2\rangle$, $\langle a_1b_2\rangle \ \langle a_1b_1\rangle$, $\langle a_1b_2\rangle \ \langle a_1b_2\rangle$. Finally, note that the ω-language $\hat{x}\hat{y}\exists\tau\mathfrak{B}(x, y, \tau)$ is the union of the ω-languages $\hat{x}\hat{y}\mathfrak{B}(x, y, 1)$, $\hat{x}\hat{y}\mathfrak{B}(x, y, 2), \ldots$ Having made these preliminary remarks, it is easily seen that an automaton representing the ω-language $\hat{x}\hat{y}\exists\tau\mathfrak{B}(x, y, \tau)$ may be effectively constructed as follows (the construction is illustrated in Figure 30*a*,*b* for the formula

$$\exists\tau[x(\tau) = a_1 \& (y(\tau) = b_1 \leftrightarrow x(\tau + 1) = a_2)]).$$

Construct a tree of height $\gamma + 1$ over the input alphabet $X \times Y$, and divide the vertices of rank $\gamma + 1$ into two groups: (I) vertices at the end of paths of length $\gamma + 1$ which carry words from K (indicated by crosses in Figure 30*a*); (II) all others. Merge all vertices of the first group into an absorbing vertex σ. Now consider some vertex σ' of type (II), and the word

$$\langle x(1)y(1)\rangle \langle x(2)y(2)\rangle \ldots \langle x(\gamma)y(\gamma)\rangle \langle x(\gamma + 1)y(\gamma + 1)\rangle,$$

carried by the path leading to it (Figure 30*b*). Draw an edge from σ', labeled with the pair $x(\gamma + 2)y(\gamma + 2)$, to a vertex which depends on the word

$$\langle x(2)y(2)\rangle \langle x(3)y(3)\rangle \ldots \langle x(\gamma + 1)y(\gamma + 1)\rangle \langle x(\gamma + 2)y(\gamma + 2)\rangle$$

of length $\gamma + 1$ in the following way. If this word belongs to K, the edge leads to the absorbing vertex σ. Otherwise it leads to the vertex σ'' of rank $\gamma + 1$ at the end of the path (from the root) carrying the above word. The sum result of this construction is an automaton which, for initial state

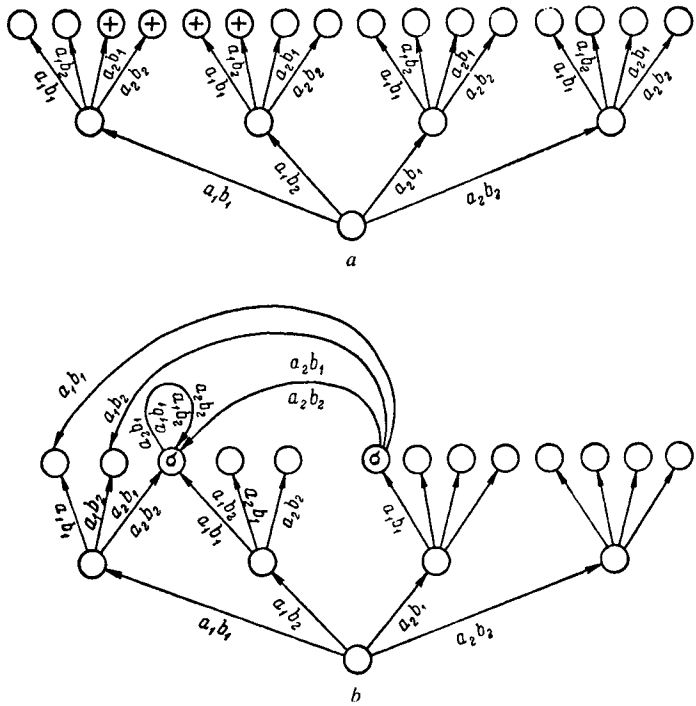


Figure 30

q_0 and final state σ, represents the language consisting of all words which have subwords in K. Hence it is clear that an automaton representing the ω-language $\hat{x}\hat{y}\exists\tau\mathfrak{B}(x,y,\tau)$ is obtained by the following macroanchoring: the root of the original tree is the initial state, and the only limit macrostate consists of the absorbing vertex σ.

We have thus effectively constructed an automaton for each elementary subformula of the normal form $\mathfrak{A}$. Since the latter is built up from its elementary subformulas via logical connectives and quantification over predicate variables, all that remains is to carry through induction for these logical operations.

Let $\mathfrak{A}'$ and $\mathfrak{A}''$ be formulas for which automata $\langle\mathfrak{M}',q_0',\mathfrak{C}'\rangle$ and $\langle\mathfrak{M}'',q_0'',\mathfrak{C}''\rangle$ have already been constructed. We shall now show how to construct an automaton for the required three types of formula.

Negation. The formula $\neg\mathfrak{A}'(x,y,\ldots)$ defines the ω-language $\neg\Omega(\mathfrak{M}', q_0', \mathfrak{C}')$; the construction of an automaton representing it is described in Theorem 1.1.

Disjunction. If the formulas $\mathfrak{A}'$ and $\mathfrak{A}''$ contain the same free predicate

variables $x, y, z, \ldots$, the disjunction $\mathfrak{A}'(x, y, z, \ldots) \vee \mathfrak{A}''(x, y, z, \ldots)$ defines the ω-language $\Omega(\mathfrak{M}', q_0', \mathfrak{C}') \vee \Omega(\mathfrak{M}'', q_0'', \mathfrak{C}'')$, and a representing automaton may be constructed as in Theorem 1.1. But if the formulas contain different free variables (e.g., $\mathfrak{A}'(x, y)$ and $\mathfrak{A}''(x, z)$) one must first add dummy variables in order to convert them into formulas $\tilde{\mathfrak{A}}', \tilde{\mathfrak{A}}''$ equivalent to $\mathfrak{A}'$ and $\mathfrak{A}''$, respectively, and containing the same free variables (i.e., $\tilde{\mathfrak{A}}' = \tilde{\mathfrak{A}}'(x, y, z)$, $\tilde{\mathfrak{A}}'' = \tilde{\mathfrak{A}}''(x, y, z)$). Now the formula $\tilde{\mathfrak{A}}' \vee \tilde{\mathfrak{A}}''$ clearly defines the same ω-language as $\mathfrak{A}' \vee \mathfrak{A}''$, and this brings us back to the preceding situation. In actual fact, there is no need to construct the formulas $\tilde{\mathfrak{A}}'$ and $\tilde{\mathfrak{A}}''$, since the ω-languages that they define are cylindrifications of the ω-languages defined by $\mathfrak{A}'$ and $\mathfrak{A}''$. Therefore one can use Theorem 1.1 to construct automata $\mathfrak{M}'$, $\mathfrak{M}''$ for these cylindrifications, based on $\langle \mathfrak{M}', q_0', \mathfrak{C}' \rangle$, $\langle \mathfrak{M}'', q_0'', \mathfrak{C}'' \rangle$; one can then construct the required automaton for disjunction.

Quantification. The formula $\exists x \mathfrak{A}'(x, y, \ldots)$ defines the projection of the ω-language $\Omega(\mathfrak{M}', q_0', \mathfrak{C}')$. A representing automaton is constructed from $\langle \mathfrak{M}', q_0', \mathfrak{C}' \rangle$ by deleting all x-labels (this was indicated in detail at the beginning of Section I.5), and then determinizing the resulting macrosource.

This completes the proof of the theorem.

It is clear from the preceding description that our algorithm is extremely complicated. Even taken separately, its component parts are quite unwieldy (e.g., reduction to normal form). Special attention should be paid to the last induction step—elimination of the existential quantifier. Every quantifier elimination comprises a series of complex algorithms—those involved in determinization of a macrosource (e.g., the algorithms used in the Concatenation and Strong Iteration Theorems in Sections I.9, I.10). Nevertheless, the algorithm is quite powerful; some idea of its power may be derived from the following fact. As previously indicated (Section III.7), it is very easy to construct an I-formula defining the ω-language represented by a given macrosource or ω-regular expression; the above synthesis algorithm must therefore be regarded as more general than those for the metalanguages of macrosources and ω-regular expressions.

The significance of the theorem is fundamental; it immediately implies important corollaries concerning decision problems of mathematical logic. Mathematical logic attaches great importance to decision problems for various formal languages. The decision problem for a formal language L is as follows: to construct an algorithm which, given any sentence of the language, will establish whether it is true or not. It is well known that in many cases no such algorithm exists. The following theorem is a simple corollary of Theorem 3.7.

THEOREM 3.8. *There is an algorithm (decision procedure) which, for any* **I***-sentence, determines whether it is true or not.*

Proof. Reduce the sentence $\mathfrak{A}$ to normal form. The latter must contain predicate quantifiers; to fix ideas, suppose that $\mathfrak{A}$ has the form $\exists x\mathfrak{B}(x)$, and let $\mathfrak{M}$ be an automaton representing the language $\hat{x}\mathfrak{B}(x)$. Then it is obvious that the formula $\exists x\mathfrak{B}(x)$ is true if and only if the ω-language represented by the automaton $\mathfrak{M}$ is not empty. To verify the latter assertion one need only construct the automaton $\mathfrak{M}$ according to Theorem 3.7, and (effectively) determine whether the ω-language that it represents is empty or not (see Theorem 1.5). Q.E.D.

Thus, consideration of automata is seen to be decisive in establishing a very important, purely "logical" proposition.

III.10. Synthesis of an automaton according to conditions imposed on an operator or a language

I. We first consider the problem of synthesizing a finite automaton when the conditions imposed on the operator that it is to realize are expressed by **I**-formulas:

It is required to construct an algorithm which, given any formula $\mathfrak{A}(x,y)$,

(I) determines whether there is a finite-state operator $Tx = y$ satisfying the formula $\mathfrak{A}(x,y)$ (i.e., uniformizing the ω-language $\hat{x}\hat{y}\mathfrak{A}(x,y)$), and, if so,

(II) constructs a finite automaton realizing the operator.

REMARK. For simplicity's sake we shall restrict ourselves to **I**-formulas in two free variables. The statement and solution of the problem carry over in an obvious way to formulas $\mathfrak{A}(x_1,\ldots,x_m,y_1,\ldots,y_n)$ and operators T transforming an m-tuple of input ω-words $x_1 \ldots x_m$ into an n-tuple of output ω-words $y_1 \ldots y_n$ (or, equivalently, an input ω-word over the cartesian product of the corresponding alphabets into an output ω-word over the product of the output alphabets).

We already know that the ω-language $\hat{x}\hat{y}\mathfrak{A}(x,y)$ is representable in a finite automaton $\langle\mathfrak{M},q_0,\mathfrak{C}\rangle$, which can be effectively constructed from the given formula $\mathfrak{A}(x,y)$ (Theorem 3.7). But once this automaton has been constructed, the problem formulated above reduces to the uniformization problem for the ω-language $\Omega(\mathfrak{M},q_0,\mathfrak{C})$ (Section II.5). Theorem 2.8 provides a complete solution to the uniformization problem. Thus, the latter theorem and the preceding remarks (based on Theorem 3.7) directly imply

THEOREM 3.9. *Let* $\mathfrak{A}(x, y)$ *be an arbitrary* **I***-formula. If* $\mathfrak{A}(x, y)$ *is satisfied by some nonanticipatory operator* $y = Tx$, *it is also satisfied by a finite-state operator. There is an algorithm which, given the formula* $\mathfrak{A}(x, y)$,

(*I*) *determines whether there exists a finite-state operator* $y = T''x$ *satisfying it*; *if so,*

(*II*) *constructs an automaton realizing it*; *if not,*

(*III*) *constructs a finite automaton with delay realizing an operator* $T'y$ *that satisfies* $\neg\, \mathfrak{A}(x, y)$.

Note that if the answer to part (I) is negative, part (III) yields additional information: for any finite automaton, one can effectively construct a counterexample which shows that the corresponding operator indeed does not satisfy the formula $\mathfrak{A}(x, y)$.

Let us clarify this in detail in the game-theoretic interpretation (Section II.8), in which the operators T'' and T' are regarded as strategies for white and black, respectively. Assume that there is no finite-state operator satisfying the formula $\mathfrak{A}(x, y)$. Then, for any finite-state operator $y = T''x$, there exists an ω-word x such that the formula $\mathfrak{A}(x, T''x)$ is false. Assertion (III) indicates that, once an automaton realizing the operator T'' is available, the pair $\langle x, T''x\rangle$ is effectively constructible. Indeed, having constructed in accordance with (III) a finite automaton realizing the operator $x = T'y$, we can effectively construct a game history $\langle T', T''\rangle$ in which white loses. This history is a periodic ω-word $\langle x, y\rangle$ satisfying the formula $\neg\, \mathfrak{A}(x, y)$, and so not satisfying $\mathfrak{A}(x, y)$.

II. Before formulating the next problem, we recall that our definition of languages employed only formulas $\mathfrak{A}(x, t)$ which are fictitious outside a finite interval: for any fixed t_0, the truth of the formula $\mathfrak{A}(x, t_0)$ depends only on the word $x(1) \ldots x(t_0)$. In this situation, the language defined by the formula is precisely the set of words $x(1)x(2) \ldots x(t_0)$ ($t_0 = 1, 2, 3, \ldots$) for which the formula is true. The problem of synthesizing an automaton from the language that it is to represent is as follows.

It is required to construct an algorithm which, given any **I**-formula $\mathfrak{A}(x, t)$,

(I) determines whether it is fictitious outside a finite interval; if so,

(II) determines whether the language that it defines is representable in a finite automaton; if so,

(III) constructs an appropriate finite automaton.

A complete solution is implicit in the following theorem.

THEOREM 3.10. *Any* **I**-*formula* $\mathfrak{A}(x,t)$ *fictitious outside a finite interval defines a language which is representable in a finite automaton. There is an algorithm which, given a formula* $\mathfrak{A}(x,t)$,

(*I*) *determines whether it is fictitious outside a finite interval*; *if so,*

(*II*) *constructs a finite automaton representing the corresponding language.*

Proof. Note that the fictitiousness condition is satisfied if and only if the following **I**-sentence is true:

$$\forall x'x''t[\forall\sigma(\sigma \leqq t \to x'(\sigma) = x''(\sigma)) \to (\mathfrak{A}(x',t) \leftrightarrow \mathfrak{A}(x'',t))].$$

The truth of this formula may be determined according to Theorem 3.8. Consequently, the fictitiousness condition is effectively decidable.

Assume that the fictitiousness condition holds, and so the formula $\mathfrak{A}(x,t)$ defines a certain language $\mathfrak{B}$; we wish to construct a finite automaton $\langle \mathfrak{M}, q_0, Q' \rangle$ representing this language. Let $y^{a,b}$ be a variable not occurring in $\mathfrak{A}(x,t)$. Then, based on the formula $\mathfrak{A}(x,t)$, construct an **I**-formula $\mathfrak{B}(x,y)$:

$$\mathfrak{B}(x,y) \underset{df}{=} \forall t[y^{a,b}(t) = a \leftrightarrow \mathfrak{A}(x,t)]. \qquad (\#)$$

Obviously, this formula is satisfied by a unique nonanticipatory operator T—the operator which represents the language $\mathfrak{B}$ by output a. By Theorem 3.9 this nonanticipatory operator is finite-state, and an appropriate initialized automaton $\langle \mathfrak{N}, \pi_0 \rangle$ may be effectively constructed. Finally, one can effectively replace the automaton $\langle \mathfrak{N}, \pi_0 \rangle$ by an outputless automaton $\langle \mathfrak{M}, q_0, Q' \rangle$ representing the same language $\mathfrak{B}$(Section II.5). This completes the proof.

III.11. Cases without a synthesis algorithm

We were led to our study of the metalanguage **I** by the need for a more comprehensive metalanguage, convenient for formulation of the conditions imposed on the synthesized automaton. In Section III.7 we presented arguments in favor of the thesis that the metalanguage **I** is in fact more expressive than the frequently used metalanguages of trees, regular expressions and ω-regular expressions. It would be interesting to try to extend the metalanguage, with a view to enhancing its expressive power. However, in so doing one must remember that, in general, the synthesis algorithm becomes more complex and, moreover, may not even exist. In this section we intend to analyze a few situations in which metalanguages have no syn-

thesis algorithm. Comparison of these languages with **I** indicates that one can hardly hope for any significant extension of the latter without forfeiting the synthesis algorithm.

As in other, similar cases (see, e.g., Section II.6 and Section III.3), our proofs of algorithmic unsolvability will employ a reduction method. Namely, we show that some problem which is known to be algorithmically unsolvable may be reduced to the synthesis problem for the relevant metalanguage. Any metalanguage for which this is possible has no synthesis algorithm.

Returning to the metalanguage **I**, we recall that its terms may have the form t (variable), c (constant, e.g., 5) or $t + c$, but terms $t + \tau$ (sum of two variables) are inadmissible. This is because **I** has a unary function symbol $\phi(\ )$ which is interpreted as addition of one, but there is no addition operation for arbitrary natural numbers. One might be tempted to consider an extension $\mathbf{I}_+$ of the metalanguage **I**, differing from the latter only in that it provides for addition of arbitrary natural numbers. Thus, $\mathbf{I}_+$ also admits all terms of type $t + \tau$, $t + \tau + u + 7$, $2t + u$ (where t, τ, u are variables), and so on. It is easy to see that the class of formulas of $\mathbf{I}_+$ is larger than that of **I**, and the formulas of $\mathbf{I}_+$ can describe ω-languages which are undefinable by formulas of **I**. For example, let L be the ω-language over the alphabet $\{a,b\}$ which consists of the single ω-word $x = x(1)x(2)\ldots$ such that $x(t) = a$ if and only if t is an exact square. Any ω-word which is the only element of a finite-state ω-language must be periodic (see Section I.4), and so L cannot be represented in a finite automaton. Nevertheless, L is definable by a formula $\mathfrak{A}(x^{a,b})$ of $\mathbf{I}_+$. First note that if $t_1 < t_2 < t_3$ are consequtive exact squares, then $t_3 + t_1 = 2t_2 + 2$. A suitable formula $\mathfrak{A}(x^{a,b})$ is thus

$$\mathfrak{A} \underset{df}{=} x^{a,b}(1) = a \& x^{a,b}(2) = b \& x^{a,b}(3) = b \& x^{a,b}(4) =$$

$$= a \& \forall t_1 t_2 \{t_1 < t_2 \& x^{a,b}(t_1) = a \& x^{a,b}(t_2) =$$

$$= a \& \forall \tau [t_1 < \tau < t_2 \rightarrow x^{a,b}(\tau) = b] \rightarrow \exists t_3 [x^{a,b}(t_3) =$$

$$= a \& \forall \rho (t_2 < \rho < t_3 \rightarrow x^{a,b}(\rho) = b) \& t_3 + t_1 = 2t_2 + 2]\}.$$

The formula $\mathfrak{A}(x^{a,b})$ may also be used to show that many important monadic and polyadic predicates, whose arguments range over the natural numbers, are definable in $\mathbf{I}_+$.

The monadic predicate "t is an exact square" is defined by the formula

$$\mathfrak{B}_1(t) \underset{df}{=} \exists x^{a,b} [\mathfrak{A}(x^{a,b}) \& x^{a,b}(t) = a].$$

The binary predicate "$t = \rho^2$" is easily defined, seeing that $\rho^2 + 2\rho + 1$ is the exact square following ρ^2:

$$\mathfrak{B}_2(t,\rho) \underset{df}{=} \mathfrak{B}_1(t) \,\&\, \exists\sigma[\mathfrak{B}_1(\sigma) \,\&\, t < \sigma \,\&$$
$$\&\, \forall\tau(t < \tau < \sigma \rightarrow \neg\mathfrak{B}_1(\tau)) \,\&\, t + \rho + \rho + 1 = \sigma].$$

Since the ternary predicate $w = u \cdot v$ is expressible as $(u + v)^2 = u^2 + v^2 + 2w$, it may be defined by the $\mathbf{I}_+$-formula

$$\mathfrak{B}_3(w,u,v) \underset{df}{=} \exists\tau_1\tau_2\tau_3\tau_4[u + v = \tau_1 \,\&\, \mathfrak{B}_2(\tau_2,\tau_1) \,\&$$
$$\&\, \mathfrak{B}_2(\tau_3,u) \,\&\, \mathfrak{B}_2(\tau_4,v) \,\&\, \tau_2 = \tau_3 + \tau_4 + 2w].$$

One can thus define the predicates $w = u + v$ and $w = u \cdot v$ in $\mathbf{I}_+$. Now it is well known that any recursive predicate can be defined by a formula built up from the formulas $w = u \cdot v$, $w = u + v$ (regarded as atomic formulas) using the logical connectives ($\&$, $\vee$, $\neg$, $\rightarrow$) and quantifiers over the variables $w,u,v,\ldots$ (ranging over the natural numbers). Gödel's Theorem states that there is no decision procedure for sentences built up in this way from atomic formulas of type $w = u \cdot v$, $w = u + v$. Since the predicates $w = u + v$ and $w = u \cdot v$ are definable in the metalanguage $\mathbf{I}_+$, it is clear that the latter is extremely rich, but, in contradistinction to $\mathbf{I}$, we have the following

THEOREM 3.11. *There is no algorithm which, given any sentence $\mathfrak{C}$ in $\mathbf{I}_+$, determines whether it is true or not.*

COROLLARY. *There is no algorithm which, given any formula $\mathfrak{A}(x^{a,b})$ in $\mathbf{I}_+$, determines whether the ω-language $\hat{x}^{a,b}\mathfrak{B}(x^{a,b})$ is representable in a finite automaton or not.*

In order to verify this, fix two formulas $\mathfrak{N}_1(x^{a,b})$ and $\mathfrak{N}_2(x^{a,b})$ such that the first defines a finite-state ω-language, while the second defines an ω-language which is not representable in a finite automaton (say the language L considered at the beginning of this section). Now, if $\mathfrak{N}$ is an arbitrary formula $\mathbf{I}_+$, consider the formula

$$(\mathfrak{N} \,\&\, \mathfrak{N}_1(x^{a,b})) \vee (\neg\mathfrak{N} \,\&\, \mathfrak{N}_2(x^{a,b})), \qquad (*)$$

which is equivalent to $\mathfrak{N}_1(x^{a,b})$ if $\mathfrak{N}$ is true and to $\mathfrak{N}_2(x^{a,b})$ if $\mathfrak{N}$ is false. Thus the decision problem for the formula $\mathfrak{N}$ is effectively reducible to the problem: is the ω-language defined by (*) representable in a finite automaton?

REMARK I. We could have restricted the class of formulas of $\mathbf{I}_+$ to a subclass $\tilde{\mathbf{I}}_+$ containing only formulas which are known to define finite-state ω-languages. Problem A′ of the general synthesis problem has a trivial solution for this subclass $\tilde{\mathbf{I}}_+$. Nevertheless, this subclass can be so constructed that there is no algorithm for Problem C′ (i.e., an algorithm constructing a finite automaton). This follows immediately from a slight modification of formula (*): we let $\mathfrak{A}_1(x^{a,b})$ and $\mathfrak{A}_2(x^{a,b})$ be formulas which define two *different* finite-state ω-languages.

To conclude this section, we consider the situation that arises when one tries to use the set of recursion schemata as a metalanguage. It is well known that any effective function or operator may be defined (in more than one way) by a suitable recursion schema; conversely, any recursion schema defines an effective operator or function.

For example, consider the schema

$$\begin{cases} q_0(1) = 0, \quad q_1(0) = 0, \\ q_0(t+1) = q_0(t) + \overline{\operatorname{sgn}}\, x(t), \\ q_1(t+1) = q_1(t) + x(t), \\ z(t) = \operatorname{sgn}[q_1(t) \dot{-} q_0(t)], \end{cases}$$

where

$$a \dot{-} b = \begin{cases} a - b, & \text{if} \quad a \geqq b, \\ 0\ , & \text{if} \quad a < b; \end{cases}$$

$$\operatorname{sgn} a = \begin{cases} 0, & \text{if} \quad a = 0, \\ 1, & \text{if} \quad a > 0; \end{cases}$$

$$\overline{\operatorname{sgn}}\, a = 1 \dot{-} a.$$

If the input function x assumes the values $\{0,1\}$ only, then, as is easily seen, this is also true of the output function z. In other words, this schema defines an operator transforming ω-words over the alphabet $\{0,1\}$ into ω-words over the same alphabet. It is easy to see that the operator in question is precisely T_4 of Section II.1, which identifies the property: "the word $x(1)\ldots x(t)$ contains more occurrences of the letter 1 than of the letter 0." Now T_4 is indeed a nonanticipatory operator, but it has infinite weight and so is not a finite-state operator. Note, moreover (this also applies to the previous examples), that by adding recursions defining the functions $a + b$, $a \dot{-} b$, $\operatorname{sgn} a$, $\overline{\operatorname{sgn}}\, a$ one converts the above schema into a primitive-recursive schema defining the operator T_4 in the sense just explained.

It is also clear that, if one identifies finite alphabets with initial segments of the natural number series, any finite-state operator is defined by a primitive-recursive schema, in the sense of the very definition of the concept "finite automaton"; this schema is obtained by adding recursions defining the functions Φ and Ψ to the relations

$$\begin{cases} q(1) = q_0, \\ q(t+1) = \Psi[q(t), x(t)], \\ z(t) = \Phi[q(t), x(t)]. \end{cases}$$

Given a primitive-recursive schema **K** with input function symbol x and output function symbol z (to fix ideas, suppose that x represents a function assuming two possible values $\{0,1\}$). The family of all such schemata may be regarded as a metalanguage, for which the usual problems of the general synthesis problem arise naturally:

A′. *Given* **K**, *determine whether it defines a finite-state operator or not.*

C′. *Construct a finite automaton realizing the finite-state operator defined by* **K**.

However (and we shall presently prove this), for neither A′ nor C′ is there a decision algorithm, since certain algorithmically unsolvable problems are reducible to them.

1. Any primitive-recursive schema **K** in which x does not actually appear (or has only a fictitious occurrence there) defines a primitive-recursive function z. In other words, it defines a constant operator T (i.e., an operator independent of x). Now T is finite-state if and only if z is a periodic function (Section II.2). However, it is known [9] that there is no algorithm which, given an arbitrary primitive-recursive schema, determines whether it defines a periodic function or not.

2. Let **K** be a primitive-recursive schema defining a function z (as before, x has only a fictitious occurrence in **K**). Let q and h be function symbols which do not appear in **K**; add the following recursions to **K**:

$$\begin{cases} q(0) = z(0), \\ q(t+1) = q(t) \cdot z(t+1), \\ h(t) = \operatorname{sgn} q(t). \end{cases}$$

The extended schema $\tilde{\mathbf{K}}$ defines a function h which is identically 1 if z never assumes the value 0. But if z first assumes the value 0 for an argument t_0, then

$$h(t) = \begin{cases} 0 & \text{for} \quad t \geqq t_0 \\ 1 & \text{for} \quad t < t_0. \end{cases}$$

Thus, for any schema **K**, the new schema $\tilde{\mathbf{K}}$ defines a periodic function h; in this sense, **K** defines a finite-state constant operator. Now, were effective construction of an appropriate finite automaton possible, this would enable us to determine effectively, given the schema **K**, whether the function z that it defines vanishes at least once. However, it is well known that there is no algorithm, which, given an arbitrary primitive-recursive schema, determines whether the function that it defines vanishes or not.

REMARK II. Though there is no single effective method for constructing finite automata corresponding to schemata **K** of the above type, any initial segment of the function defined by a schema $\mathfrak{N}$ (of the ω-word generated by an automaton) *is* effectively constructible. This is also the situation in any specific case when it is known that a given primitive-recursive schema defines a finite-state operator T; any finite tree of this operator is effectively constructible.

Supplementary material, problems

I. With any regular expression $\mathfrak{A}$ one can associate a nonnegative number $*(\mathfrak{A})$, called the *star-height* of $\mathfrak{A}$, defined inductively:

1) if $\mathfrak{A}$ contains no occurrence of the symbol * (the iteration symbol), then $*(\mathfrak{A}) = 0$;

2) $*(\mathfrak{A}_1 \vee \mathfrak{A}_2) = *(\mathfrak{A}_1 \cdot \mathfrak{A}_2) = \max\{*(\mathfrak{A}_1), *(\mathfrak{A}_2)\}$;

3) $*(\mathfrak{A}^*) = *(\mathfrak{A}) + 1$.

In other words, the star-height of an expression is the maximal length of a sequence of stars in the expression such that each star in the sequence is in the "scope" of one of the following stars.

For example, the star-heights of the expressions $((0^*1 \vee 110)^*10^*)^*$ and $(0^*1 \vee 110)^*(0^*111)^*$ are 3 and 2, respectively.

Define the *star-height of a finite-state language* $\mathfrak{A}$ as the minimal star-height of any regular expression defining the language.

Show that for any n the source with 2^n vertices illustrated in Figure 31 defines a language whose star-height is exactly n (Dejean and Schützenberger [88]). The vertex 1 is the sole initial vertex and the sole final vertex.

Is there an algorithm which, given a finite automaton, determines the star-height of the language that it represents? (McNaughton's problem [106]).

REMARK. A language has star-height 0 if and only if it is finite.

II. Define the *weight of a regular expression* to be the number of states

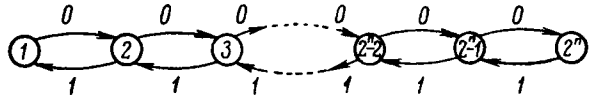


Figure 31

in the minimal automaton representing the language that it defines. The method presented in Chapter III for the synthesis of regular expressions yields the following bound for the weight of an expression: if the length* of the expression is n, its weight is at most 2^{3n-2}. Actually, one can prove a better estimate (Grinberg [26]): if an expression has length n, its weight is at most $2^{\mu(n)}$, where $\mu(n) \sim n/2$ $(n \to \infty)$. This estimate cannot be essentially improved, since for any $p = 1,2,3,\ldots$ one can construct a regular expression over the alphabet $\{0,1\}$ of length $2p$ and weight 2^{p-1}.

III. Length and weight can be defined analogously for ω-regular expressions. Find a bound for the weight of an expression in terms of its length.

IV. The concepts of star-height, weight and length may be extended naturally to expressions which involve the intersection ($\cap$) and complementation ($\neg$) symbols besides the symbols $\vee, \cdot, *$. Problems I, II, III may be rephrased accordingly.**

V. Throughout this problem we allow languages to contain the empty word.

Let S_{ij} $(j = 1,\ldots,n,\ i = 1,2,\ldots,n)$ and R_i $(i = 1,2,\ldots,n)$ be languages over the same alphabet X. Consider the following system of equations in n unknown languages $A_1, A_2, \ldots, A_n$:

$$\begin{aligned} A_1 &= S_{11} \cdot A_1 \vee S_{12} \cdot A_2 \vee \ldots \vee S_{1n} \cdot A_n \vee R_1, \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ A_n &= S_{n1} \cdot A_1 \vee S_{n2} \cdot A_2 \vee \ldots \vee S_{nn} \cdot A_n \vee R_n, \end{aligned} \qquad (*)$$

where $\cdot$ and $\vee$ denote, as usual, concatenation product and union of languages. The problem of existence and uniqueness of a solution of the system (*) may be solved by considering an ordered graph with n vertices q_1, $q_2, \ldots, q_n$ (the so-called characteristic graph of the system), in which an edge goes from q_i to q_j if and only if S_{ji} contains the empty word. The following assertions hold (Bondarchuk [21]):

* *The length of a regular expression* is the number of occurrences of letters of the alphabet (not counting occurrences of operation symbols).

** *Translator's note*: To the best of my knowledge, Problem I has been solved; i.e., with the additional operations, any expression has an equivalent expression with star-height $\leqq 1$.

If all S_{ji} and R_i in (*) are finite-state languages and the characteristic graph of the system contains no loops, then the system (*) has a unique solution consisting of n finite-state languages.

There is an algorithm which, given finite automata representing the initial languages, constructs finite automata representing the solution-languages

VI. A finite-state ω-language is closed* if and only if it is definable by an **I**-formula whose prefix consists of quantifiers over predicate variables followed by $\forall t$.

VII. Church's decision problem [84]: Given an **I**-formula $\mathfrak{A}(x, y)$ and a finite automaton $\langle \mathfrak{M}, q_0 \rangle$, to determine whether the operator $T(\mathfrak{M}, q_0)$ satisfies the formula $\mathfrak{A}(x, y)$.

Show that there exists an algorithm which, given any **I**-formula and any finite automaton, solves Church's decision problem. Which of the problems of Section I.4 are decision problems in this sense?

VIII. Define a language **I'**, which differs from **I** only in that the predicate variables are interpreted not as arbitrary ω-words but as special ω-words.** As usual, $\mathfrak{A}(x)$, $\mathfrak{A}(x, y)$, ... will denote formulas (**I'**-formulas) with the free variables indicated in the parentheses; $\hat{x}\,\mathfrak{A}(x)$, $\widehat{xy}\,\mathfrak{A}(x, y)$, ... will denote the ω-languages that they define (which obviously consist of special ω-words). The following theorem is true (Büchi [73]):

1. There is an algorithm (synthesis algorithm) which, given any formula $\mathfrak{A}(x_1, \ldots, x_m)$, constructs a special finite automaton whose behavior is the ω-language $\widehat{x_1 \ldots x_m}\,\mathfrak{A}(x_1, \ldots, x_m)$.

2. There is an algorithm (analysis algorithm) which, given any special finite automaton with input alphabet $\{0, 1\}$, constructs a formula $\mathfrak{A}(x^{0,1,\ldots}, \ldots)$ such that $\hat{x}\mathfrak{A}(x, \ldots)$ is the behavior of the automaton.

3. There is an algorithm which, for any sentence of **I'**, decides whether it is true or false.

Prove these assertions as corollaries of the corresponding assertions for **I**.

IX. Any **I**-formula $\mathfrak{A}(t)$ with one free individual variable defines an ultimately periodic predicate of a natural argument; i.e., there exist constants l (the phase) and p (the period) such that the **I**-formula $\forall t[\mathfrak{A}(l + t + p) \leftrightarrow \leftrightarrow \mathfrak{A}(l + t)]$ is true. Conversely, any ultimately periodic predicate $A(t)$ is definable in **I** by a suitable formula $\mathfrak{A}(t)$.

* In the sense of Problem I in Chapter II.

** See Problem X in Chapter II.

Translator's note: This is Büchi's "Sequential Calculus" or "Weak Second-order Arithmetic" [73].

Find a characterization of the class of binary predicates of natural arguments which are definable in **I**.

X. Show that a monadic (binary) predicate is definable in **I** if and only if it is definable in **I'**.

XI. Is the statement of Problem X true for m-ary predicates?

XII. Consider the following variation $\mathbf{I}_{pr}$ of **I**: predicate variables are interpreted not as arbitrary ω-words but as ultimately periodic ω-words. The following proposition is true (Büchi [74]): A sentence is true in **I** if and only if it is true in $\mathbf{I}_{pr}$.

The analogue of this assertion, in which $\mathbf{I}_{pr}$ is replaced by **I'**, is false; find a suitable counterexample.

XIII. Consider the extension of the language **I** which includes, besides the successor function ϕ ($\phi(t) = t + 1$), the function ψ such that $\psi(t) = 2t$. It is obvious that the extension obtained by admitting the sum function (see Section III.11) is stronger than this extension, since $2x$ is expressible as $x + x$. Nevertheless, even this seemingly slight extension has no decision algorithm for sentences (R. M. Robinson [118]).

XIV. Let $R_i(x_1, \ldots, x_n)$, $i = 1,2$, be regular expressions over the alphabet $\{x_1, \ldots, x_n\} \cup \{\wedge\}$ ($\wedge$ is the empty word).

Consider the following properties of the pair of regular expressions $R_1(x_1, \ldots, x_n)$, $R_2(x_1, \ldots, x_n)$:

1) $R_1(x_1, \ldots, x_n)$, $R_2(x_1, \ldots, x_n)$ define the same language over the alphabet $\{x_1, \ldots, x_n\}$.

2) For any languages $S_1, S_2, \ldots, S_n$ (over an arbitrary alphabet), $R_1(S_1, \ldots, S_n) = R_2(S_1, \ldots, S_n)$.

3) For any finite-state languages $S_1, \ldots, S_n$ (over an arbitrary alphabet), $R_1(S_1, \ldots, S_n) = R_2(S_1, \ldots, S_n)$.

4) For any languages $S_1, \ldots, S_n$ over a k-letter alphabet ($k = 1,2,3, \ldots$), $R_1(S_1, \ldots, S_n) = R_2(S_1, \ldots, S_n)$.

5) For any finite-state languages $S_1, \ldots, S_n$ over a k-letter alphabet ($k = 1,2,3, \ldots$), $R_1(S_1, \ldots, S_n) = R_2(S_1, \ldots, S_n)$.

Show that

a) for $k \geqq 2$, the assertions 1) through 5) are equivalent, i.e., the i-th assertion implies the j-th ($i,j = 1,2,3,4,5$);

b) for $k = 1$, assertions 4) and 5) are equivalent, but they do not imply 1), 2), 3).

XV. A pair of regular expressions $R_1(x_1, \ldots, x_n)$, $R_2(x_1, \ldots, x_n)$ is said to be an *identity in the algebra of languages* over a k-letter alphabet (notation: $R_1(x_1, \ldots, x_n) \equiv R_2(x_1, \ldots, x_n)$) if it satisfies assertion 4).

A system Σ of identities in the algebra of languages over a k-letter alphabet is said to be *complete* if any identity in the algebra of languages over a k-letter alphabet is deducible from Σ (as an axiom system) by means of the following rules of inference:

α) Substitution rule: for any regular expression P, the identity $R_1(x_1, \ldots, P, \ldots, x_n) \equiv R_2(x_1, \ldots, P, \ldots, x_n)$ is deducible from $R_1(x_1, \ldots, x_i, \ldots, x_n) \equiv R_2(x_1, \ldots, x_i, \ldots, x_n)$.

β) Replacement rule: from the identities $R_1(S_1) \equiv R_2$ and $S_1 \equiv S_2$, the identity $R_1(S_2) \equiv R_2$ is deducible.

Show that, when iteration is defined by $S^* \underset{df}{=} \wedge \vee S \vee SS \vee \ldots$,

a) the algebra of languages over a k-letter alphabet has no finite complete system of identities ($k = 1,2,\ldots$) (Red'ko [51]);

b) for any k, there is a complete system of identities in the algebra of languages over a k-letter alphabet which contain the empty word (Yanov [67]).

Similarly, one defines an identity in the algebra of languages not containing the empty word as a pair of regular expressions over the alphabet $\{x_1, \ldots, x_n\}$ with the usual definition of iteration: $S^* \underset{df}{=} S \vee SS \vee SSS \vee \ldots$.

Show that there is no finite complete system of identities in the algebra of languages over a k-letter alphabet which do not contain the empty word ($k = 1,2,\ldots$) [51].

Notes

The metalanguage of regular expressions was first described by Kleene [100], who also established appropriate analysis and synthesis algorithms. However, since Kleene considered McCulloch nerve nets (or other, similar objects) rather than finite automata, his algorithm constructs a nerve net, but not transition and output matrices. In other words, Kleene's work does not yet clearly differentiate between the stages of functional and structural synthesis. Analogous remarks apply to the analysis of nerve nets.

Subsequently, many authors, working independently, dealt with improvement of analysis and synthesis algorithms for regular expressions (Glushkov [6], Copi, Elgot and Wright [87], McNaughton, Yamada and others). In the process, certain cumbersome details, inherent in Kleene's formulation of the metalanguage of regular expressions because of his utilization of nerve nets, were eliminated.

The synthesis procedure employed in this book is based on operations over sources; it is a slight modification of the Rabin–Scott procedure [114].

ω-regular expressions were defined by McNaughton in [102];* the synthesis algorithm makes essential use of his theorems on concatenation and strong iteration for ω-languages.

Some hints as to the advisability and possibility of using the predicate calculus as a metalanguage in synthesis theory may already be found in the above-mentioned paper of Kleene [100]. Independently, Trakhtenbrot [57] and Church [84] developed suitable metalanguages based on the calculus of monadic predicates of a natural argument, and constructed appropriate synthesis algorithms. These metalanguages may be regarded as fragments of the metalanguage **I**, obtained by imposing special restrictions on the individual quantifiers [57] or by excluding predicate quantifiers [84]. Subsequently, Büchi [73] investigated the language **I**′ with the same purpose in mind, and constructed a synthesis algorithm for special automata (see Problem VIII). The languages **I** and **I**′ have also been investigated, independently of automata theory, on the initiative of Tarski, who has posed the following problems:

A) Is there an algorithm deciding the truth of formulas in **I** (or **I**′)? In other words, is the language **I** (or **I**′) decidable?

B) Is the predicate $\tau + t = \sigma$ definable in **I** (**I**′)?

Büchi was the first to direct attention to the possibility of using automata to solve problems of logic. In [73] he gave a positive solution to problem A), and so a negative solution to B), for the language **I**′. A negative answer to problem B) for the language **I** was given by Trakhtenbrot in [58]. However, Büchi [74] substantially strengthened the latter result, proving that **I** is decidable; he also described the ω-languages definable in **I**. Büchi's ingenious proof utilized Ramsey's Theorem (see Problem XIV, Chapter I). These achievements notwithstanding, the synthesis of an automaton whose operator satisfies an **I**-formula $\mathfrak{A}(x, y)$ remained open, not counting a few special cases solved by Church and Trakhtenbrot. The latter authors, imposing various restrictions on the prefixes of the formulas, proved the existence of a general solution (see p. 145) and gave an algorithm for its construction. In particular, Trakhtenbrot proved in [59] that any ω-word operator definable in **I** is an operator with finite memory.

A definitive solution to this problem became possible only after Büchi and Landweber had proved the Uniformization Theorem (Chapter II). It is interesting to note that the Büchi–Landweber theorem on ω-languages

* *Translator's note*: McNaughton uses the unmodified term "regular expressions."

yields a proof of the decidability of **I** without using Ramsey's Theorem (see Theorem 3.10).

Later [75], Büchi generalized the concept of the behavior of an automaton, considering transfinite input and output sequences. This enabled him to prove the decidability of languages $\mathbf{I}_\xi$ for certain transfinite ξ, where $\mathbf{I}_\xi$ is like **I** except that the predicate variables are interpreted as transfinite sequences of ordinal ξ.

Theorem 3.7 is due to Bar-Hillel, Perles and Shamir [70] (see also Chomsky [83]).

Dejean and Schützenberger [88], McNaughton [106] and Eggan [89] have studied the star-height of regular expressions and finite-state languages, proving the existence of finite-state languages with arbitrarily large star-height.

CHAPTER IV

AUTOMATON IDENTIFICATION

IV.1. Introduction

In the preceding chapter we studied the synthesis problem for some specific metalanguages used to phrase the requirements from the functioning of the projected automaton. In this chapter we shall be concerned with the synthesis problem in the following situation: no general description (in a metalanguage) of the behavior of the automaton is available; nevertheless, for any input word the output of the projected automaton can be determined. Here the situation may be described as follows. We are given an initialized automaton $\mathfrak{M}$—called a "*black box*"—about whose internal structure (diagram) nothing (or almost nothing) is known. Input words can be applied to the input of the automaton and the corresponding output words observed. The problem is to identify* the automaton, i.e., to construct the diagram of an automaton which functions in the same way as $\mathfrak{M}$.

We consider two typical examples involving the identification of black boxes.

EXAMPLE 1. Suppose that the client has planned a certain operator T, and the designer's task is to construct an automaton realizing T. Suppose, moreover, that the client, though not in a position to describe his operator in a language accessible to the designer, is nevertheless capable of answering any question of the type "Into what does the operator T transform the word $x(1)\ldots x(t)$?". The designer then has, as it were, an imaginary black box which he must identify. In so doing, he can determine for any input word x the output generated by the black box, provided the latter is in its initial state before application of the input x.

EXAMPLE 2. Let $\mathfrak{M}$ be an actual unknown automaton whose only accessible elements are the input and output terminals. For example, this might be a secret lock with unknown combination. It is required to identify

* *Translator's note*: The Russian word used here means "decode," "decipher."

the automaton. We may apply different input words to the automaton $\mathfrak{M}$ and observe the resulting output words. However, in contradistinction to the preceding example, we do not assume that the automaton can be returned to its initial state after each application of an input word: when the next input word is applied the automaton is in the state to which it was brought by the previous input words.

The precise formulation of the identification problem is closely bound up with the rigorous definition of an algorithm over black boxes (identification algorithm).* This will be given presently, and at the same time we shall present a classification of these algorithms.

We begin with an informal discussion. An identification algorithm should comprise effective instructions as to what questions of the type "What is the output of the black box for input x?" should be asked, and how the answers to these questions should be used to construct an automaton which, presumably, reproduces the functioning of the black box.

Identification algorithms may be classified according to the following three criteria.

I. *Resettability.* There are two possibilities:

1) Each question is addressed to the black box in the initial state (it is assumed that the black box is equipped with a "reset button" by means of which the initial state is restored before each question; this corresponds to Example 1). In this case we shall call the algorithm multiple.

2) Each question is addressed to the black box in the state to which previous operations have brought it (the black box has no "reset button"; this corresponds to Example 2). In this case we shall call the algorithm simple.

II. *Dependence on the previous history of the process.* There are two possibilities:

1) Each question depends on the previous questions and answers. We shall then call the algorithm conditional.

2) Each question is independent of previous questions and answers. In other words, all the questions addressed to the black box are predetermined. In this case we shall call the algorithm unconditional.

III. *Dependence on* a priori *information.* We shall assume that the input and output alphabets of the black boxes are fixed and may be used for identification. Apart from this, there are two possibilities:

* *Translator's note*: The equivalent concept in Western literature is "experiment," a term which Trakhtenbrot and Barzdin' use in a different, though related sense (see Section II.4).

1) The algorithm utilizes an upper bound on the number of states of the black box. In this case we assume that for every black box $\mathfrak{M}$ there is a preassigned upper bound on the number of states (which we shall denote by $K_{\mathfrak{M}}$). We shall call such black boxes *relative black boxes*, and the algorithm itself will be termed an algorithm over relative black boxes.

2) The algorithm utilizes neither an upper bound on the number of states of the black box nor other information (except for the information concerning the input and output alphabets) not obtainable by applying input words and observing the corresponding response. Thus no *a priori* information about the black boxes is required. We shall call the latter *absolute black boxes* and the algorithm will be termed an algorithm over absolute black boxes.*

In all, these criteria give rise to eight natural types of algorithms (see figure on the following page).

These informal explanations should more or less clarify our terminology for algorithms over black boxes. Nevertheless, to avoid possible misunderstandings we shall present more rigorous definitions of these algorithms.

First, abbreviate the phrase "apply a word x to the input of an automaton $\mathfrak{M}$, in its initial state q_0 (in state q_i), and observe the corresponding output word" by the phrase "*test an automaton* $\mathfrak{M}$ (*an automaton* $\mathfrak{M}$ *in state* q_i) *with the word* x." By the *result of a test* on an automaton $\mathfrak{M}$ with input words $x, x', x'', \ldots$ we shall mean the tree of a partial operator defined only on the words $x, x', x'', \ldots$ and their initial segments, realized by the automaton $\langle \mathfrak{M}, q_0 \rangle$. For example, for the automaton of Figure 32*a* the result of a test with the input words

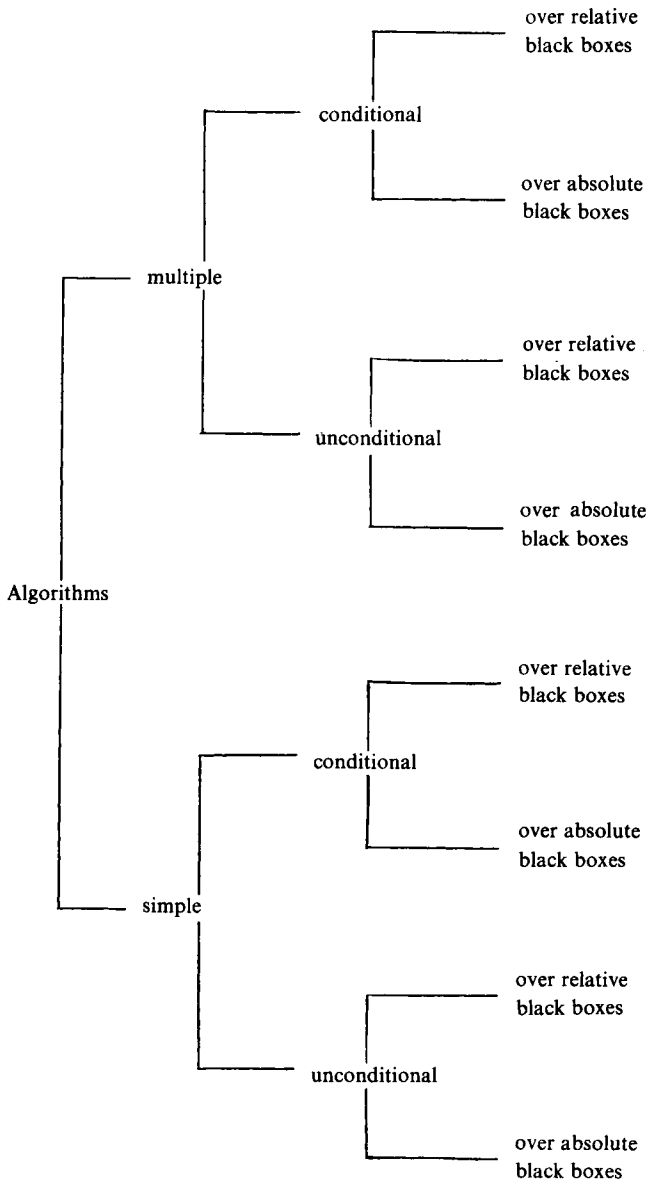
$$x = x_1 x_1 x_2 \quad \text{and} \quad x' = x_1 x_2 x_1 x_2$$

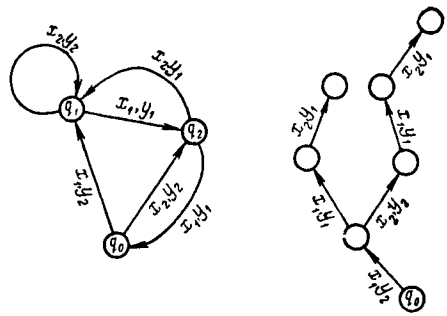
is the tree of Figure 32*b*. The tree of a partial operator realized by an automaton $\langle \mathfrak{M}, q_0 \rangle$ is said to be *compatible* with the automaton $\langle \mathfrak{M}, q_0 \rangle$. Therefore, the result of a test on an automaton $\langle \mathfrak{M}, q_0 \rangle$ with (all) input words of length l will sometimes be called a total tree of height l compatible with $\langle \mathfrak{M}, q_0 \rangle$.

Unless otherwise stated, q_0 will always denote the initial state of an automaton or black box.

1. A *multiple unconditional algorithm over relative black boxes* is a set of instructions which, for every natural K, specifies:

* Apart from these two cases, describing the "minimality" of the *a priori* information, a fairly natural and widely studied case is that in which the diagram of each black box is known but not its initial state (see, e.g., [108, 4, 98] and Problem IV at the end of this chapter).


Algorithms
multiple
simple
conditional
unconditional
conditional
unconditional
over relative black boxes
over absolute black boxes
over relative black boxes
over absolute black boxes
over relative black boxes
over absolute black boxes
over relative black boxes
over absolute black boxes



a *b*

Figure 32

a) a finite set of input words with which to test black boxes whose number of states is bounded by K;

b) a procedure for construction of a suitable automaton, given the test result and the upper bound K (both here and below construction of an automaton will mean construction of its diagram).

2. The characteristic feature of a *multiple unconditional algorithm over absolute black boxes* is that all black boxes are tested with the same finite set of input words. Algorithms of this type for black box identification are considerably restricted in ability, and we shall not consider them in the sequel.

3. A *multiple conditional algorithm over relative black boxes* operates in steps. At each step, the black box $\mathfrak{M}$ is tested with a certain finite set of input words. According to the result of this test, the results of tests performed at previous steps, and the upper bound K on the number of states of $\mathfrak{M}$, the algorithm produces either

a) an automaton; or

b) a new set of input words (with which to test the black box at the next step).

An algorithm of this type may be described, for example, by a recursive function $\Phi(w,k)$, where w is either the empty word $\wedge$ or a finite tree (test result), k is a natural number (upper bound on the number of states), and the value of the function $\Phi(w,k)$ is either an automaton (diagram) or a set of input words. Applied to a black box $\mathfrak{M}$ with upper bound K on the number of states, the algorithm functions as follows:

Step 0 (this step is degenerate and independent of $\mathfrak{M}$). Compute $\Phi(\wedge, K)$. There are two cases:

a) If $\Phi(\wedge, K)$ is an automaton, the algorithm stops and the result is the the automaton $\Phi(\wedge, K)$.

b) If $\Phi(\wedge, K)$ is a set v_0 of input words, proceed to the next step.

Step i ($i = 1,2,\ldots$). Test the black box $\mathfrak{M}$ with the input words of the set v_{i-1}, and construct a tree w_{i-1}, which is the result of this and the previous tests (so that w_{i-1} is the result of testing with the words of the set $v_0 \cup v_1 \cup \ldots \cup v_{i-1}$). Then compute $\Phi(w_{i-1}, K)$. There are two cases:

a) If $\Phi(w_{i-1}, K)$ is an automaton, the algorithm stops and the result is the automaton $\Phi(w_{i-1}, K)$.

b) If $\Phi(w_{i-1}, K)$ is a set v_i of input words, proceed to the next step.

4. The only difference between a *multiple conditional algorithm over absolute black boxes* and the preceding variant is that it makes no use of the upper bound K for the number of states. In other words, the function Φ which describes the algorithm depends on a single argument w.

5. A *simple unconditional algorithm over relative black boxes* differs from its multiple counterpart only in that, for each K, it specifies a single test word, rather than a finite set of words, for black boxes with upper bound K on the number of states.

6. Like its multiple counterpart, we shall not consider the *simple unconditional algorithm over absolute black boxes.*

7. A *simple conditional algorithm over relative black boxes*, like its multiple analogue, functions in steps. At each step, the black box $\mathfrak{M}$ is in the state to which the previous steps have brought it; it is tested with a single input word. According to the result of this test, that of tests at previous steps, and the upper bound K on the number of states, the algorithm produces either

a) an automaton; or

b) a new input word (which is used to test the black box at the next step).

As before, the algorithm can be described by a recursive function $F(w,k)$, where w is either the empty word or a finite tree consisting of a single branch (the result of testing with a single input word), k is a natural number (upper bound on the number of states), and the value of the function $F(w,k)$ is either an automaton or an input word. The application of this algorithm to a black box $\mathfrak{M}$ differs from that of its multiple counterpart only in that the result of the i-th step (if the algorithm has not stopped) is not a set of input words but a single input word $v_i = F(w_{i-1}, K)$, and this word is applied at the $(i+1)$-th step to the black box $\mathfrak{M}$ in the state to which the i-th step has brought $\mathfrak{M}$. In this case, w_i will denote a tree consisting of a single branch: the result of the last and the preceding tests (i.e., the result of testing with the word $v_0 v_1 \ldots v_i$).

8. The only difference between a *simple conditional algorithm over absolute black boxes* and the preceding variant is that it makes no use of an upper bound K for the number of states. In other words, the function F depends on a single argument w.

This completes our classification of algorithms.*

Denote the outcome of an algorithm Ω applied to a black box $\mathfrak{M}$ by $\Omega(\mathfrak{M})$.

Note that a simple algorithm Ω tests a black box $\mathfrak{M}$ with a sequence of input words $x, x', x'', \ldots$ and, in the final analysis, observes the response of the black box $\mathfrak{M}$ to a single input word—the concatenation $xx'x''\ldots$ We denote this concatenation by $x(\Omega, \mathfrak{M})$, and the state into which $\mathfrak{M}$ goes after application of the simple algorithm Ω by $q_0x(\Omega, \mathfrak{M})$. The residual operator with respect to the word $x(\Omega, \mathfrak{M})$, i.e., the operator $T(\mathfrak{M}, q_0x(\Omega, \mathfrak{M}))$, will be called the *operator realized by the black box* $\mathfrak{M}$ *after application of the simple algorithm* Ω.

We shall now rigorously define one of the central concepts of this chapter—identification. There will be two definitions: initial identification and residual identification. An algorithm Ω is said to *initially identify* a black box $\mathfrak{M}$ if the automaton $\Omega(\mathfrak{M})$ realizes the same operator as the black box $\mathfrak{M}$, i.e., $T(\Omega(\mathfrak{M}), q_0) = T(\mathfrak{M}, q_0)$. Henceforth all our discussions of identification by multiple algorithms will refer to initial identification.

For simple algorithms, we introduce a weaker concept.** A simple algorithm Ω is said to *residually identify* a black box $\mathfrak{M}$ if the automaton $\Omega(\mathfrak{M})$ realizes the same operator as the black box $\mathfrak{M}$ after application of the algorithm Ω, i.e., $T(\Omega(\mathfrak{M}), q_0) = T(\mathfrak{M}, q_0x(\Omega, \mathfrak{M}))$. Henceforth, identification by simple algorithms will always refer to residual identification.

* Note that the multiple conditional algorithm over absolute black boxes coincides, in effect, with the concept of computable functional (see, e.g., [39]), with the (inessential!) restriction that the argument ranges over word operators and the values of the functional over descriptions of finite automata. The simple algorithm may be regarded as a special case of a functional of this type. It is also worth mentioning that a frequently employed alternative term in the literature for the process described by a simple (conditional, unconditional) algorithm is a simple (preset or inhomogeneous, adaptive or homogeneous) experiment on black boxes, and similarly for the multiple variant. The word "experiment" was used in a somewhat different sense in Chapter II of this book.

** The point is that the requirement of initial identification by simple algorithms is so strong that one cannot expect positive results in interesting situations. On the other hand, if this requirement is replaced by the weaker modification which we have called residual identification and only this variant studied, the situation changes significantly. We shall see later that the theorems valid for residual identification by simple algorithms are approximately the same as those valid for initial identification by multiple algorithms.

Residual identification is indeed weaker than initial identification, but it nevertheless provides full information on the subsequent behavior of the black box (after the identification algorithm has been applied). In many cases it provides full information, up to state equivalence, on the original black box. This is the case for strongly connected automata. Recall that an automaton is *strongly connected* if, for any ordered pair of states (q_i, q_j), there is an input word which takes the automaton from state q_i to state q_j. It is easily seen that if the black box $\mathfrak{M}$ is strongly connected, initial identification is precisely the same as construction of an automaton equivalent to the black box (equivalence without regard for initial states).

In the sequel we shall study not only the theoretical possibility of identifying black boxes, but also the complexity of identification. The parameter characterizing complexity of identification is the signaling function.† The *signaling function* $\Omega^*(\mathfrak{M})$ of an algorithm Ω over black boxes is defined as the length of the longest input word with which the algorithm Ω tests $\mathfrak{M}$ from the initial state q_0. Thus, for a multiple algorithm $\Omega^*(\mathfrak{M})$ is the maximal length of the input words with which Ω tests $\mathfrak{M}$; for a simple algorithm it is the length of the word $x(\Omega, \mathfrak{M})$. It is obvious that if a black box $\mathfrak{M}'$ is indistinguishable from a black box $\mathfrak{M}$ by words of length $\Omega^*(\mathfrak{M})$ then $\Omega(\mathfrak{M}') = \Omega(\mathfrak{M})$.

Above we introduced eight types of algorithms over black boxes. Each of these types obviously has a specific identification problem: can the corresponding type of black box be identified by algorithms of this type, and, if so, what are their signaling functions? In cases where not all black boxes are identifiable, one may ask what proportion of black boxes are nevertheless identifiable. The present chapter deals with these questions. We conclude this section with a few conventions.

First, as already indicated, we shall consider automata (black boxes) with the same input alphabet $X = \{x_1, \ldots, x_m\}$, where $m = \text{const} \geqq 2$, and output alphabet $Y = \{y_1, \ldots, y_n\}$, where $n = \text{const} \geqq 2$.†† Thus, for example, "the class of all automata" will mean "the class of all automata with fixed alphabets X and Y."

Second (a much less essential restriction), we shall regard the states

† The term "signaling function" is often used in the theory of algorithms; it means a function describing the complexity of an algorithmic process (see, e.g., [11]).

†† In the sequel we shall often have to deal with quantities depending only on m and n. Naturally, we shall call such quantities constants, generally denoting them by the letter C (with or without subscripts).

of our automata as indexed, $q_0, q_1, q_2, \ldots$, and, unless otherwise stated, q_0 will always be the initial state. These automata will be denoted simply by Gothic letters $\mathfrak{M}, \mathfrak{A}, \ldots$ (instead of the previously used notation $\langle \mathfrak{M}, q_0 \rangle$, $\langle \mathfrak{A}, q_0 \rangle, \ldots$). Notation of the type $\langle \mathfrak{M}, q_i \rangle$ will be used when the automaton has an initial state q_i which may differ from q_0.

Similarly, all discussions of automaton graphs will refer to automaton graphs over a fixed alphabet $X = \{x_1, \ldots, x_m\}$ and with indexed states $q_0, q_1, q_2, \ldots$

Third, all our automata (automaton graphs) will have only finitely many states. The number of states of an automaton (automaton graph) $\mathfrak{M}$ will be denoted by $|\mathfrak{M}|$. Similar notation will be used for cardinalities of sets, i.e., if U is a set then $|U|$ is its cardinality.

IV.2. Identification of relative black boxes

In this section we shall prove that it is theoretically possible to identify all relative black boxes, even by means of unconditional algorithms; we shall also determine signaling functions for the algorithms (both multiple and simple) that perform the identification (Theorems 4.1 and 4.2).

We first consider identification by multiple algorithms.

It follows from Section II.13 that any relative black box $\mathfrak{M}$ with upper bound $K_{\mathfrak{M}}$ on the number of states can be initially identified by the following multiple unconditional algorithm Ω_0:

1) Test $\mathfrak{M}$ with input words of length $2K_{\mathfrak{M}} - 1$, and construct a total tree of height $2K_{\mathfrak{M}} - 1$ compatible with $\mathfrak{M}$.

2) Construct the required automaton $\mathfrak{A}$ by applying the contraction procedure of Section II.4 to the above tree (in this case the contraction procedure has a unique outcome).

It is clear that

$$\Omega_0^*(\mathfrak{M}) = 2K_{\mathfrak{M}} - 1,$$

where $\Omega_0^*(\mathfrak{M})$ is the signaling function of the algorithm Ω_0.

We have thus proved

THEOREM 4.1. *There is a multiple unconditional algorithm Ω with signaling function*

$$\Omega^*(\mathfrak{M}) = 2K_{\mathfrak{M}} - 1,$$

which initially identifies any relative black box $\mathfrak{M}$.

It is not hard to verify that the upper bound given by Theorem 4.1 for the signaling function, as a function of $K_{\mathfrak{M}}$, cannot be lowered (even by recourse to conditional algorithms), i.e., there is no (conditional or unconditional) algorithm Ω such that $\Omega^*(\mathfrak{M}) < 2K_{\mathfrak{M}} - 1$, which identifies all relative black boxes $\mathfrak{M}$. To see this, for each K define an automaton $\mathfrak{M}_K$ by Table 11 and an automaton $\mathfrak{M}'_K$ by Table 12 (here we assume that

$$X = \{x_1, x_2\} = \{0,1\} \text{ and } Y = \{y_1, y_2\} = \{0,1\}).$$

The diagrams of $\mathfrak{M}_K$ and $\mathfrak{M}'_K$ are illustrated in Figures 33*a* and 33*b*, respectively, for $K = 3$. It is easily seen that the shortest word by means of which the state q_0 of $\mathfrak{M}_K$ can be distinguished from the state q_0 of $\mathfrak{M}'_K$ is of length $2K - 1$. Set $K_{\mathfrak{M}_K} = K_{\mathfrak{M}'_K} = K$. Then the above observation implies that there is no algorithm Ω with $\Omega^*(\mathfrak{M}) < 2K_{\mathfrak{M}} - 1$ which identifies $\mathfrak{M}_K$ and $\mathfrak{M}'_K$.

We now consider identification of relative black boxes by simple algorithms, which is more difficult than in the multiple case.

In dealing with identification by simple algorithms the degree of reconstructibility $B(\mathfrak{M}, q_0)$ will be replaced by $\max B(\mathfrak{M}, q_i)$, where the maximum is taken over all states q_i of the automaton $\mathfrak{M}$. We denote this maximum by $B^*(\mathfrak{M})$ and call it the *absolute degree of reconstructibility* of $\mathfrak{M}$. By Theorem 2.16′ the degree of reconstructibility $B(\mathfrak{M}, q_0)$ is at most $\delta(\mathfrak{M}, q_0) + \rho(\mathfrak{M}) + 1$, where $\delta(\mathfrak{M}, q_0)$ and $\rho(\mathfrak{M})$ are the degrees of

TABLE 11
$\Psi(x, q), \Phi(x, q)$

x \ q	q_0	q_1	. . .	q_i	. . .	q_{K-2}	q_{K-1}
0	$q_1, 0$	$q_2, 0$	. . .	$q_{i+1}, 0$	. . .	$q_{K-1}, 0$	$q_{K-1}, 0$
1	$q_0, 1$	$q_0, 0$	. . .	$q_{i-1}, 0$	. . .	$q_{K-3}, 0$	$q_{K-2}, 0$

TABLE 12
$\Psi(x, q), \Phi(x, q)$

x \ q	q_0	q_1	. . .	q_i	. . .	q_{K-2}	q_{K-1}
0	$q_1, 0$	$q_2, 0$	. . .	$q_{i+1}, 0$	. . .	$q_{K-1}, 0$	$q_{K-2}, 0$
1	$q_0, 1$	$q_0, 1$	. . .	$q_{i-1}, 0$	. . .	$q_{K-3}, 0$	$q_{K-1}, 0$

accessibility and distinguishability, respectively, of $\mathfrak{M}$. To derive an analogous estimate for the absolute degree of reconstructibility, consider $\max \delta(\mathfrak{M}, q_i)$, where the maximum is taken over all states q_i of $\mathfrak{M}$. Denote this maximum by $\delta^*(\mathfrak{M})$ and call it the *absolute degree of accessibility* of $\mathfrak{M}$. It follows from Theorem 2.16′ that the absolute degree of reconstructibility satisfies the inequalities

$$B^*(\mathfrak{M}) \leqq \delta^*(\mathfrak{M}) + \rho(\mathfrak{M}) + 1,$$

$$B^*(\mathfrak{M}) \leqq 2|\mathfrak{M}| - 1.$$

We shall denote the state to which a word x takes a state q_i by $q_i x$; the *operator realized by* $\mathfrak{M}$ *after application of the word* x is the residual operator $T(\mathfrak{M}, q_0 x)$.

We shall now prove two lemmas.

Let U be a set of automata, and x an input word such that, for any two automata $\mathfrak{M}_1$ and $\mathfrak{M}_2$ in U, either $\mathfrak{M}_1$ and $\mathfrak{M}_2$ generate different output words on application of x, or they realize the same operator after application of x, i.e., $T(\mathfrak{M}_1, q_0 x) = T(\mathfrak{M}_2, q_0 x)$. Then we shall say that the input word x *residually distinguishes* the set U (or automata of the set U).

LEMMA 1. *For any set* U *of automata whose absolute degree of reconstructibility is at most some natural number* s, *there exists an input word of length* $s\,]2m^s \ln |U|[^*$ *which residually distinguishes the set* U.

Proof. Let $\mathfrak{M}_1$ and $\mathfrak{M}_2$ be arbitrary automata whose absolute degree of reconstructibility is at most s. Let us find an upper bound for the number of input words of length ls (where l is any natural number) which do not residually distinguish these two automata. To this end, consider any input

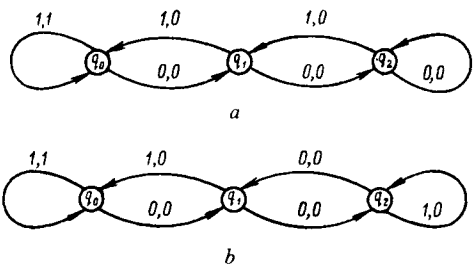


Figure 33

* $]r[$ denotes the smallest natural number $\geqq r$.

word x which does not residually distinguish $\mathfrak{M}_1$ and $\mathfrak{M}_2$; let us find an upper bound for the number of different input words z of length s that can be "tacked onto" x such that xz still does not residually distinguish $\mathfrak{M}_1$ and $\mathfrak{M}_2$. The states $q^{(1)} = q_0x$ and $q^{(2)} = q_0x$ of the automata $\mathfrak{M}_1$ and $\mathfrak{M}_2$, respectively, are clearly distinguishable, since otherwise we would have $T(\mathfrak{M}_1, q_0x) = T(\mathfrak{M}_2, q_0x)$ and the word x would residually distinguish $\mathfrak{M}_1$ and $\mathfrak{M}_2$. Since the absolute degree of reconstructibility of $\mathfrak{M}_1$ and $\mathfrak{M}_2$ is at most s, there must be at least one input word z of length s that distinguishes states $q^{(1)}$ and $q^{(2)}$. Obviously, for this z the automata $\mathfrak{M}_1$ and $\mathfrak{M}_2$ transform the word xz into different output words, and so xz residually distinguishes the automata $\mathfrak{M}_1$ and $\mathfrak{M}_2$. We have thus verified that the number of input words of length s that can be "tacked onto" the word x in such a way that the resulting word does not residually distinguish $\mathfrak{M}_1$ and $\mathfrak{M}_2$ is at most $m^s - 1$, where m^s is the number of input words of length s. It follows by induction on l that the number of different input words of length ls which do not residually distinguish $\mathfrak{M}_1$ and $\mathfrak{M}_2$ is at most $(m^s - 1)^l$.

Now let U be any set of automata with absolute degree of reconstructibility at most s. If some input word does not residually distinguish the set U, it must have this property for at least two automata in the set. The number of possible pairs of automata from the set U is $C^2_{|U|}$. Hence, using the above bound on the number of input words which do not residually distinguish two automata, we see that the number of input words of length ls which do not residually distinguish the set U is at most

$$C^2_{|U|}(m^s - 1)^l = \frac{|U|(|U| - 1)}{2}(m^s - 1)^l.$$

Now, if l_0 is a natural number such that the number of input words of length l_0s not residually distinguishing the set U is less than the total number of words of length l_0s (i.e., less than m^{l_0s}), then there must be an input word of length l_0s which residually distinguishes U. We must therefore find l_0 such that

$$\frac{|U|(|U| - 1)}{2}(m^s - 1)^{l_0} < m^{l_0s}.$$

Solving for l_0, we get

$$l_0 > \frac{\ln|U| + \ln(|U| - 1) - \ln 2}{|\ln(1 - 1/m^s)|}.$$

Since $|\ln(1 - 1/m^s)| > 1/m^s$, it follows that

$$\frac{\ln|U| + \ln(|U| - 1) - 1) - \ln 2}{|\ln(1 - (1/m^s))|} < 2m^s \ln|U|.$$

Set

$$l_0 =]|2m^s \ln|U|[.$$

Then l_0 clearly satisfies the required inequality, and so there exists an input word of length

$$l_0 s = s]2m^s \ln|U|[$$

which residually distinguishes the set U. This proves the lemma.

LEMMA 2. *For any natural numbers k and s, there exists an input word $d(k,s)$ of length $s]2mk(\ln nk)m^s[$ which residually distinguishes automata with at most k states and absolute degree of reconstructibility at most s. Moreover, the word $d(k,s)$ is effectively constructible for any k and s.**

Proof. We may confine the reasoning to automata with exactly k states.

The first assertion of the lemma follows directly from Lemma 1, since the number of different automata with k states is at most $(nk)^{mk}$ (Section 0.3).

We proceed to the second assertion. The required input word $d(k,s)$ of length $s]2mk(\ln nk)m^s[$ can obviously be constructed as follows. Order all input words of length $s]2mk(\ln nk)m^s[$ and, beginning with the first, check whether they residually distinguish automata with k states and absolute degree of reconstructibility at most s. Continue until an input word is encountered which residually distinguishes the automata in question; this word will be $d(k,s)$. The procedure is clearly effective.

Since the absolute degree of reconstructibility of automata with k states is at most $2k - 1$, Lemma 2 directly implies the following

COROLLARY. *For any natural number k, one can effectively construct an input word $d(k)$ of length $]4k^2(\ln nk)m^{2k}[$ which residually distinguishes all automata with at most k states.*

It is now no longer difficult to devise a simple algorithm which residually distinguishes relative black boxes.

Consider the following simple unconditional algorithm Ω_1 over relative black boxes. Let $\mathfrak{M}$ be a black box with at most $K_{\mathfrak{M}}$ states.

* This means that there is an algorithm (e.g., Turing machine) which constructs the word $d(s,k)$ for any natural k and s.

1) Construct an input word $d(K_{\mathfrak{M}})$ of length

$$]4K^2_{\mathfrak{M}}(\ln nK_{\mathfrak{M}})m^{2K_{\mathfrak{M}}}[,$$

which, by the corollary of Lemma 2, residually distinguishes all automata with at most $K_{\mathfrak{M}}$ states. Then test the black box $\mathfrak{M}$ with the input word $d(K_{\mathfrak{M}})$ to find the corresponding output word y_0.

2) Look for an automaton $\mathfrak{A}'$ with minimal number of states (for instance, by checking through all automata with at most $K_{\mathfrak{M}}$ states) which, like the black box $\mathfrak{M}$, responds to input $d(K_{\mathfrak{M}})$ with output y_0. If there are several such automata, choose one of them. The outcome of the algorithm is the automaton $\mathfrak{A}$ obtained from $\mathfrak{A}'$ by defining the initial state to be the state into which the word $d(K_{\mathfrak{M}})$ takes $\mathfrak{A}'$.

It follows from the corollary to Lemma 2 that the automaton $\mathfrak{A}$ realizes the same operator as the black box $\mathfrak{M}$ after application of the word $d(K_{\mathfrak{M}})$, i.e.,

$$T(\mathfrak{A}) = T(\mathfrak{A}', q_0 d(K_{\mathfrak{M}})) = T(\mathfrak{M}, q_0 d(K_{\mathfrak{M}}))$$

(otherwise the input word $d(K_{\mathfrak{M}})$ would not residually distinguish the automata $\mathfrak{M}$ and $\mathfrak{A}'$, which both have at most $K_{\mathfrak{M}}$ states). This proves that the algorithm Ω_1 residually identifies the black box $\mathfrak{M}$, and moreover

$$\Omega_1^*(\mathfrak{M}) =]4K^2_{\mathfrak{M}}(\ln nK_{\mathfrak{M}})m^{2K_{\mathfrak{M}}}[.$$

We have thus proved

THEOREM 4.2. *There is a simple unconditional algorithm Ω with signaling function*

$$\Omega^*(\mathfrak{M}) =]4K^2_{\mathfrak{M}}(\ln nK_{\mathfrak{M}})m^{2K_{\mathfrak{M}}}[$$

which residually identifies any relative black box $\mathfrak{M}$.

It seems probable that the upper bound given by Theorem 4.2 for the signaling function, as a function of $K_{\mathfrak{M}}$, can be considerably improved. But it is easy to see that this bound can never be lower than $m^{K_{\mathfrak{M}}}$, i.e., there is no simple algorithm Ω over relative black boxes, with $\Omega^*(\mathfrak{M}) < m^{K_{\mathfrak{M}}}$, which identifies all relative black boxes.

To see this, consider the abstract model of a combination lock with K states illustrated in Figure 34 ($X = \{x_1, \ldots, x_m\}$, $Y = \{y_1, y_2\} = \{0,1\}$, q_0 the initial state). The characteristic feature of this automaton is that the only way to generate the output letter 1 ("unlock") is to apply an input word ending in $x_{\alpha_1}x_{\alpha_2}\ldots x_{\alpha_K}$; call this word the combination. Combination locks differ from each other in their combinations, and so the number of different combination locks with K states is m^K.

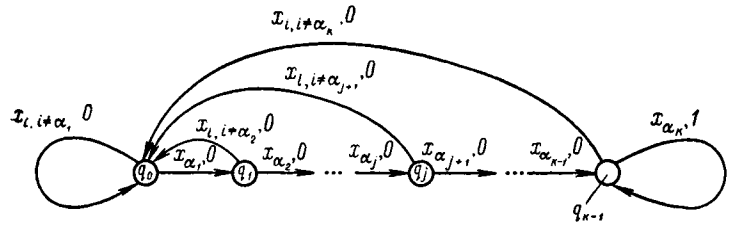


Figure 34

Let Ω be an arbitrary simple algorithm which residually identifies any relative black box. It will clearly identify ("unlock") any combination lock $\mathfrak{M}$ with K states and upper bound $K_{\mathfrak{M}} = K$ on its number of states [sic]. Let $\mathfrak{M}_1$ and $\mathfrak{M}_2$ be two such locks. When applied to these locks, the algorithm Ω will test each of them with a specific input word. Denote these input words by $x^{(1)} = x^{(1)}(1)\dots x^{(1)}(a)$ and $x^{(2)} = x^{(2)}(1)\dots x^{(2)}(b)$, respectively. Let $y^{(1)} = y^{(1)}(1)\dots y^{(1)}(a)$ and $y^{(2)} = y^{(2)}(1)\dots y^{(2)}(b)$ be the corresponding output words. Let us say that the algorithm Ω opens the lock $\mathfrak{M}_1$ ($\mathfrak{M}_2$) at the j-th step if $y^{(1)}(1) = y^{(1)}(2) = \dots = y^{(1)}(j-1) = 0$, but $y^{(1)}(j) = 1$ ($y^{(2)}(1) = y^{(2)}(2) = \dots = y^{(2)}(j-1) = 0$, $y^{(2)}(j) = 1$)). We shall say that the algorithm Ω opens the lock $\mathfrak{M}_1$ ($\mathfrak{M}_2$) if it opens it at some step $j \leqq a$ ($j \leqq b$). Obviously, the algorithm Ω must open any of these combination locks (since otherwise it could not residually identify them). Now let j be the smallest number such that the algorithm Ω first opens one of the locks $\mathfrak{M}_1$ and $\mathfrak{M}_2$ at the j-th step. Then, clearly, the output words

$$y^{(1)}(1)\dots y^{(1)}(j-1) \quad \text{and} \quad y^{(2)}(1)\dots y^{(2)}(j-1)$$

are equal (to be precise, are both blocks of zeros), and consequently the input words $x^{(1)}(1)\dots x^{(1)}(j)$ and $x^{(2)}(1)\dots x^{(2)}(j)$ are also equal. It follows that if the locks $\mathfrak{M}_1$ and $\mathfrak{M}_2$ are different, i.e., have different combinations, the algorithm Ω cannot open both of them at the same step. Since the number of different combinations for locks with K states is m^K and the algorithm Ω must open all of them, there must be a lock $\mathfrak{M}$ with K states which is not opened by the algorithm Ω before the m^K-th step. For this lock, therefore, $\Omega^*(\mathfrak{M}) \geqq m^K$. Since $K_{\mathfrak{M}} = K$, it follows that $\Omega^*(\mathfrak{M}) \geqq m^{K_{\mathfrak{M}}}$.

IV.3. Frequency criteria. Complexity of identification of almost all relative black boxes

In the preceding section we considered algorithms which identify *all* relative black boxes. The upper bounds for the signaling functions of these algorithms

turned out to be very high ($2K - 1$ for multiple algorithms, $C(K^2 \ln K)m^{2K}$ for simple algorithms). It is natural to ask whether we can substantially lower these bounds by considering only *almost all* automata, rather than *all* automata. In this section we shall show that this is indeed so (Theorems 4.3 and 4.4). However, this and similar questions have no rigorous meaning unless we first clarify the meaning of the phrase "almost all automata possess property E with a preassigned frequency." This will also yield a rigorous definition of frequency algorithms, which produce the correct outcome not always but (at least) with a preassigned frequency.

We first clarify when two automata (automaton graphs) are to be considered identical. Two automata (automaton graphs) $\mathfrak{M}_1$ and $\mathfrak{M}_2$ are said to be *identical* if and only if any two identically numbered vertices in the diagrams of $\mathfrak{M}_1$ and $\mathfrak{M}_2$ are connected by edges with the same direction and identical labels. Note that nonidentical automata (graphs) may be isomorphic. We recall (see e.g., Section 0.3) that the number of pairwise nonidentical automata with k states is $(nk)^{mk}$ and the number of pairwise nonidentical automaton graphs with k vertices is k^{mk}. We are dealing with automata (automaton graphs) with fixed alphabets $X = \{x_1, \ldots, x_m\}$ and $Y = \{y_1, \ldots, y_n\}$ ($m,n = \text{const} \geqq 2$), and all references to the class of all automata (automaton graphs) in fact refer to the class of all automata (automaton graphs) with these alphabets.

Now let $\mathscr{L}$ be the class of all pairwise nonidentical finite automata (automaton graphs) and E some property which any specific automaton (automaton graph) may or may not have. Assume that we have some partition $\{\mathscr{L}_\lambda\}$ of the class $\mathscr{L}$ into finite subclasses: $\mathscr{L} = \cup_\lambda \mathscr{L}_\lambda$. Let $\mathscr{L}_\lambda^E$ denote the set of all elements of $\mathscr{L}_\lambda$ which possess property E. One can consider the frequency with which this property occurs in $\mathscr{L}_\lambda$, i.e., the quotient

$$P_\lambda = \frac{|\mathscr{L}_\lambda^E|}{|\mathscr{L}_\lambda|}.$$

We shall say that *the property E occurs with frequency $1 - \varepsilon$ in a given partition* $\{\mathscr{L}_\lambda\}$ if $P \geqq 1 - \varepsilon$ for any λ. It is easy to see that the "frequency" pattern may vary considerably for the same property, depending on the partition $\{\mathscr{L}_\lambda\}$. Consequently, in formulating and solving actual problems the partition must be chosen on the basis of pertinent arguments. Thus, for example, since we are dealing with automata (automaton graphs) and with those of their properties related to the number of states, it is natural to consider partitions according to the number of states, i.e., a partition

into subclasses $\mathscr{L}_\lambda$ each of which contains all automata (automaton graphs) with λ states ($\lambda = 1, 2, \ldots$). Henceforth any statement that *automata (automaton graphs) possess a property with frequency* $1 - \varepsilon$ will refer specifically to this partition (unless explicitly stated otherwise).

Consider two partitions, of which the second is a refinement of the first (i.e., every subclass of the second partition is included in a subclass of the first). Then, obviously, if property E occurs with frequency $1 - \varepsilon$ in the second, finer partition, this surely holds for the first, coarser partition. One might say that the finer the partition, the more "uniform" the total frequency pattern. For example, consider the following refinement of our standard partition of the class of automata (by number of states). Let two automata belong to the same subclass if they can differ only in their output functions. In other words, any of these classes may be obtained from some automaton graph G with input alphabet X and indexed vertices $q_0, q_1, \ldots$ by assigning output labels to the edges in all possible ways; denote the subclass of pairwise nonidentical automata associated in this way with the graph G by $\widetilde{G}$. Henceforth, any statement that *a property E occurs uniformly with frequency* $1 - \varepsilon$ will refer to this partition (we call this a partion by graph G); it is finer than the partion by number of states.

Especially worthy of mention are cases in which the frequencies in the subclasses of these partitions tend to unity as the number of states k tends to infinity. When the partition is by number of states, we shall say that *almost all automata (automaton graphs) with k states*, or simply *almost all automata (automaton graphs) possess property E*; when the partition is by graphs, we shall say that *almost all automata with k states*, or simply *almost all automata, uniformly possess property E.*

We illustrate these concepts by a few examples. We know (Theorem 2.16′) that all automata $\mathfrak{M}$ have the property $B(\mathfrak{M}) \leqq 2\,|\,\mathfrak{M}\,| - 1$ (i.e., the degree of reconstructibility is less than twice the number of states). Is there a stronger property, namely, a much smaller upper bound, valid for almost all automata? Alternatively, can we perhaps prove that a stronger inequality holds with frequency $1 - \varepsilon$, for sufficiently small ε? In Chapter V we shall study the frequencies with which various parameters and spectra occur, and establish several results of this type. In particular, we shall prove (Theorem 5.10) that *almost all automata with k states have absolute degree of reconstructibility at most* $C_0 \log_m k$ (*and so at most** $[C_0 \log_m k]$), *where* C_0 *is a constant independent of k.* We shall be able to use this theorem (and various

* $[r]$ denotes the greatest natural number $\leqq r$.

other results of Chapter V) to investigate bounds for the complexity of automaton identification.

Employing the terms introduced above, one can give a frequency characterization of algorithms which do not always produce the correct outcome. For example, the statement that *an algorithm* Ω *identifies almost all relative black boxes* means that almost all automata $\mathfrak{M}$ have the following property: if the automaton $\mathfrak{M}$ is associated with some upper bound on the number of states and the algorithm Ω is applied to it as to a relative black box, then Ω will identify $\mathfrak{M}$. Similarly, when we say that an *algorithm* Ω *identifies absolute black boxes with frequency* $1 - \varepsilon$ (*uniformly with frequency* $1 - \varepsilon$), we mean that automata have the following property with frequency $1 - \varepsilon$ (uniformly with frequency $1 - \varepsilon$): the algorithm Ω, applied to the automaton as to an absolute black box, will identify it.

In this connection, consider the following multiple unconditional algorithm $\overline{\Omega}_0$, which is a modification of the algorithm Ω_0 of the preceding section. Let $\mathfrak{M}$ be a relative black box with the upper bound $K_{\mathfrak{M}} = K$ on the number of states.

1) Test $\mathfrak{M}$ with input words of length $[C_0 \log_m K]$, and construct a total tree of height $[C_0 \log_m K]$ compatible with $\mathfrak{M}$.

2) Apply the contraction procedure (which may not be unique here) to this tree and construct an automaton $\mathfrak{A}$.

Clearly, the signaling function of the algorithm $\overline{\Omega}_0$ is given by

$$\overline{\Omega}_0^*(\mathfrak{M}) = [C_0 \log_m K_{\mathfrak{M}}].$$

It follows from Theorem 5.10 (stated above) on the upper bound for the degree of reconstructibility of almost all automata that the algorithm $\overline{\Omega}_0$ initially identifies almost all relative black boxes. Thus, we have

THEOREM 4.3. *There is a multiple unconditiona algorithm* Ω *with signaling function*

$$\Omega^*(\mathfrak{M}) = [C_0 \log_m K_{\mathfrak{M}}]$$

which initially identifies almost all relative black boxes (C_0 is the constant of Theorem 5.10).

We now consider the identification of relative black boxes by simple algorithms. The following proposition follows directly from Lemma 2:

A. *For any natural* k, *one can effectively construct an input word* $d'(k)$ *of length* $[C_0 \log_m k] \cdot]2mk(\ln nk)k^{C_0}[$ *which residually distinguishes all automata with at most* k *states and absolute degree of reconstructibility at most* $[C_0 \log_m k]$.

Consider the following algorithm $\bar{\Omega}_1$, which is a modification of the algorithm Ω_1 of the preceding section. Let $\mathfrak{M}$ be a relative "black box" with upper bound $K_{\mathfrak{M}} = K$ on the number of states.

1) Construct an input word $d'(K)$ of length $[C_0 \log_m K] \cdot]2mK(\ln nK)K^{C_0}[$ as in Proposition A which residually distinguishes automata with at most K states and absolute degree of reconstructibility at most $[C_0 \log_m K]$. Then, testing $\mathfrak{M}$ with the input word $d'(K)$, find the corresponding output word y'.

2) Look for an automaton $\mathfrak{A}'$ with at most K states and absolute degree of reconstructibility at most $[C_0 \log_m K]$ which, like $\mathfrak{M}$, transforms the input word $d'(K)$ into the output word y'. If there are several such automata, choose one; if there are none, the outcome of the algorithm $\bar{\Omega}_1$ is either left undefined on the black box $\mathfrak{M}$ or is defined arbitrarily. The outcome of the algorithm is the automaton $\mathfrak{A}$ obtained from $\mathfrak{A}'$ by defining the initial state as the state to which the word $d'(K)$ takes $\mathfrak{A}'$.

What is the outcome of the algorithm $\bar{\Omega}_1$, applied to a black box $\mathfrak{M}$ with at most K states and absolute degree of reconstructibility at most $[C_0 \log_m K]$? Obviously, one can find an automaton $\mathfrak{A}'$ satisfying the above-mentioned conditions (since the black box $\mathfrak{M}$ itself already satisfies these conditions). Now consider the automaton $\mathfrak{A}$ produced by the algorithm $\bar{\Omega}_1$ applied to $\mathfrak{M}$. We claim that the automaton $\mathfrak{A}$ realizes the same operator as the black box $\mathfrak{M}$ after application of the input word $d'(K)$. Reasoning by *reductio ad absurdum*, assume that this is not so, i.e.,

$$T(\mathfrak{A}', q_0 d'(K)) \neq T(\mathfrak{M}, q_0 d'(K)).$$

Then the word $d'(K)$ does not residually distinguish the automata $\mathfrak{A}'$ and $\mathfrak{M}$ (recall that $\mathfrak{A}'$ and $\mathfrak{M}$ subject the word $d'(K)$ to the same transformation). But both these automata have at most K states and their absolute degree of reconstructibility is at most $[C_0 \log_m K]$, and by Proposition A the word $d'(K)$ must residually distinguish them. This contradiction proves that the algorithm $\bar{\Omega}_1$ residually identifies the black box $\mathfrak{M}$.

It is obvious that the signaling function of the algorithm Ω_1 is given by

$$\Omega_1^*(\mathfrak{M}) = [C_0 \log_m K_{\mathfrak{M}}] \cdot]2mK_{\mathfrak{M}}(\ln nK_{\mathfrak{M}})K_{\mathfrak{M}}^{C_0}[.$$

It is easily seen that for sufficiently large $K_{\mathfrak{M}}$ this quantity is bounded above by $K_{\mathfrak{M}}^{C_0+2}$.

Hence, and from Theorem 5.10 on the upper bound for the absolute degree of reconstructibility of almost all automata, we get

THEOREM 4.4. *There is a simple unconditional algorithm* Ω *with signaling function*

$$\Omega^*(\mathfrak{M}) \leqq K_{\mathfrak{M}}^{C_0+2}$$

which residually identifies almost all relative black boxes (C_0 is the constant of Theorem 5.10).

IV.4. General remarks on identification of absolute black boxes

After examining the identification of relative black boxes, we now consider identification of absolute black boxes. As we known (Section IV.1), identification may utilize only such information as can be acquired by applying input words and observing the corresponding output words (besides information on the input and output alphabets, which are fixed and assumed to be known).

The first question appearing in this connection is, does there exist an algorithm that identifies all absolute black boxes? It is easily seen that the answer is negative. For any algorithm for identification of an arbitrary black box $\mathfrak{M}$ yields its outcome on the basis of a finite test of $\mathfrak{M}$, i.e., a finite tree compatible with $\mathfrak{M}$. But for any given tree there is an infinite set of compatible automata (black boxes), and the algorithm will clearly produce the same answer for all these automata irrespective of the operators that they realize.

Thus there can be no algorithm which identifies all absolute black boxes. What can one say about an algorithm identifying *almost all* absolute black boxes? Again the answer is negative:

THEOREM 4.5. *There is no algorithm which identifies almost all absolute black boxes.*

Proof. Let Ω be an arbitrary algorithm over absolute black boxes. To prove the theorem, we must show that

$$\lim_{k\to\infty} \frac{|\mathscr{L}_\Omega(k)|}{|\mathscr{L}(k)|} \neq 0,$$

where $\mathscr{L}(k)$ is the set of all pairwise nonidentical automata with k states, and $\mathscr{L}_\Omega(k)$ the set of all pairwise nonidentical automata with k states which are not identified by Ω. Since simple and multiple algorithms are associated with different identification concepts (that for simple algorithms being the weaker), we must consider these two types of algorithm separately.

Let Ω be a multiple algorithm. Let $\mathfrak{M}_0$ be some automaton which always generates the same letter, say y_1. Let G be an arbitrary automaton graph, and, as before, $\tilde{G}$ the set of pairwise nonidentical automata obtained from G by assigning output labels to its edges in all possible ways. Let $\tilde{G}_\Omega$ denote the set of all pairwise nonidentical automata in $\tilde{G}$ which are indistinguishable from $\mathfrak{M}_0$ by input words of length $l_0 = \Omega^*(\mathfrak{M}_0)$, but realize operators different from $\Omega(\mathfrak{M}_0)$. Obviously, the algorithm Ω will not identify any of these automata. Let G' denote the graph obtained from G by deleting all vertices inaccessible from q_0.

Obviously

$$\frac{|\tilde{G}_\Omega|}{|\tilde{G}|} = \frac{|\tilde{G}'_\Omega| \cdot n^{m(|G| - |G'|)}}{|\tilde{G}'| \cdot n^{m(|G| - |G'|)}} = \frac{|\tilde{G}'_\Omega|}{|\tilde{G}'|}.$$

Now in any automaton graph there are at most $m^{l_0+1} - 1$ vertices which are accessible from q_0 by input words of length at most l_0. Thus the number of automata in $\tilde{G}'$ which realize different operators and are indistinguishable from $\mathfrak{M}_0$ by input words of length l_0 is at least $n^{m(|G'| - (m^{l_0+1}-1))}$.

Hence,

$$|\tilde{G}'_\Omega| \geqq n^{m(|G'| - (m^{l_0+1}-1))} - 1$$

and if $|G'| > m^{l_0+1}$,

$$|\tilde{G}'_\Omega| \geqq n^{m(|G'| - m^{l_0+1})}.$$

It follows that if $|G'| > m^{l_0+1}$,

$$\frac{|\tilde{G}'_\Omega|}{|\tilde{G}'|} \geqq \frac{n^{m(|G'| - m^{l_0+1})}}{n^{m|G'|}} = n^{-m^{l_0+2}},$$

and so

$$\frac{|\tilde{G}_\Omega|}{|\tilde{G}|} \geqq n^{-m^{l_0+2}}.$$

Now it is not difficult to show (see, e.g., Chapter V, Theorem 5.4) that the ratio of the number of all automaton graphs G with k vertices such that $|G'| > m^{l_0+2}$ to the number of all automaton graphs with k vertices tends to unity as $k \to \infty$. Together with the preceding arguments, this implies that

$$\lim_{k \to \infty} \frac{|\mathscr{L}_\Omega(k)|}{|\mathscr{L}(k)|} \geqq n^{-m^{l_0+2}} \neq 0.$$

The case of a simple algorithm Ω is treated analogously. This proves the theorem.

The negative result implied by this theorem motivates the following question: Given arbitrary $\varepsilon > 0$, is there an algorithm which identifies absolute black boxes with frequency $1 - \varepsilon$?* A positive answer to this question will be given in the following sections.

IV.5. Iterative algorithms

In Section 1 of this chapter we introduced general algorithms, multiple and simple, over absolute black boxes. We shall now consider a particular case—the so-called iterative algorithms, which will play a significant role in the sequel. Our aim is to show that, using these algorithms, one can identify absolute black boxes with any preassigned frequency.

We first define a multiple iterative algorithm over absolute black boxes.

A *multiple iterative algorithm* is defined by a computable nondecreasing arithmetical function $\phi(s)$.** We call this function the *control function* of the algorithm and denote the algorithm itself by Σ_ϕ. Applied to a black box $\mathfrak{M}$, the algorithm Σ_ϕ operates as follows.

Step 1. Choose a fixed automaton $\mathfrak{A}_0$ (say, the automaton illustrated in Figure 35), and call it the initial hypothesis. Then, testing the black box $\mathfrak{M}$ with input words of length $\phi(|\mathfrak{A}_0|)$, determine whether the hypothesis $\mathfrak{A}_0$ is distinguishable from $\mathfrak{M}$ by input words of length $\phi(|\mathfrak{A}_0|)$. If not, the algorithm halts and its outcome is defined to be $\mathfrak{A}_0$. If it is, construct the total tree of height $\phi(|\mathfrak{A}_0|)$ produced by testing $\mathfrak{M}$ with input words of length $\phi(|\mathfrak{A}_0|)$, and apply the contraction procedure (Section II.4). The resulting automaton $\mathfrak{A}_1$ is defined to be the hypothesis generated at Step 1.† Now proceed to Step 2.

Step i ($i = 2, 3, \ldots$) is analogous to Step 1, except that the initial hypothesis $\mathfrak{A}_0$ is replaced by the hypothesis $\mathfrak{A}_{i-1}$ generated by the preceding step. If this hypothesis is indistinguishable from $\mathfrak{M}$ by input words of length $\phi(|\mathfrak{A}_{i-1}|)$, the algorithm halts and its outcome is by definition $\mathfrak{A}_{i-1}$. If it is distinguishable from $\mathfrak{M}$, construct a new hypothesis $\mathfrak{A}_i$ by contracting

* Of course, one is interested only in the case $\varepsilon < 1$; this will be assumed throughout the sequel without special mention.

** A function is arithmetical if it maps natural numbers onto natural numbers.

† If the contraction procedure does not yield a unique result, choose one of the possible results (it is immaterial which is chosen).

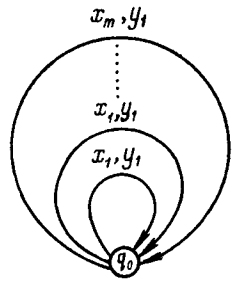


Figure 35

the tree produced by testing $\mathfrak{M}$ with input words of length $\phi(|\mathfrak{A}_{i-1}|)$, and proceed to Step $i + 1$.

It is quite clear from the definition of an iterative algorithm that it utilizes no information on the number of states of the black box.

EXAMPLE. Consider the multiple iterative algorithm Σ_{s+1} with control function $\phi(s) = s + 1$. Imagine a black box $\mathfrak{M}_1$ with alphabets $X = \{x_1, x_2\} = \{0,1\}$ and $Y = \{y_1, y_2\} = \{0,1\}$ which checks for divisibility by 3: for input $x(1) \ldots x(t)$, the automaton generates a word $y(1) \ldots y(t)$ such that, for every $1 \leqq i \leqq t$,

$$y(i) = \begin{cases} 1, \text{ if the word } x(1) \ldots x(i) \text{ is the binary expansion of} \\ \quad \text{a number divisible by 3} \\ \\ 0 \quad \text{otherwise.} \end{cases}$$

Let us apply the algorithm Σ_{s+1} to the black box $\mathfrak{M}_1$. As initial hypothesis take the automaton illustrated in Figure 36 (which is the automaton of Figure 35 for $X = \{0,1\}$ and $Y = \{0,1\}$). The outcome of Step 1 (contraction of the tree of Figure 37*a*) is the automaton $\mathfrak{A}_1$ illustrated in Figure 37*b*.* The outcome of Step 2 (contraction of the tree of Figure 38*a*) is the automaton $\mathfrak{A}_2$ of Figure 38*b*. The outcome of Step 3 (contraction of the tree of Figure 39*a*) is the automaton $\mathfrak{A}_3$ illustrated in Figure 39*b*. Finally, the algorithm halts at Step 4 and the outcome is the automaton $\mathfrak{A}_3$. It can be shown (we leave this to the reader) that the automaton $\mathfrak{A}_3$ indeed checks for divisibility by 3, and so is a suitable model of the black box $\mathfrak{M}_1$. This means that the algorithm Σ_{s+1} initially identifies $\mathfrak{M}_1$ (and does so without utilizing any

* Generally speaking, other automata may result at this step, since the contraction procedure is not unique here.

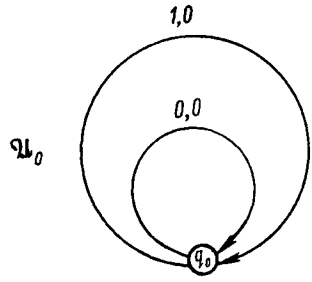


Figure 36

information on the number of states!). Note moreover that $\Sigma^*_{s+1}(\mathfrak{M}_1) = 4$ (since the maximal length of input words with which the algorithm Σ_{s+1} tests $\mathfrak{M}_1$ is 4).

This positive result notwithstanding, it is not hard to devise black boxes which are not identified by Σ_{s+1}. For example, Σ_{s+1} cannot identify an automaton $\mathfrak{M}_2$ which accepts input words of length greater than two, i.e., responds to input words $x(1)\ldots x(t)$ with output word $y(1)\ldots y(t)$, where

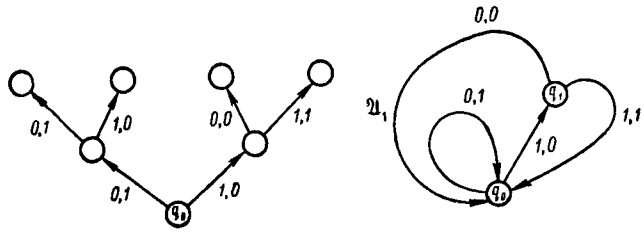
$$y(t) = \begin{cases} 1 & \text{if} \quad t > 2, \\ 0 & \text{if} \quad t \leqq 2. \end{cases}$$

The reader will easily convince himself that the algorithm Σ_{s+1} applied to $\mathfrak{M}_2$ halts at the initial hypothesis, illustrated in Figure 36.

We now prove the following important property of multiple iterative algorithms:

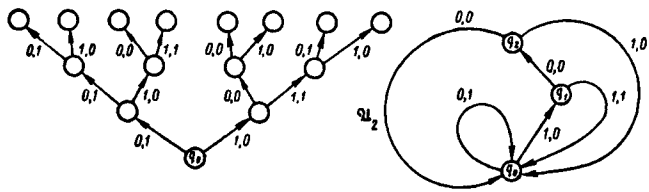
A multiple iterative algorithm Σ_ϕ applied to an arbitrary black box $\mathfrak{M}$ must halt at some step.

Let $\mathfrak{M}$ be a black box to which an iterative algorithm Σ_ϕ is applied. It follows from the definition of the contraction procedure (Section II.4) that, if V_1 and V_2 are two total trees compatible with $\mathfrak{M}$, such that the height of V_2 is at least that of V_1, then the automaton obtained by contracting V_2 has at least as many states as that obtained from V_1. In view of this property and the fact that the control function is nondecreasing, it is easy to prove by induction on i that the hypothesis generated at Step i has at least as many states as that generated at the preceding step (for $i = 1$, at least as many states as the initial hypothesis, Figure 35). It follows that the hypotheses generated at Steps i and i', where $i' \geqq i + 1$, must be different (otherwise the algorithm Σ_ϕ would have halted at Step $i + 1$). On



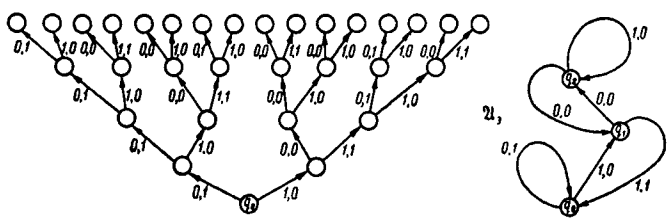
a *b*

Figure 37



a *b*

Figure 38



a *b*

Figure 39

the other hand, none of the hypotheses can have more states than the black box $\mathfrak{M}$, since contraction of a tree compatible with $\mathfrak{M}$ cannot yield an automaton with more states than $\mathfrak{M}$. All this implies that the algorithm Σ_ϕ, applied to $\mathfrak{M}$, must halt at some step.

Let Σ_ϕ be an arbitrary multiple iterative algorithm with control function $\phi(s)$. What can we say of its signaling function? As already mentioned, none of the hypotheses generated during application of Σ_ϕ to a black box $\mathfrak{M}$

can have more states than $\mathfrak{M}$. Moreover, the control function $\phi(s)$ is non-decreasing by definition.

It follows that the signaling function $\Sigma_\phi^*(\mathfrak{M})$ of an iterative algorithm Σ_ϕ satisfies the inequality

$$\Sigma_\phi^*(\mathfrak{M}) \leqq \phi(|\mathfrak{M}|).$$

We now define simple iterative algorithms over absolute black boxes.

A *simple iterative algorithm* is defined by a computable function $g(s)$ defined on natural numbers, whose values are words over the input alphabet X such that, for any natural s, $g(s)$ is an initial segment of $g(s+1)$ (so that $g(s+1)$ is not shorter than $g(s)$). We call this function the *control function* of the simple algorithm and denote the algorithm itself by Π_g. Applied to a black box $\mathfrak{M}$, the algorithm Π_g operates as follows.

Step 1. Choose some fixed automaton $\mathfrak{A}_0$ (say the automaton illustrated in Figure 35) and call it the initial hypothesis. Then test the black box $\mathfrak{M}$ with the input word $g(|\mathfrak{A}_0|)$ (so that $\mathfrak{M}$ goes to the state $q_0 g(|\mathfrak{A}_0|)$), and, according to the result of this test, check whether the hypothesis $\mathfrak{A}_0$ is distinguishable from $\mathfrak{M}$ by the input word $g(|\mathfrak{A}_0|)$. If not, the algorithm halts and its outcome is by definition the automaton $\langle \mathfrak{A}_0, q_0 g(|\mathfrak{A}_0|)\rangle$, i.e., the automaton obtained from $\mathfrak{A}_0$ by stipulating that its initial state is that to which the input word $g(|\mathfrak{A}_0|)$ takes $\mathfrak{A}_0$. If it is distinguishable, use the result of the test to construct a minimal automaton which is indistinguishable from $\mathfrak{M}$ by the input word $g(|\mathfrak{A}_0|)$. This automaton $\mathfrak{A}_1$ is defined to be the hypothesis generated at Step 1. Now proceed to Step 2.

Step i ($i = 2, 3, \ldots$) is analogous to Step 1, except that the initial hypothesis $\mathfrak{A}_0$ is replaced by the hypothesis $\mathfrak{A}_{i-1}$ generated at the preceding step. First determine the output word which is the response of the black box $\mathfrak{M}$ to the input word $g(|\mathfrak{A}_{i-1}|)$ (since the previous steps have already established the response of the black box $\mathfrak{M}$ to the input word $g(|\mathfrak{A}_{i-2}|)$, all that remains at this step is to determine the response of $\mathfrak{M}$, in the state to which it is taken by the word $g(|\mathfrak{A}_{i-2}|)$, to the "tail" of the word $g(|\mathfrak{A}_{i-1}|)$ following $g(|\mathfrak{A}_{i-2}|)$). Now check whether the hypothesis $\mathfrak{A}_{i-1}$ is distinguishable from the black box $\mathfrak{M}$ by the input word $g(|\mathfrak{A}_{i-1}|)$. If not, the algorithm halts and its outcome is the automaton $\langle \mathfrak{A}_{i-1}, q_0 g(|\mathfrak{A}_{i-1}|)\rangle$. If it is, construct a new hypothesis $\mathfrak{A}_i$—a minimal automaton which is indistinguishable from $\mathfrak{M}$ by the input word $g(|\mathfrak{A}_{i-1}|)$—and proceed to Step $i+1$.

It is clear that, like its multiple counterpart, a simple iterative algorithm utilizes no information on the number of states of the black box.

EXAMPLE. Consider the simple iterative algorithm Π_{g_0} with the control function $g_0(s)$ defined as the word 01010101 ... of length $3s$ (for example, $g(2) = 010101$). Let $\mathfrak{M}_3$ be a black box with alphabets

$$X = \{x_1, x_2\} = \{0,1\} \text{ and } Y = \{y_1, y_2\} = \{0,1\},$$

which checks whether the input word contains an even number of ones, i.e., its response to the input word $x(1) \dots x(t)$ is the word $y(1) \dots y(t)$, where for any $1 \leqq i \leqq t$

$$y(i) = \begin{cases} 1 & \text{if } x(1) \dots x(i) \text{ contains an even number of ones,} \\ 0 & \text{otherwise.} \end{cases}$$

Apply the algorithm Π_{g_0} to $\mathfrak{M}_3$. For the initial hypothesis $\mathfrak{A}_0$, take the automaton of Figure 36. The outcome of Step 1 is the automaton $\mathfrak{A}_1$ illustrated in Figure 40,* while that of Step 2 is the automaton $\mathfrak{A}_2$ illustrated in Figure 41. At Step 3 the algorithm halts and its outcome is the automaton

$$\langle \mathfrak{A}_2, q_0 g(|\mathfrak{A}_2|) \rangle = \langle \mathfrak{A}_2, q_0 g(2) \rangle = \langle \mathfrak{A}_2, q_0 010101 \rangle = \langle \mathfrak{A}_2, q_1 \rangle.$$

It is not difficult to see that the automaton $\langle \mathfrak{A}_2, q_1 \rangle$ realizes the same operator as the black box $\mathfrak{M}_3$, when the latter starts from the state to which application of the algorithm Π_{g_0} has taken it (i.e., after application of the word $g(|\mathfrak{A}_2|) = 010101$). It follows that the algorithm Π_{g_0} residually identifies $\mathfrak{M}_3$, and also $\Pi^*_{g_0}(\mathfrak{M}_3) = 6$.

Again, it is not hard to find examples of black boxes which cannot be identified by the algorithm Π_{g_0}.

Just as in the multiple case, one can prove the following important property of simple iterative algorithms:

A simple iterative algorithm Π_g, applied to any black box $\mathfrak{M}$, must halt at some step.

To see this, note the following direct consequence of the definition of the control function $g(s)$: If $s' \geqq s$, then the minimal automaton which is indistinguishable from $\mathfrak{M}$ by the word $g(s')$ has at least as many states as the minimal automaton indistinguishable from $\mathfrak{M}$ by $g(s)$. The rest of the argument is exactly the same as in the multiple case. Let Π_g be an arbitrary simple iterative algorithm with control function $g(s)$. Let $l(x)$ denote the length of the word x. It is easy to see that the signaling function $\Pi^*_g(\mathfrak{M})$ of the algorithm Π_g satisfies the inequality

$$\Pi^*_g(\mathfrak{M}) \leqq l(g(|\mathfrak{M}|)).$$

* Generally speaking, this step is not unambiguous, since there are other minimal automata which are indistinguishable from $\mathfrak{M}_3$ by the input word $g(|\mathfrak{A}_0|) = g(1) = 010$.

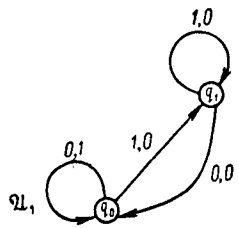


Figure 40

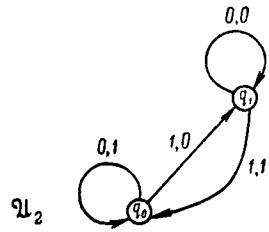


Figure 41

REMARK. In a certain sense, the iterative algorithms are a formal version of incomplete induction. For at each step the algorithm constructs a hypothesis which is tested only finitely many times (in a multiple algorithm—by several input words, in a simple algorithm—by a single input word). If this finite test does not refute the hypothesis, the latter is accepted as true (the outcome is produced). If it is refuted, a new hypothesis is constructed (as simply as possibly, i.e., with the minimal number of states). We may thus interpret identification of black boxes by an iterative algorithm as identification by incomplete induction. In the next sections it will be shown that iterative algorithms (both multiple and simple) are capable of identifying the majority of absolute black boxes. This result may be interpreted as a justification of the use of incomplete induction in black-box identification. One may therefore hope for a justification of incomplete induction in more extensive classes of problems.

IV.6. Identification of absolute black boxes by multiple algorithms, with arbitrary preassigned frequency

In this section we shall show that by means of multiple algorithms one can identify absolute black boxes with any preassigned frequency. We shall prove an even stronger result (Theorem 4.6), implying that this can be done within the class of multiple iterative algorithms, and moreover uniformly.

We recall the definition of uniform identification. An algorithm is said to identify absolute black boxes uniformly with frequency $1 - \varepsilon$ if, for any automaton graph G,

$$\frac{|\tilde{G}^{\Sigma}|}{|\tilde{G}|} \geqq 1 - \varepsilon,$$

where $\tilde{G}^{\Sigma}$ is the set of all automata in $\tilde{G}$ which the algorithm Σ identifies (as absolute black boxes).

THEOREM 4.6. *For any $\varepsilon > 0$, there is a multiple iterative algorithm $\Sigma_{\phi_\varepsilon}$ which initially identifies absolute black boxes uniformly with frequency $1 - \varepsilon$.*

The proof of this theorem will require several auxiliary concepts and propositions.

In Section II.12 we defined the accessibility spectrum of an automaton $\mathfrak{M}$ as the function of $D_{\langle \mathfrak{M}, q_0 \rangle}(l)$ equal to the number of states accessible from q_0 by words of length at most l. Similarly, the *accessibility spectrum of an automaton graph G* is the function $D_{\langle G, q_0 \rangle}(l)$ defined as the number of vertices accessible from q_0 by words of length at most l. Besides the accessibility spectrum we shall need another spectrum—the saturation spectrum which we now define. For any natural number l, consider a total tree $V^{(l)}$ of height l compatible with the automaton $\mathfrak{M}$. The *saturation spectrum* of the automaton $\mathfrak{M}$ is the function $F_{\langle \mathfrak{M}, q_0 \rangle}(l)$ defined as the weight of the tree $V^{(l)}$, or, equivalently, the number of states of the automaton obtained by contracting the tree $V^{(l)}$. It is obvious that the saturation spectrum is a nondecreasing function, and if the automaton has finite weight μ its spectrum is bounded by the constant μ. In a certain sense, the saturation spectrum of an automaton resembles the accessibility spectrum of the operator that it realizes, with the difference that in assigning vertices to the basis only a finite truncation of the infinite tree is considered, rather than the entire tree. Clearly,

$$F_{\langle \mathfrak{M}, q_0 \rangle}(l) \leqq D_{\langle \mathfrak{M}, q_0 \rangle}(l).$$

We shall now establish a certain relation between the accessibility and saturation spectra. Let G be an automaton graph, and denote by $\tilde{G}_{F(l) \leqq s}$ the subset of $\tilde{G}$ consisting of all automata $\mathfrak{M}$ such that $F_{\langle \mathfrak{M}, q_0 \rangle}(l) \leqq s$. Consider the quotient $|\tilde{G}_{F(l) \leqq s}| / |\tilde{G}|$ (which depends on l and s). We shall prove a lemma which gives an upper bound for this quotient in terms of the accessibility spectrum $D_{\langle G, q_0 \rangle}(l)$ of G.

LEMMA 3. *Given an automaton graph G with accessibility spectrum $D(l)$. Then, for any natural numbers l and s,*

$$\frac{|\tilde{G}_{F(l) \leqq s}|}{|\tilde{G}|} \leqq \frac{(ns)^{ms}}{n^{D(l)-1}}$$

(where m and n are the cardinalities of the input alphabet X and the output alphabet Y, respectively).

Proof. Let d and d' be edges in the diagrams of automata $\mathfrak{M} \in \tilde{G}$ and $\mathfrak{M}' \in \tilde{G}$, respectively, which correspond to the same edge of the graph G; call them corresponding edges. For fixed l, partition the set of automata $\tilde{G}$ into disjoint classes as follows: two automata belong to the same class if and only if any two corresponding edges of their diagrams which are inaccessible from q_0 by words of length at most l have identical output labels. In other words, automata in the same class may differ only in the output labels of edges accessible from q_0 by input words of length at most l. Let U be one of these classes. We extend the notation $\tilde{G}_{F(l) \leqq s}$, introduced above for $\tilde{G}$, to the set U, so that $U_{F(l) \leqq s}$ will denote the subset of U consisting of all automata $\mathfrak{M}$ such that $F_{\langle \mathfrak{M}, q_0 \rangle}(l) \leqq s$. Obviously, automata in the class U are distinguishable, moreover by input words of length l (since otherwise U, and *a fortiori* $\tilde{G}$, would contain identical automata). Therefore, if an automaton $\mathfrak{M} \in U$ is indistinguishable from some automaton $\mathfrak{A}$ by input words of length l, this is no longer true for any other $\mathfrak{M}' \in U$. It follows that $|U_{F(l) \leqq s}|$ cannot exceed the number of pairwise nonidentical automata with s states, i.e., it is at most $(ns)^{ms}$. On the other hand, at least $D(l) - 1$ edges of the graph G are accessible from q_0 by words of length at most l (since every vertex accessible by an input word x, except possibly q_0 itself, is the endpoint of at least one edge accessible by the same input word). Thus the number of automata in the class U is at least $n^{D(l)-1}$.

It follows that

$$\frac{|U_{F(l) \leqq s}|}{|U|} \leqq \frac{(ns)^{ms}}{n^{D(l)-1}}.$$

Hence (since the classes U are disjoint) the statement of our lemma.

We can now proceed to a more direct investigation of the actual identification procedure. Let G be an arbitrary automaton graph, and denote by G' the graph obtained from G by deleting all vertices not accessible from q_0. Obviously,

$$\frac{|\tilde{G}^{\Sigma}|}{|\tilde{G}|} = \frac{|\tilde{G}'^{\Sigma}|}{|\tilde{G}'|}.$$

This proves

LEMMA 4. *Let Σ be an algorithm such that for any automaton graph G all of whose vertices are accessible from q_0*

$$\frac{|\tilde{G}^{\Sigma}|}{|\tilde{G}|} \geqq 1 - \varepsilon.$$

Then the algorithm Σ identifies absolute black boxes uniformly with frequency $1 - \varepsilon$.

We shall say that an algorithm Σ_ϕ with control function ϕ, applied to a black box $\mathfrak{M}$, *admits an s-error* ($s = 1, 2, 3, \dots$) if the automata obtained by contracting the total tree of height $\phi(s)$ compatible with $\mathfrak{M}$ have at most s states (i.e., $F_{\langle \mathfrak{M}, q_0 \rangle}(\phi(s)) \leqq s$) and at least one of these automata realizes an operator different from $T(\mathfrak{M})$. The meaning of this definition is clarified by the following simple

LEMMA 5. *If the algorithm Σ_ϕ, applied to an automaton $\mathfrak{M}$, admits no s-error for any natural number s, the algorithm Σ_ϕ identifies $\mathfrak{M}$.*

Proof. Suppose that Σ_ϕ does not identify $\mathfrak{M}$. This means that Σ_ϕ applied to $\mathfrak{M}$ halts at some hypothesis $\mathfrak{A}_i$ which realizes an operator different from $T(\mathfrak{M})$. By the definition of the algorithm Σ_ϕ, it can halt at a hypothesis $\mathfrak{A}_i$ only if the total tree of height $\phi(|\mathfrak{A}_i|)$ compatible with $\mathfrak{M}$ is also compatible with $\mathfrak{A}_i$. But then, when this tree is contracted, the resulting automaton cannot have more than $|\mathfrak{A}_i|$ states. This means that $\Sigma_{\phi_\varepsilon}$ necessarily admits an s-error, with $s = |\mathfrak{A}_i|$. This implies the lemma.

Let $O(\tilde{G}, \phi(s), s)$ denote the set of all automata in $\tilde{G}$ for which the algorithm Σ_ϕ admits an s-error. Obviously,

$$O(\tilde{G}, \phi(s), s) \subseteq \tilde{G}_{F(\phi(s)) \leqq s}.$$

LEMMA 6. *Let $\phi(s)$ be a control function such that for any automaton graph G all of whose vertices are accessible from q_0 and for any natural number s*

$$\frac{|O(\tilde{G}, \phi(s), s|}{|\tilde{G}|} \leqq \frac{\varepsilon}{(s+1)^2}.$$

Then the algorithm Σ_ϕ identifies absolute black boxes uniformly with frequency $1 - \varepsilon$.

Proof. By Lemma 5, if the algorithm $\Sigma_{\phi_\varepsilon}$ does not identify some automaton (absolute black box), the latter must belong to the union $\bigcup_{s=1}^{\infty} O(\tilde{G}, \phi(s), s)$. Therefore, if the function $\phi(s)$ satisfies the inequality

$$\frac{\left| \bigcup_{s=1}^{\sigma} O(\tilde{G}, \phi(s), s) \right|}{|\tilde{G}|} < \varepsilon,$$

then

$$\frac{|\tilde{G}^{\Sigma_\phi}|}{|\tilde{G}|} \geqq 1 - \varepsilon.$$

Now assume that $\phi(s)$ satisfies the assumptions of the lemma. Let G be an arbitrary automaton graph all of whose vertices are accessible from q_0. Then

$$\frac{\left|\bigcup_{s=1}^{\infty} O(\tilde{G}, \phi(s), s)\right|}{|\tilde{G}|} \leqq \sum_{s=1}^{\infty} \frac{|O(\tilde{G}, \phi(s), s)|}{|\tilde{G}|} \leqq \sum_{s=1}^{\infty} \frac{\varepsilon}{(s+1)^2} < \varepsilon.$$

Consequently,

$$\frac{|\tilde{G}^{\Sigma_\phi}|}{|\tilde{G}|} \geqq 1 - \varepsilon.$$

This inequality and Lemma 4 complete the proof.

It is now not difficult to prove Theorem 4.6. Let $\varepsilon > 0$ be arbitrary but fixed. Let G be any automaton graph and $\phi_\varepsilon(s)$ an arbitrary control function. Assume, moreover, that for some s_0

$$\phi_\varepsilon(s_0) \geqq 2|G|.$$

Consider an arbitrary automaton $\mathfrak{M}$ from the set $\tilde{G}$ and the total tree of height $\phi_\varepsilon(s_0)$ compatible with $\mathfrak{M}$. Since $\phi_\varepsilon(s_0) \geqq 2|\mathfrak{M}|$, it follows from Theorem 2.16′ that the automaton obtained by contracting this tree must realize the same operator as $\mathfrak{M}$. This means that the algorithm $\Sigma_{\phi_\varepsilon}$ applied to $\mathfrak{M}$ admits no s-error for $s = s_0$. We have thus established that, for s such that $\phi_\varepsilon(s) \geqq 2|G|$, the set $O(\tilde{G}, \phi_\varepsilon(s), s)$ is empty, and so

$$\frac{|O(\tilde{G}, \phi_\varepsilon(s), s)|}{|\tilde{G}|} = 0 < \frac{\varepsilon}{(s+1)^2}.$$

By Lemma 6, this implies that the algorithm $\Sigma_{\phi_\varepsilon}$ with control function $\phi_\varepsilon(s)$ identifies absolute black boxes uniformly with frequency $1 - \varepsilon$, provided the function $\phi_\varepsilon(s)$ has the following property: for any automaton graph G all of whose vertices are accessible from q_0, and for any s such that

$$\frac{|O(G, \phi_\varepsilon(s), s)|}{|\tilde{G}|} \leqq \frac{\varepsilon}{(s+1)^2}.$$

Thus, to complete the proof of our theorem it will suffice to construct a computable arithmetical function $\phi_\varepsilon(s)$ possessing this property.

Since $O(\tilde{G}, \phi_\varepsilon(s), s) \subseteq \tilde{G}_{F(\phi_\varepsilon(s)) \leqq s}$, it follows that

$$\frac{|O(\tilde{G}, \phi_\varepsilon(s), s)|}{|\tilde{G}|} \leqq \frac{|\tilde{G}_{F(\phi_\varepsilon(s)) \leqq s}|}{|\tilde{G}|}.$$

By Lemma 3,

$$\frac{|\tilde{G}_{F(\phi_\varepsilon(s)) \leqq s}|}{|\tilde{G}|} \leqq \frac{(ns)^{ms}}{n^{D(\phi_\varepsilon(s))-1}},$$

where $D(l)$ is the accessibility spectrum of the graph G. Let G be an automaton graph all of whose vertices are accessible from q_0, and $\phi_\varepsilon(s) < 2|G|$. Then, obviously, $D(\phi_\varepsilon(s)) > \frac{1}{2}\phi_\varepsilon(s)$ (since $D(l) \leqq |G|$ for any l and $D(l) \geqq l$ for $l \leqq |G|$). Consequently,

$$\frac{(ns)^{ms}}{n^{D(\phi_\varepsilon(s))-1}} < \frac{(ns)^{ms}}{n^{(1/2)\phi_\varepsilon(s)-1}}.$$

Hence a computable nondecreasing arithmetical function $\phi_\varepsilon(s)$ will satisfy the requirements of Theorem 4.6 if, for all natural numbers s,

$$\frac{(ns)^{ms}}{n^{(1/2)\phi_\varepsilon(s)-1}} \leqq \frac{\varepsilon}{(s+1)^2}.$$

Solving this inequality for $\phi_\varepsilon(s)$, we get

$$\phi_\varepsilon(s) \geqq 2ms \log_n ns + 4 \log_n(s+1) + 2 + \log_n(1/\varepsilon).$$

Thus, the algorithm Σ_ϕ with control function

$$\phi_\varepsilon(s) =]2ms \log_n ns + 4 \log_n(s+1) + 2[+]2 \log_n(1/\varepsilon)[$$

(it is easily seen that this is indeed a computable function of s) will identify absolute black boxes uniformly with frequency $1 - \varepsilon$. This completes the proof.

As for the signaling function of the algorithm, it follows from the reasoning in Section IV.1 that $\Sigma^*_{\phi_\varepsilon}(\mathfrak{M}) \leqq \phi_\varepsilon(|\mathfrak{M}|)$, and so

$$\Sigma^*_{\phi_\varepsilon}(\mathfrak{M}) \leqq]2m|\mathfrak{M}| \log_n n|\mathfrak{M}| + 4 \log_n(|\mathfrak{M}| + 1) + 2[+]2 \log_n(1/\varepsilon)[.$$

One can now ask whether this bound can be lowered, i.e., whether an iterative algorithm can be constructed with a smaller signaling function (as a function of the number of states of the black box) which will also identify absolute black boxes uniformly with frequency $1 - \varepsilon$. The answer to this question will be given in the next section.

IV.7. Bound on the complexity of uniform identification

We shall first show that, *for any multiple algorithm* Σ (*not necessarily iterative*) *which identifies absolute black boxes uniformly with frequency* $1 - \varepsilon$ (*where* $\varepsilon < 1$), *the best possible asymptotic upper bound for the signaling function, as a function of* $|\mathfrak{M}|$, *is* $|\mathfrak{M}|$.

Assume that this is not so. Then, for some $\varepsilon < 1$, there is an algorithm Σ which identifies absolute black boxes uniformly with frequency $1 - \varepsilon$ and, for any C, possesses the following property:

A. There is a natural number $k(C)$ such that, for all black boxes $\mathfrak{M}$ with $|\mathfrak{M}| = k(C)$, $\Sigma^*(\mathfrak{M}) \leqq |\mathfrak{M}| - C$.

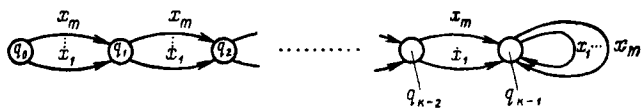


Figure 42

For each natural k construct the automaton graph G_k illustrated in Figure 42. Let C be some fixed number, and consider the graph $G_{k(C)}$. Clearly, the outcome of the algorithm Σ applied to the automata derived from the graph $G_{k(C)}$ is independent of the output labels assigned to the edges issuing from the last C vertices. There are mC such edges in all; they can be assigned output labels from the alphabet Y in n^{mC} different ways, and the operators realized by the resulting automata depend essentially on these output labels. Therefore,

$$\frac{|\tilde{G}^{\Sigma}_{k(C)}|}{|\tilde{G}_{k(C)}|} \leqq \frac{1}{n^{mC}}.$$

Obviously, for any $\varepsilon < 1$ one can select a constant C_ε such that the right-hand side of this inequality is smaller than $1 - \varepsilon$ for $C = C_\varepsilon$. It follows that if the algorithm Σ possesses property A for $C = C_\varepsilon$, it cannot identify absolute black boxes uniformly with frequency $1 - \varepsilon$. This contradiction proves our assertion concerning the bound on the signaling function.

The next theorem gives an upper bound for the signaling function which is asymptotically equal to the above bound (and is therefore best possible asymptotically).

THEOREM 4.7. *For any $\varepsilon > 0$ there is a multiple iterative algorithm $\Sigma_{\phi_\varepsilon}$ which initially identifies absolute black boxes uniformly with frequency $1 - \varepsilon$ and has a signaling function such that*

$$\Sigma^*_{\phi_\varepsilon}(\mathfrak{M}) \leqq |\mathfrak{M}| + 5\log_n|\mathfrak{M}| + 11\log_n(|\mathfrak{M}| + 1) + \\ + 4\log_n m + 9 + 7\log_n(1/\varepsilon) \lesssim |\mathfrak{M}|.$$

Essentially, the proof of this theorem resembles that of Theorem 4.6. The difference is that more precise estimates are used at various stages of the proof. To prove Theorem 4.7, therefore, we shall need, apart from Lemmas 3 and 6 proved in the preceding section, another two theorems, whose proofs will be postponed to Chapter V.

Let $\tilde{G}_{\rho>r}$ be the set of all automata in $\tilde{G}$ with degree of distinguishability greater than r.

THEOREM 5.1 *For any automaton graph G and any natural number r,*

$$\frac{|\tilde{G}_{\rho>r}|}{|G|} < |G|^2 \left(\frac{1}{n}\right)^{r/2}.$$

As before, let $\tilde{G}_{F(l)\leqq s}$ be the set of all automata $\mathfrak{M}$ in G such that $F_{\langle\mathfrak{M},q_0\rangle}(l) \leqq s$.

THEOREM 5.3. *For any automaton graph G of height* h_G and any natural numbers s and ψ such that $s + \psi \leqq h_G$,*

$$\frac{|\tilde{G}_{F(s+\psi)\leqq s}|}{|\tilde{G}|} \leqq \frac{s(s+1)}{2} \left(\frac{1}{n}\right)^{\psi}.$$

Throughout the sequel, G will be a fixed automaton graph of height h_G all of whose vertices are accessible from q_0. It is obvious that, for any $l \geqq h_G$, the accessibility spectrum of the graph G is given by $D_{\langle G,q_0\rangle}(l) = |G|$. Our goal is to determine the smallest possible function $\phi_\varepsilon(s)$,independent of G, that satisfies the requirements of Lemma 6.

LEMMA 7. *Let $\phi(s)$ be an arbitrary control function and ψ an arbitrary natural number. Then, for any natural number s such that $\phi(s) \geqq s + \psi$ and $s + \psi \leqq h_G$,*

$$\frac{|O(\tilde{G}, \phi(s), s)|}{|\tilde{G}|} \leqq \frac{s(s+1)}{2} \left(\frac{1}{n}\right)^{\psi}.$$

Proof. Let s satisfy the inequality $\phi(s) \geqq s + \psi$.

* See p. 277.

Since $O(\tilde{G}, \phi(s), s) \subseteq O(\tilde{G}, l(s), s)$ for $\phi(s) \geqq l(s)$ and $O(\tilde{G}, l(s), s) \subseteq \subseteq \tilde{G}_{F(l(s)) \leqq s}$, it follows that

$$\frac{|O(\tilde{G}, \phi(s), s)|}{|\tilde{G}|} \leqq \frac{|O(\tilde{G}, s+\psi, s)|}{|\tilde{G}|} \leqq \frac{|\tilde{G}_{F(s+\psi) \leqq s}|}{|\tilde{G}|}.$$

Applying Theorem 5.3 to $|\tilde{G}_{F(s+\psi) \leqq s}| / |\tilde{G}|$, we get the required inequality.

LEMMA 8. *Let $\phi(s)$ be an arbitrary control function. Then, for any natural number s such that $\phi(s) > h_G$,*

$$\frac{|O(\tilde{G}, \phi(s), s)|}{|\tilde{G}|} \leqq \frac{(ns)^{ms}}{n^{|G|-1}}.$$

Proof. Let s satisfy the inequality $\phi(s) > h_G$. Then $D_{\langle G, q_0 \rangle}(\phi(s)) = |G|$ and by Lemma 3

$$\frac{|\tilde{G}_{F(\phi(s)) \leqq s}|}{|\tilde{G}|} \leqq \frac{(ns)^{ms}}{n^{|G|-1}}.$$

Since $O(\tilde{G}, \phi(s), s) \subseteq G_{F(\phi(s)) \leqq s}$, this proves the lemma.

LEMMA 9. *Let $\phi(s)$ be an arbitrary control function and r an arbitrary natural number. Then, for any natural number s such that $\phi(s) \geqq h_G + r + 1$,*

$$\frac{|O(\tilde{G}, \phi(s), s)|}{|\tilde{G}|} \leqq |G|^2 \left(\frac{1}{n}\right)^{r/2}.$$

Proof. Let $\mathfrak{M}$ be an automaton with degree of accessibility $\delta(\mathfrak{M}, q_0)$ and degree of distinguishability $\rho(\mathfrak{M})$. Consider the total tree of height $\delta(\mathfrak{M}, q_0) + \rho(\mathfrak{M}) + 1$ compatible with $\mathfrak{M}$. It follows from Theorem 2.16′ that the automaton derived from this tree by contraction realizes the same operator as $\mathfrak{M}$. This means that the algorithm Σ_ϕ, applied to $\mathfrak{M}$, cannot admit an s-error if $\phi(s) \geqq \delta(\mathfrak{M}, q_0) + \rho(\mathfrak{M}) + 1$.

Now assume that s satisfies the inequality $\phi(s) \geqq h_G + r + 1$. The degree of accessibility of the automata obtained from the graph G is h_G. By the foregoing arguments, therefore, the algorithm Σ_ϕ cannot admit an s-error for any automaton of $\tilde{G}$ whose degree of distinguishability is at most r. In other words, if $\mathfrak{M} \notin \tilde{G}_{\rho > r}$, then $\mathfrak{M} \notin O(\tilde{G}, \phi(s), s)$. Consequently, $O(\tilde{G}, \phi(s), s) \subseteq \tilde{G}_{\rho > r}$. Thus,

$$\frac{|O(\tilde{G}, \phi(s), s)|}{|\tilde{G}|} \leqq \frac{|\tilde{G}_{\rho > r}|}{|\tilde{G}|}.$$

Lemma 9 now follows from this inequality and the inequality

$$\frac{|\tilde{G}_{\rho > r}|}{|\tilde{G}|} < |G|^2 \left(\frac{1}{n}\right)^{r/2}$$

(Theorem 5.1).

Now suppose that the natural number s also satisfies the inequality

$$\frac{(ns)^{ms}}{n^{|G|-1}} > \frac{\varepsilon}{(s+1)^2}.$$

Then $|G|$ obviously satisfies the inequality

$$|G| < ms \log_n ns + 2 \log_n (s+1) + 1 + \log_n (1/\varepsilon).$$

Since s is a natural number (and so $s \geqq 1$), $m \geqq 2$, and $n \geqq 2$, we see that $|G|$ satisfies the inequality

$$|G| < mns(s+1) + \log_n (1/\varepsilon).$$

Thus, Lemma 9 implies

COROLARY. *Let $\phi(s)$ be an arbitrary control function and r an arbitrary natural number. Then, for any natural number s such that*

$$\phi(s) \geqq h_G + r + 1$$

and

$$\frac{(ns)^{ms}}{n^{|G|-1}} > \frac{\varepsilon}{(s+1)^2},$$

we have

$$\frac{|O(\tilde{G}, \phi(s), s)|}{|\tilde{G}|} < (mns(s+1) + \log(1/\varepsilon))^2 (1/n)^{r/2}.$$

LEMMA 10. *Let $\psi_\varepsilon(s)$ and $r_\varepsilon(s)$ be computable nondecreasing arithmetical functions such that for any natural number s*

a)
$$\frac{s(s+1)}{2} \left(\frac{1}{n}\right)^{\psi_\varepsilon(s)} \leqq \frac{\varepsilon}{(s+1)^2},$$

b)
$$(mns(s+1) + \log_n (1/\varepsilon))^2 \left(\frac{1}{n}\right)^{r_\varepsilon(s)/2} \leqq \frac{\varepsilon}{(s+1)^2}.$$

Then the algorithm $\Sigma_{\phi_\varepsilon}$ with control function $\phi_\varepsilon(s) = s + \psi_\varepsilon(s) + r_\varepsilon(s) + 1$ identifies absolute black boxes uniformly with frequency $1 - \varepsilon$.

Proof. Let ψ_ε and r_ε satisfy the assumptions of the lemma and $\phi_\varepsilon(s) = s + \psi_\varepsilon(s) + r_\varepsilon(s) + 1$. Consider an arbitrary automaton graph G all of whose vertices are accessible from q_0. By Lemma 6, our lemma will be proved if we can show that, for any natural number s,

$$\frac{|O(\tilde{G}, \phi_\varepsilon(s), s)|}{|\tilde{G}|} \leqq \frac{\varepsilon}{(s+1)^2}.$$

Thus, let s be an arbitrary fixed natural number. We shall distinguish three cases (it is easily seen that they exhaust all possible situations).

1) $s + \psi_\varepsilon(s) \leqq h_G$; then Lemma 7 applies, and we get

$$\frac{|O(\tilde{G}, \phi_\varepsilon(s), s)|}{|\tilde{G}|} \leqq \frac{s(s+1)}{2}\left(\frac{1}{n}\right)^{\psi_\varepsilon(s)} \leqq \frac{\varepsilon}{(s+1)^2}.$$

2) $s + \psi_\varepsilon(s) > h_G$ and $(ns)^{ms}/n^{|G|-1} \leqq \varepsilon/(s+1)^2$; then Lemma 8 applies, and we get

$$\frac{|O(\tilde{G}, \phi_\varepsilon(s), s)|}{|\tilde{G}|} \leqq \frac{\varepsilon}{(s+1)^2}.$$

3) $s + \psi_\varepsilon(s) > h_G$ and $(ns)^{ms}/n^{|G|-1} > \varepsilon/(s+1)^2$; then

$$\phi_\varepsilon(s) = s + \psi_\varepsilon(s) + r_\varepsilon(s) + 1 > h_G + r_\varepsilon(s) + 1$$

and so the Corollary to Lemma 9 applies; the result is

$$\frac{|O(\tilde{G}, \phi_\varepsilon(s), s)|}{|\tilde{G}|} < \left(mns(s+1) + \log_n \frac{1}{\varepsilon}\right)^2 \left(\frac{1}{n}\right)^{r_\varepsilon(s)/2} \leqq \frac{\varepsilon}{(s+1)^2}.$$

We have thus verified that in all cases

$$\frac{|O(\tilde{G}, \phi_\varepsilon(s), s)|}{|\tilde{G}|} \leqq \frac{\varepsilon}{(s+1)^2}.$$

The lemma is proved.

It is now easy to prove Theorem 4.7. To this end, isolate ψ_ε and r_ε from inequalities a) and b) of Lemma 10. We obtain

$$\psi_\varepsilon(s) \geqq \log_n s + 3\log_n(s+1) + \log_n(1/\varepsilon) - \log_n 2,$$

$$r_\varepsilon(s) \geqq 4\log_n\left\{mns(s+1) + \log_n(1/\varepsilon)\right\} + 4\log_n(s+1) + 2\log_n(1/\varepsilon).$$

Since $\varepsilon \leqq 1$, $m \geqq 2$ and $n \geqq 2$,

$$mns(s+1) + \log_n(1/\varepsilon) \leqq mns(s+1)(1/\varepsilon).$$

Therefore,

$$4\log_n\left\{(mns(s+1)+\log_n(1/\varepsilon)\right\}+4\log_n(s+1)+2\log_n(1/\varepsilon)\leqq$$
$$\leqq 4\log_n(mns(s+1)(1/\varepsilon))+4\log_n(s+1)+2\log_n(1/\varepsilon)=$$
$$=4\log_n s+8\log_n(s+1)+4\log_n m+4+6\log_n(1/\varepsilon).$$

As functions ψ_ε and r_ε satisfying inequalities a) and b) of Lemma 10, take

$$\psi_\varepsilon(s)=\left]\log_n s+3\log_n(s+1)\right[+\left]\log_n(1/\varepsilon)\right[,$$
$$r_\varepsilon(s)=\left]4\log_n s+8\log_n(s+1)+4\log_n m+4\right[+\left]6\log_n(1/\varepsilon)\right[.$$

Now define

$$\phi_\varepsilon(s)=s+\psi_\varepsilon(s)+r_\varepsilon(s)+1=$$
$$=s+\left]\log_n s+3\log_n(s+1)\right[+\left]\log_n(1/\varepsilon)\right[+$$
$$+\left]4\log_n s+8\log_n(s+1)+4\log_n m+4\right[+\left]6\log_n(1/\varepsilon)\right[+1.$$

By Lemma 10, the algorithm $\Sigma_{\phi_\varepsilon}$ will identify absolute black boxes uniformly with frequency $1-\varepsilon$. The signaling function of this algorithm satisfies the inequalities

$$\Sigma^*_{\phi_\varepsilon}(\mathfrak{M})\leqq\phi_\varepsilon(|\mathfrak{M}|)<|\mathfrak{M}|+5\log_n|\mathfrak{M}|+$$
$$+11\log_n(|\mathfrak{M}|+1)+4\log_n m+9+7\log_n(1/\varepsilon)\lesssim|\mathfrak{M}|.$$

This completes the proof of the theorem.

IV.8. Bound on the complexity of (nonuniform) identification. Statement of the fundamental results

Recall that an algorithm Σ is said to identify absolute black boxes (nonuniformly) with frequency $1-\varepsilon$ if, for any natural k,

$$\frac{|\mathscr{L}^\Sigma(k)|}{|\mathscr{L}(k)|}\geqq 1-\varepsilon,$$

where $\mathscr{L}(k)$ is the set of all pairwise nonidentical automata with k states and $\mathscr{L}^\Sigma(k)$ the subset of all those which are identified (as absolute black boxes) by the algorithm Σ.

The question arises as to whether the bound on the signaling function established in Theorem 4.7 can be essentially improved if one drops the uniformity requirement, i.e., if one simply considers identification rather

than uniform identification The following fundamental theorem shows that this is indeed the case.

THEOREM 4.8. *For any $\varepsilon > 0$ there exists a multiple iterative algorithm $\Sigma_{\phi_\varepsilon}$ which initially identifies absolute black boxes with frequency $1 - \varepsilon$ and whose signaling function satisfies the inequalities*

$$\Sigma^*_{\phi_\varepsilon}(\mathfrak{M}) \leqq C \log_n |\mathfrak{M}| + C_\varepsilon \leqq \log |\mathfrak{M}|,$$

where C is a constant independent of $\mathfrak{M}$ and ε, and C_ε is a constant independent of $\mathfrak{M}$ but dependent on ε.

The proof of this theorem reduces to that of three other theorems, which we now state without proof.

THEOREM 5.7. *There exist positive constants C_1 and C_2 such that almost all automata $\mathfrak{M}$ with k states possess the following property: if q_i is any state of the automaton and $l \leqq C_1 \log_m k$, the accessibility spectrum $D_{\langle \mathfrak{M}, q_i \rangle}(l)$ satisfies the inequality*

$$D_{\langle \mathfrak{M}, q_i \rangle}(l) \geqq m^{C_2 l}.$$

THEOREM 5.10. *There exists a constant C_0 such that almost all automata $\mathfrak{M}$ with k states have absolute degree of reconstructibility $B^*(\mathfrak{M})$ at most $C_0 \log_m k$.*

These theorems will be proved in Chapter V.

Let us call an automaton graph G a (C_1, C_2)-*graph* if its accessibility spectrum satisfies the inequality

$$D_{\langle G, q_0 \rangle}(l) \geqq m^{C_2 l}$$

for $l \leqq C_1 \log_m |G|$. An automaton $\mathfrak{M}$ will be called a (C_1, C_2, C_3)-*automaton* if it is derived from a (C_1, C_2)-graph and its degree of reconstructibility satisfies the inequality

$$B(\mathfrak{M}, q_0) \leqq C_3 \log_m |\mathfrak{M}|.$$

Obviously, Theorems 5.7 and 5.10 imply the following corollary as a particular case:

COROLLARY. *There exist positive constants C_1, C_2, C_3 such that almost all automata are (C_1, C_2, C_3)-automata.*

We shall say that an algorithm Σ *identifies* (C_1, C_2, C_3)-*automata uni-*

formly with frequency $1 - \varepsilon$ if, for any automaton graph G,

$$\frac{|\tilde{G}_{C_1,C_2,C_3,\Sigma}|}{|\tilde{G}|} < \varepsilon,$$

where $\tilde{G}_{C_1,C_2,C_3,\Sigma}$ is the set of all (C_1,C_2,C_3)-automata in $\tilde{G}$ which the algorithm Σ does not initially identify as absolute black boxes (clearly, if G is not a (C_1,C_2)-graph the set $\tilde{G}_{C_1,C_2,C_3,\Sigma}$ is empty, since then none of the automata in the set $\tilde{G}$ is a (C_1,C_2,C_3)-automaton).

THEOREM 4.8′. *Let* C_1,C_2,C_3 *be arbitrary fixed positive constants. Then there exist constants* C,B,A *such that for any* $\varepsilon > 0$ *the multiple iterative algorithm* $\Sigma_{\phi_\varepsilon}$ *with control function* $\phi_\varepsilon(s) =]C \log_n s[+]B \log_n(1/\varepsilon) + + A[$ *identifies* (C_1,C_2,C_3)*-automata uniformly with frequency* $1 - \varepsilon$.

The proof of this theorem is postponed to Section IV.9. Here we shall show that Theorems 5.7 and 5.10, as cited above, and Theorem 4.8′ imply Theorem 4.8.

We first observe that if the algorithm Σ identifies (C_1,C_2,C_3)-automata uniformly with frequency $1 - \varepsilon$, so that (by definition) for any automaton graph G,

$$\frac{|\tilde{G}_{C_1,C_2,C_3,\Sigma}|}{|\tilde{G}|} < \varepsilon,$$

then we also have the stronger assertion: for any natural number k,

$$\frac{|\mathscr{L}_{C_1,C_2,C_3,\Sigma}(k)|}{|\mathscr{L}(k)|} < \varepsilon,$$

where, as usual, $\mathscr{L}(k)$ is the set of all pairwise nonidentical automata with k states and $\mathscr{L}_{C_1,C_2,C_3,\Sigma}(k)$ the set of all (C_1,C_2,C_3)-automata in $\mathscr{L}(k)$ which are not identified by the algorithm Σ.

Let C_1^0,C_2^0,C_3^0 be constants (whose existence is ensured by the Corollary to Theorems 5.7 and 5.10) such that almost all automata are (C_1^0,C_2^0,C_3^0)-automata. Then, obviously there exists K_ε such that for all $k \geqq K_\varepsilon$

$$\frac{|\mathscr{L}_{C_1^0,C_2^0,C_3^0}(k)|}{|\mathscr{L}(k)|} \geqq 1 - \varepsilon,$$

where $\mathscr{L}_{C_1^0,C_2^0,C_3^0}(k)$ is the set of all pairwise nonidentical (C_1^0,C_2^0,C_3^0)-automata with k states.

The last two inequalities imply that if Σ is an algorithm with control

function

$$\phi_\varepsilon(s) =]C \log_n s[+]B \log_n(1/\varepsilon) + A[$$

which identifies (C_1^0, C_2^0, C_3^0)-automata with frequency $1 - \varepsilon$, then it also identifies arbitrary automata (not only (C_1^0, C_2^0, C_3^0)-automata) with at least K_ε states, with frequency $1 - 2\varepsilon$; in other words, when $k \geqq K_\varepsilon$,

$$\frac{|\mathscr{L}^\Sigma(k)|}{|\mathscr{L}(k)|} \geqq 1 - 2\varepsilon.$$

Now consider the iterative algorithm Σ' with control function

$$\phi'_\varepsilon(s) = \phi_\varepsilon(s) + 2K_\varepsilon =]C \log_n s[+]B \log_n(1/\varepsilon) + A[+ 2K_\varepsilon.$$

This algorithm will obviously identify any automaton with at most K_ε states (since as early as Step 1 it examines a tree of height at least $2K_\varepsilon$, from which an operator realized by an automaton with K_ε states can be unambiguously constructed). This means that the algorithm Σ' satisfies the inequality

$$\frac{|\mathscr{L}^{\Sigma'}(k)|}{|\mathscr{L}(k)|} \geqq 1 - 2\varepsilon$$

for *all* natural numbers k. It follows that the iterative algorithm Σ'' with control function

$$\phi''_\varepsilon(s) = \phi'_{\varepsilon/2}(s) =$$

$$=]C \log_n s[+]B \log_n(2/\varepsilon) + A[+ 2K_{\varepsilon/2} =]C \log_n s + C'_\varepsilon[$$

will identify absolute black boxes with frequency $1 - \varepsilon$. The signaling function of this algorithm satisfies the inequality

$$\Sigma''^*(\mathfrak{M}) \leqq C \log_n |\mathfrak{M}| + C_\varepsilon,$$

where C is a constant independent of $\mathfrak{M}$ and ε, and C_ε a constant which depends on ε but not on $\mathfrak{M}$.

Thus Theorem 4.8 follows from Theorems 5.7, 5.10 and 4.8′.

Is the bound on the signaling function given by Theorem 4.8 the best possible? We shall show that it is, up to order of magnitude. In other words *the upper bound on the signaling function* $\Sigma^*(\mathfrak{M})$ *of any (not necessarily iterative) algorithm* Σ *which identifies absolute black boxes with frequency* $1 - \varepsilon$ *(where* $\varepsilon < 1$*) has order of magnitude at least* $\log |\mathfrak{M}|$; more precisely *it is asymptotically bounded below by* $\log_m |\mathfrak{M}|$.

Suppose that this is not true. Then for some $\varepsilon < 1$ there exists an algorithm Σ which identifies absolute black boxes with frequency $1 - \varepsilon$ and for any C possesses the following property:

B. There exists an infinite increasing sequence $k_1, k_2, \ldots, k_i \ldots$ of natural numbers such that for any black box $\mathfrak{M}$ with $|\mathfrak{M}| = k_i$ the signaling function satisfies the inequality $\Sigma^*(\mathfrak{M}) \leqq \log_m |\mathfrak{M}| - C$.

Theorem 5.6, which will be proved in Chapter V, states that almost all automaton graphs with k vertices have height greater than $\log_m k - 2$. Let G_{k_i} be any such graph with k_i vertices. Consider the set M of edges of the graph G_{k_i} which are accessible from q_0 only by input words of length at most $\log_m k_i - C$. The cardinality of M is at least

$$m((\log_m k_i - 2) - (\log_m k_i - C)) = m(C - 2)$$

(since the height of G_{k_i} is greater than $\log_m k_i - 2$). Therefore, the edges may be assigned output labels in $n^{m(C-2)}$ different ways, all of which essentially affect the operators realized by the resulting automata. On the other hand, since the algorithm Σ has property B, the outcome when it is applied to an automaton obtained from the graph G_{k_i} is independent of the output labels assigned to the edges in M. Therefore,

$$\frac{|\tilde{G}_{k_i}^{\Sigma}|}{|\tilde{G}_{k_i}|} \leqq \frac{1}{n^{m(C-2)}}.$$

Hence and from Theorem 5.6 it follows that there exists i_0 such that for all $i \geqq i_0$

$$\frac{|\mathscr{L}^{\Sigma}(k_i)|}{|\mathscr{L}(k_i)|} \leqq \frac{2}{n^{m(C-2)}}.$$

Now, given any $\varepsilon < 1$ one can clearly choose a constant C_ε such that when $C = C_\varepsilon$ the right-hand side of this inequality is smaller than $1 - \varepsilon$. This means that if the algorithm Σ has property B for $C = C_\varepsilon$, it cannot identify absolute black boxes with frequency $1 - \varepsilon$. This contradiction proves our assertion concerning the bound on the signaling function.

IV.9. Proof of Theorem 4.8′

Let $O_{C_1,C_2,C_3}(\tilde{G}, \phi(s), s)$ denote the set of all (C_1, C_2, C_3)-automata in $\tilde{G}$ on which the algorithm Σ_ϕ admits an s-error. It is obvious that $O_{C_1,C_2,C_3}(\tilde{G}, \phi(s), s) \subseteq O(\tilde{G}, \phi(s), s)$. It is also easily seen that when $\phi(s_0) \geqq \geqq C_3 \log_m |G|$ the set $O_{C_1,C_2,C_3}(\tilde{G}, \phi(s_0), s_0)$ is empty.

LEMMA 11. *Let the control function $\phi(s)$ be such that for any (C_1, C_2)-graph G and any natural number s*

$$\frac{|O_{C_1,C_2,C_3}(\tilde{G}, \phi(s), s)|}{|\tilde{G}|} \leqq \frac{\varepsilon}{(s+1)^2}.$$

Then the algorithm Σ_ϕ identifies (C_1, C_2, C_3)-automata uniformly with frequency $1 - \varepsilon$.

We omit the proof, since it is analogous to that of Lemma 6.

LEMMA 12. *Let $\phi(s)$ be an arbitrary control function, G an arbitrary (C_1, C_2)-graph and ψ a natural number such that $\psi \leqq C_1 \log_m |G|$. Then for any natural number s such that $\phi(s) \geqq \psi$,*

$$\frac{|O_{C_1,C_2,C_3}(\tilde{G}, \phi(s), s)|}{|\tilde{G}|} \leqq \frac{(ns)^{ms}}{n^{m^{C_2\psi}-1}}.$$

Proof. Let G be an arbitrary (C_1, C_2)-graph, $\psi \leqq C_1 \log_m |G|$, and assume that s is such that $\phi(s) \geqq \psi$. Then

$$D_{\langle G, q_0\rangle}(\phi(s)) \geqq D_{\langle G, q_0\rangle}(\psi) \geqq m^{C_2\psi}$$

and it follows from Lemma 3 that

$$\frac{|\tilde{G}_{F(\phi(s)) \leqq s}|}{|\tilde{G}|} \leqq \frac{(ns)^{ms}}{n^{m^{C_2\psi}-1}}.$$

Since $O_{C_1,C_2,C_3}(\tilde{G}, \phi(s), s) \subseteq O(\tilde{G}, \phi(s), s)$, this implies the lemma.

LEMMA 13. *Let $\psi(s)$ be a computable nondecreasing arithmetical function such that for any natural number s*

$$\frac{(ns)^{ms}}{n^{m^{C_2\psi(s)}-1}} \leqq \frac{\varepsilon}{(s+1)^2}.$$

Then the algorithm $\Sigma_{\phi_\varepsilon}$ with test function

$$\phi_\varepsilon(s) =]\max(1, C_3/C_1)[\psi(s)$$

identifies (C_1, C_2, C_3)-automata uniformly with frequency $1 - \varepsilon$.

Proof. Let $\psi(s)$ be a function satisfying the assumptions of the lemma and let $\phi_\varepsilon(s) =]\max(1, C_3/C_1)[\psi(s)$. Consider an arbitrary (C_1, C_2)-graph G. Let s be an arbitrary fixed natural number. We shall distinguish two cases:

1) $\psi(s) \leqq C_1 \log_m |G|$; then, by Lemma 12,

$$\frac{|O_{C_1,C_2,C_3}(\tilde{G}, \phi(s), s)|}{|\tilde{G}|} \leqq \frac{ns^{ms}}{n^{m^{C_2\psi(s)}-1}} \leqq \frac{\varepsilon}{(s+1)^2}.$$

2) $\psi(s) > C_1 \log_m |G|$; then

$$\phi_\varepsilon(s) =]\max(1, C_3/C_1)[\ \psi(s) > C_3 \log_m |G|$$

and so $O_{C_1,C_2,C_3}(G, \phi_\varepsilon(s), s)$ is empty; therefore,

$$\frac{|O_{C_1,C_2,C_3}(\tilde{G}, \phi_\varepsilon(s), s)|}{|\tilde{G}|} = 0 < \frac{\varepsilon}{(s+1)^2}.$$

Thus, for any natural number s,

$$\frac{|O_{C_1,C_2,C_3}(\tilde{G}, \phi_\varepsilon(s), s)|}{|\tilde{G}|} \leqq \frac{\varepsilon}{(s+1)^2}.$$

Together with Lemma 11, this implies Lemma 13.

To prove Theorem 4.8′, we shall now try to find a computable nondecreasing arithmetical function $\psi(s)$ satisfying the assumptions of Lemma 13. One possible choice for this function is*

$$\psi(s) =]1/C_2 \log_n m[\cdot(]\log_n(ms \log_n ns) + 2\log_n(s+1) + 1[+]\log_n(1/\varepsilon)[).$$

By Lemma 13, the algorithm Σ with control function

$$\phi'_\varepsilon(s) =]\max(1, C_3/C_1)[\cdot]1/C_2 \log_n m[\cdot(]\log_n(ms \log_n ns) + 2\log_n(s+1) + 1[+]\log_n(1/\varepsilon)[)$$

will identify (C_1, C_2, C_3)-automata uniformly with frequency $1 - \varepsilon$. Since m and n are constants greater than 1, it is clear that one can choose constants C, B and A independent of ε such that for all natural numbers s the control function $\phi'_\varepsilon(s)$ satisfies the inequality

$$\phi'_\varepsilon(s) \leqq C \log_n s + B \log_n(1/\varepsilon) + A.$$

Hence it follows that the algorithm $\Sigma_{\phi_\varepsilon}$ with control function

$$\phi_\varepsilon(s) =]C \log_n s[+]B \log_n(1/\varepsilon) + A[$$

will identify (C_1, C_2, C_3)-automata uniformly with frequency $1 - \varepsilon$. This completes the proof.

* As a function of s, this function is computable even if the constant C_2 is not a constructive number (in view of the special position of the symbols]and[in the expression for the function).

IV.10. Identification of absolute black boxes by simple algorithms with arbitrary preassigned frequency

In this section we shall show that simple algorithms, like multiple algorithms, are also capable of identifying absolute black boxes with any preassigned frequency (though here we are dealing with residual identification). We shall prove an even stronger proposition (Theorem 4.9), showing that this identification (which is moreover uniform!) may be performed with simple iterative algorithms.

THEOREM 4.9. *For any $\varepsilon > 0$, there exists a simple iterative algorithm Π_{g_ε} which residually identifies absolute black boxes uniformly with frequency $1 - \varepsilon$ and whose signaling function satisfies the inequality*

$$\Pi^*_{g_\varepsilon}(\mathfrak{M}) < C_\varepsilon m^{C|\mathfrak{M}|\log|\mathfrak{M}|}$$

(C is a constant independent of $\mathfrak{M}$ and ε, C_ε is a constant independent of $\mathfrak{M}$ but not of ε).

To prove this theorem, we shall need the spectra of accessibility and saturation, viewed as functions not of a natural argument but rather of input words x. We now define these spectra.

The *word spectrum of accessibility* of an automaton (automaton graph) $\mathfrak{M}$ is the function $\bar{D}_{\langle \mathfrak{M}, q_0 \rangle}(x)$ defined as the number of states of the automaton (graph) $\mathfrak{M}$ accessible from q_0 by initial segments of the word x (counting the state q_0 itself, which is accessible by the empty segment, and the state $q_0 x$, which is accessible by the word x).

Now, for every input word x, consider the tree $V_{(x)}$ compatible with the automaton $\mathfrak{M}$, which is determined by the word x (this tree clearly consists of a single branch). The *word spectrum of saturation* of the automaton $\mathfrak{M}$ is the function $\bar{F}_{\langle \mathfrak{M}, q_0 \rangle}(x)$ defined as the weight of the tree $V_{(x)}$, or, equivalently, the number of states of a minimal automaton which generates the same output as $\mathfrak{M}$ for the input word x. As before, $\tilde{G}_{\bar{F}(x) \leqq s}$ will denote the subset of $\tilde{G}$ consisting of all automata $\mathfrak{M}$ for which $\bar{F}_{\langle \mathfrak{M}, q_0 \rangle}(x) \leqq s$.

It is not difficult to see that Lemma 3 (Section IV.6) remains valid if the spectra $D_{\langle G, q_0 \rangle}(l)$ and $F_{\langle \mathfrak{M}, q_0 \rangle}(l)$ are replaced by the word spectra $\bar{D}_{\langle G, q_0 \rangle}(x)$ and $\bar{F}_{\langle \mathfrak{M}, q_0 \rangle}(x)$. We state this as an independent lemma.

LEMMA 14. *Let G be an arbitrary automaton graph with word spectrum of accessibility $\bar{D}(x)$. Then for any input word x and any natural number s,*

$$\frac{|\tilde{G}_{\bar{F}(x) \leqq s}|}{|\tilde{G}|} \leqq \frac{(ns)^{ms}}{n^{\bar{D}(x)-1}}.$$

We shall say that an *input word x identifies an automaton* $\mathfrak{M}$ if any minimal automaton $\mathfrak{A}$ which is indistinguishable from $\mathfrak{M}$ by the word x realizes the same operator after application of x as does the automaton $\mathfrak{M}$ after application of x, i.e., $T(\mathfrak{A}, q_0x) = T(\mathfrak{M}, q_0x)$.

In Section IV.2 we introduced the concept of residual distinguishability. There we proved (Corollary to Lemma 2) that for any natural number k one can effectively construct a word $d(k)$ of length $]4k^2(\ln nk)m^{2k}[$ which residually distinguishes the set of automata with at most k states. It is easy to see that the word $d(k)$ identifies any such automaton.

We wish to find a control function $g_\varepsilon(s)$ such that

$$g_\varepsilon(s) = f(\phi_\varepsilon(s)),$$

where

$$f(k) = d(1)d(2)\dots d(k)$$

and $\phi_\varepsilon(s)$ is a computable nondecreasing arithmetical function.

It is easily seen that any function $g_\varepsilon(s)$ with this property is the control function of a simple iterative algorithm. In fact, it is computable, its values are words over the input alphabet, and $g_\varepsilon(s)$ is an initial segment of $g_\varepsilon(s+1)$ (since $f(k)$ is an initial segment of $f(k+1)$ and $\phi_\varepsilon(s)$ is a nondecreasing function).

We now claim that the word $f(k) = d(1)d(2)\dots d(k)$ identifies all automata having at most k states. To prove this it suffices to show that this word residually distinguishes any two automata $\mathfrak{M}_1$ and $\mathfrak{M}_2$ which have at most k states. Consider the automata

$$\langle \mathfrak{M}_1, q_0d(1)\dots d(k-1)\rangle \text{ and } \langle \mathfrak{M}_2, q_0d(1)\dots d(k-1)\rangle.$$

These automata also have at most k states, and so the word $d(k)$ residually distinguishes them. This means that if

$$T(\mathfrak{M}_1, q_0d(1)\dots d(k-1)d(k)) \neq T(\mathfrak{M}_2, q_0d(1)\dots d(k-1)d(k)),$$

then the automata

$$\langle \mathfrak{M}_1, q_0d(1)\dots d(k-1)\rangle \text{ and } \langle \mathfrak{M}_2, q_0d(1)\dots d(k-1)\rangle$$

will generate different output words in response to the input word $d(k)$. But if this is so, the automata $\mathfrak{M}_1$ and $\mathfrak{M}_2$ will also generate different output words in response to the input word $d(1)\dots d(k)$. Hence the input word $d(1)\dots d(k)$ residually distinguishes the automata $\mathfrak{M}_1$ and $\mathfrak{M}_2$.

Let G be an arbitrary automaton graph. Let $A_G(x)$ ($A_{\mathfrak{M}}(x)$) denote the

set of all vertices of the graph G (the automaton $\mathfrak{M}$) which are accessible from the vertex q_0 by initial segments of the word x or from the vertex q_0x by arbitrary words. For example, if G is the graph illustrated in Figure 43 and $x = x_1x_2$, then $A_G(x) = (q_0, q_2, q_5, q_4, q_6, q_7)$. It is easy to see that in order to decide whether a word x identifies an automaton it is sufficient to consider only that part of $\mathfrak{M}$ corresponding to $A_{\mathfrak{M}}(x)$. This statement is formulated rigorously in the following lemma.

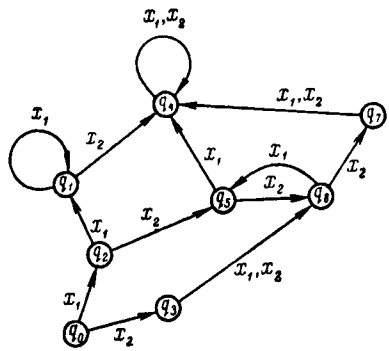


Figure 43

LEMMA 15. *If the word x identifies all automata with at most k states and $|A_G(x)| \leqq k$, then the word x identifies all automata in $\tilde{G}$.*

Proof. Let $\mathfrak{M}$ be an arbitrary automaton in $\tilde{G}$. Consider the automaton $\mathfrak{M}'$ obtained from $\mathfrak{M}$ by deleting all vertices outside $A_G(x)$ and joining all edges issuing from $A_G(x)$ and incident on deleted vertices to some fixed vertex of $A_G(x)$. Obviously, $\mathfrak{M}$ and $\mathfrak{M}'$ are indistinguishable by the word x, and $T(\mathfrak{M}, q_0x) = T(\mathfrak{M}', q_0x)$. Thus, if the word x identifies $\mathfrak{M}'$ it will also identify $\mathfrak{M}$. Now, if x satisfies the assumptions of the lemma, then $|\mathfrak{M}'| = |A_G(x)| \leqq k$, and so the word x identifies $\mathfrak{M}'$. This proves the lemma.

Since the word $f(k) = d(1) \ldots d(k)$ identifies all automata with at most k states, Lemma 15 implies the following

COROLLARY. *If $|A_G(f(k))| \leqq k$, the word $f(k)$ identifies all automata in $\tilde{G}$.*

LEMMA 16. *If the word x identifies all automata having at most k states and $|A_G(x)| \geqq k$, the word spectrum of accessibility of the graph G satisfies the inequality $\bar{D}_{\langle G,q_0\rangle}(x) \geqq k$.*

Proof. Let $B_G(x)$ denote the set of all vertices of the graph G accessible from q_0 by initial segments of the word x. Assume that the statement of

the lemma is false, i.e., $\bar{D}_{\langle G, q_0\rangle}(x) < k$. In other words, $|B_G(x)| < k$, and so $|A_G(x)| > |B_G(x)|$. Consider the automaton graph G' obtained from G by deleting all vertices outside $B_G(x)$ and joining edges issuing from $B_G(x)$ and incident on deleted vertices to the initial vertex q_0 (where $q_0 \in B_G(x)$). In view of the inequality $|A_G(x)| > |B_G(x)|$, it is easily seen that the graph G' is strongly connected. Since by assumption the word x identifies all automata with at most k states, it will also identify all automata in $\tilde{G}'$. But if this is so, then clearly all edges of G', including those issuing from $B_G(x)$ in the original graph G and previously incident upon vertices outside $B_G(x)$, must be accessible from q_0 by initial segments of x. Therefore, again in view of the inequality $|A_G(x)| > |B_G(x)|$, the graph G must contain vertices, accessible from q_0 by initial segments of x, which are not included in $B_G(x)$. This contradiction proves the lemma.

Lemma 16 directly implies the following:

COROLLARY. *If* $|A_G(f(x))| \geqq k$, *then*

$$\bar{D}_{\langle G, q_0\rangle}(f(k)) \geqq k.$$

The rest of the proof of Theorem 4.9 proceeds in analogy to that of Theorem 4.6. Let Π_g be a simple iterative algorithm with control function $g(s)$. We shall say that Π_g, applied to an automaton $\mathfrak{M}$, *admits an s-error* ($s = 1,2,3,\ldots$) if there exists an automaton $\mathfrak{A}$ with s states which is indistinguishable from $\mathfrak{M}$ by the word $g(s)$ (i.e., $\bar{F}_{\langle \mathfrak{M}, q_0\rangle}(g(s)) \leqq s$), and

$$T(\mathfrak{A}, q_0 g(s)) \neq T(\mathfrak{M}, q_0 g(s)).$$

LEMMA 17. *If an algorithm* Π_g *applied to an automaton* $\mathfrak{M}$ *admits no s-error for any natural number s, then* Π_g *identifies* $\mathfrak{M}$.

The proof, which is analogous to that of Lemma 5, is omitted.

Let $Q(\tilde{G}, g(s), s)$ denote the set of all automata in $\tilde{G}$ for which the algorithm Π_g admits an s-error. Obviously,

$$Q(\tilde{G}, g(s), s) \subseteq \tilde{G}_{\bar{F}(g(s)) \leqq s}.$$

LEMMA 18. *Let control function* $g(s)$ *be such that for any automaton graph G and any natural number s*

$$\frac{|Q(\tilde{G}, g(s), s)|}{|\tilde{G}|} \leqq \frac{\varepsilon}{(s+1)^2}.$$

Then the algorithm Π_g identifies absolute black boxes uniformly with frequency $1 - \varepsilon$.

We omit the proof; it is analogous to that of Lemma 6, except insofar as Lemma 17 is used instead of Lemma 5.

It is now easy to prove Theorem 4.9. Let $\varepsilon > 0$ be an arbitrary fixed number. It follows from the Corollary to Lemma 15 (replacing k by $\phi_\varepsilon(s)$) that for all s such that $\phi_\varepsilon(s) > |A_G(f(\phi_\varepsilon(s)))|$ the word $f(\phi_\varepsilon(s))$ identifies all automata in $\tilde{G}$ and so, for these values of s,

$$\frac{|Q(\tilde{G}, f(\phi_\varepsilon(s)), s)|}{|\tilde{G}|} = 0 < \frac{\varepsilon}{(s+1)^2}.$$

Together with Lemma 18, this implies that the algorithm Π_g with control function $g_\varepsilon(s) = f(\phi_\varepsilon(s))$ will identify absolute black boxes uniformly with frequency $1 - \varepsilon$, provided that

$$\frac{|Q(\tilde{G}, f(\phi_\varepsilon(s)), s)|}{|\tilde{G}|} \leqq \frac{\varepsilon}{(s+1)^2}$$

for any automaton graph G and any natural number s such that $\phi_\varepsilon(s) \leqq \leqq |A_G(f(\phi_\varepsilon(s)))|$. Thus, to prove the theorem we must find a computable nondecreasing arithmetical function $\phi_\varepsilon(s)$ with the indicated property.

Since $Q(\tilde{G}, g(s), s) \subseteq \tilde{G}_{\bar{F}(g(s)) \leqq s}$, we have

$$\frac{|Q(\tilde{G}, f(\phi_\varepsilon(s)), s)|}{|\tilde{G}|} \leqq \frac{|\tilde{G}_{\bar{F}(f(\phi_\varepsilon(s))) \leqq s}|}{|\tilde{G}|}.$$

By Lemma 14,

$$\frac{|\tilde{G}_{\bar{F}(f(\phi_\varepsilon(s))) \leqq s}|}{|\tilde{G}|} \leqq \frac{(ns)^{ms}}{n^{\bar{D}_{\langle G, q_0 \rangle}(f(\phi_\varepsilon(s))) - 1}}.$$

Let $\phi_\varepsilon(s) \leqq |A_G(f(\phi_\varepsilon(s)))|$. Then, by the Corrollary to Lemma 16 $D_{\langle G, q_0 \rangle}(f(\phi_\varepsilon(s))) \geqq \phi_\varepsilon(s)$, and so

$$\frac{(ns)^{ms}}{n^{\bar{D}_{\langle G, q_0 \rangle}(f(\phi_\varepsilon(s))) - 1}} \leqq \frac{(ns)^{ms}}{n^{\phi_\varepsilon(s) - 1}}.$$

It follows that the computable nondecreasing arithmetical function $\phi_\varepsilon(s)$ will possess the required property if the right-hand side of the last inequality does not exceed $\varepsilon/(s+1)^2$. Therefore, one possible choice for $\phi_\varepsilon(s)$ is

$$\phi_\varepsilon(s) =]ms \log_n ns + 2 \log_n(s+1) + 1[+] \log_n(1/\varepsilon)[.$$

Thus, the simple iterative algorithm Π_{g_ε} with control function

$$g_\varepsilon(s) = f(\phi_\varepsilon(s)) = f(]ms \log_n ns + 2\log_n(s+1) + 1[+]\log_n(1/\varepsilon)[)$$

will identify absolute black boxes uniformly with frequency $1 - \varepsilon$.

We shall now estimate the signaling function of the algorithm Π_{g_ε}. By Section IV.5,

$$\Pi^*_{g_\varepsilon}(\mathfrak{M}) \leqq l(g_\varepsilon(|\mathfrak{M}|)).$$

Therefore, in the present case,

$$\Pi^*_{g_\varepsilon}(\mathfrak{M}) \leqq l(f(\phi_\varepsilon(|\mathfrak{M}|))).$$

It remains to estimate $l(f(k))$.

Since

$$f(k) = d(1)\dots d(k) \quad \text{and} \quad l(d_i) =]4i^2(\ln ni)m^{2i}[,$$

it follows that

$$l(f(k)) \leqq C'k^2(\ln nk)m^{2k},$$

where C' is a constant independent of k.

Substitute

$$\phi_\varepsilon(s) =]ms \log_n ns + 2\log_n(s+1) + 1[+]\log_n(1/\varepsilon)[$$

for k. The result is

$$l(f(\phi_\varepsilon(s))) \leqq C_\varepsilon m^{Cs \log s},$$

where C is a constant independent of s and ε, and C_ε a constant independent of s but not of ε. Thus

$$\Pi^*_{g_\varepsilon}(\mathfrak{M}) \leqq C_\varepsilon m^{C|\mathfrak{M}|\log|\mathfrak{M}|}.$$

This completes the proof of Theorem 4.9

IV.11. Bounds on the complexity of (nonuniform) identification by simple algorithms

It was proved in the preceding section that there exists a simple algorithm which identifies absolute black boxes uniformly with frequency $1 - \varepsilon$, and an upper bound for the signaling function was given. This bound turns out to be quite exaggerated, and one may well ask whether it can be significantly improved. In this section we intend to show that if uniform identi-

fication is replaced by identification (nonuniform), this question can be answered in the affirmative.

We first state our fundamental theorem on the identification of absolute black boxes by simple algorithms.

THEOREM 4.10. *For any $\varepsilon > 0$, there is a simple algorithm Ω which residually identifies absolute black boxes with frequency $1 - \varepsilon$ and whose signaling function satisfies the inequality*

$$\Omega^*(\mathfrak{M}) \leqq C_\varepsilon \,|\, \mathfrak{M} \,|^C$$

(C is a constant independent of $\mathfrak{M}$ and ε, C_ε a constant independent of $\mathfrak{M}$ but not of ε).

To prove this theorem, we shall consider modifications of simple iterative algorithms—we call them (C_1, C_2, C_3)-algorithms—and show that attention may be confined to algorithms of this type.

Before proceeding to a definition of these algorithms, a few auxiliary concepts must be defined. Call an automaton graph G an *absolute (C_1, C_2)-graph* if, for any of its vertices q_i and any natural number $l \leqq C_1 \log_m |\, G \,|$, the accessibility spectrum satisfies the inequality $D_{\langle G, q_i \rangle}(l) \geqq m^{C_2 l}$. An automaton $\mathfrak{M}$ will be called an *absolute (C_1, C_2, C_3)-automaton* if it is derived from a (C_1, C_2)-graph and its absolute degree of reconstructibility satisfies the inequality $B^*(\mathfrak{M}) \leqq C_3 \log_m |\, \mathfrak{M} \,|$. Call a constant C *strongly effective* if its decimal expansion is finite (the number of digits after the decimal point is finite).

Theorems 5.7 and 5.10 imply the following

COROLLARY. *There exist strongly effective positive constants C_1, C_2, C_3 such that almost all automata are absolute (C_1, C_2, C_3)-automata.*

The requirement that the constants C_1, C_2, C_3 be strongly effective reflects the fact that in the sequel we shall define an algorithm using these constants (an algorithm may employ only constructive objects!).

Let C_1, C_2, C_3 be strongly effective constants. A *(C_1, C_2, C_3)-algorithm* is defined by two functions $g(s)$ and $\chi(s)$; the function $g(s)$, which we call the *word control function*, satisfies the same conditions as the control function of a simple iterative algorithm; $\chi(s)$, which we call the *boundary function*, is a computable nondecreasing arithmetical function. A (C_1, C_2, C_3)-algorithm $\Omega_{g,\chi}$ with word control function $g(s)$ and boundary function $\chi(s)$ works in exactly the same way as a simple iterative algorithm Π_g, except for the following point. In an iterative algorithm Π_g, if the hypoth-

esis $\mathfrak{A}_{i-1}$ is indistinguishable from $\mathfrak{M}$ by the word $g(|\mathfrak{A}_{i-1}|)$, the algorithm halts and its outcome is the automaton $\langle \mathfrak{A}_{i-1}, q_0 g(|\mathfrak{A}_{i-1}|)\rangle$. In the algorithm $\Omega_{g,\chi}$ if the hypothesis $\mathfrak{A}_{i-1}$ is indistinguishable from $\mathfrak{M}$ by the word $g(|\mathfrak{A}_{i-1}|)$, the algorithm also halts, but its outcome is defined differently: among all automata with at most $\chi(|\mathfrak{A}_{i-1}|)$ states, look for an absolute (C_1, C_2, C_3)-automaton which is indistinguishable from $\mathfrak{M}$ by the word $g(|\mathfrak{A}_{i-1}|)$ and has minimal number of states; if such an automaton exists (denote it by $\mathfrak{B}$), the outcome of the algorithm is the automaton $\langle \mathfrak{B}, q_0 g(|\mathfrak{A}_{i-1}|)\rangle$; if there is no such automaton, then, as before, the outcome is the automaton $\langle \mathfrak{A}_{i-1}, q_0 g(|\mathfrak{A}_{i-1}|)\rangle$.

It is clear that the signaling function of a (C_1, C_2, C_3)-algorithm $\Omega_{g,\chi}$ coincides with that of the simple iterative algorithm Π_g.

Consequently

$$\Omega^*_{g,\chi}(\mathfrak{M}) \leqq l'(g(s)).$$

As in the case of Theorem 4.8, the proof of Theorem 4.10 reduces to that of another theorem which deals exclusively with (C_1, C_2, C_3)-automata.

We shall say that an algorithm Ω *identifies absolute* (C_1, C_2, C_3)*-automata uniformly with frequency* $1 - \varepsilon$ if, for any automaton graph G,

$$\frac{|\tilde{G}_{C_1, C_2, C_3, \Omega}|}{|\tilde{G}|} < \varepsilon,$$

where $\tilde{G}_{C_1, C_2, C_3, \Omega}$ is the set of all absolute (C_1, C_2, C_3)-automata in $\tilde{G}$ which are not residually identified by the algorithm Ω as absolute black boxes.

THEOREM 4.10. *Let* C_1, C_2, C_3 *be arbitrary fixed strongly effective positive constants. Then for any* $\varepsilon > 0$ *there exist a word control function* $g_\varepsilon(s)$ *and a boundary function* $\chi_\varepsilon(s)$ *such that*

1) *for any input word* d *(possibly empty), the* (C_1, C_2, C_3)*-algorithm* $\Omega_{dg_\varepsilon, \chi_\varepsilon}$ *with word control function* $dg_\varepsilon(s)$ *and boundary function* $\chi_\varepsilon(s)$ *identifies absolute* $(C_1, C_2, C_3$*-automata uniformly with frequency* $1 - \varepsilon$;

2) *for any natural number* s

$$l(g_\varepsilon(s)) \leqq C_\varepsilon s^C,$$

where C *is a constant independent of* s *and* ε, C_ε *a constant independent of* s *but not of* ε.

We shall first show how Theorem 4.10 follows from the above Corollary to Theorems 5.7 and 5.10 and from Theorem 4.10′. Determine constants

C_1^0, C_2^0, C_3^0 according to the above-mentioned Corollary such that almost all automata are absolute (C_1^0, C_2^0, C_3^0)-automata. Let $g_\varepsilon^0(s)$ and $\chi_\varepsilon^0(s)$ be functions that satisfy the assertion of Theorem 4.10′ for $C_1 = C_1^0$, $C_2 = C_2^0$, $C_3 = C_3^0$. Denote the (C_1^0, C_2^0, C_3^0)-algorithm $\Omega_{dg_\varepsilon^0, \chi_\varepsilon^0}$ by $\Omega_{(d)}$. Reasoning just as in Section IV.8, it is not hard to verify that there exists K_ε such that, for all $k \geqq K_\varepsilon$,

$$\frac{|\mathscr{L}^{\Omega_{(d)}}(k)|}{|\mathscr{L}(k)|} \geqq 1 - 2\varepsilon.$$

Let d_ε be a word which residually identifies all automata with at most $\max(K_\varepsilon, \chi_\varepsilon^0(K_\varepsilon))$ states. Obviously, the (C_1^0, C_2^0, C_3^0)-algorithm $\Omega_{(d_\varepsilon)}$ with word control function $d_\varepsilon g_\varepsilon^0(s)$ and boundary function $\chi_\varepsilon^0(s)$ will residually identify any automata having at most K_ε^0 states. Therefore, the algorithm $\Omega_{(d_\varepsilon)}$ satisfies the inequality

$$\frac{|\mathscr{L}^{\Omega_{(d_\varepsilon)}}(k)|}{|\mathscr{L}(k)|} \geqq 1 - 2\varepsilon$$

for *all* natural numbers k. This means that the algorithm $\Omega_{(d_\varepsilon)}$ residually identifies absolute black boxes with frequency $1 - 2\varepsilon$. We now estimate the signaling function of this algorithm. Now, by Theorem 4.10′,

$$l(g_\varepsilon^0(s)) \leqq C_\varepsilon s^C.$$

Therefore,

$$\Omega^*_{(d_\varepsilon)}(\mathfrak{M}) \leqq l(d_\varepsilon g_\varepsilon^0(|\mathfrak{M}|)) = l(d_\varepsilon) + l(g_\varepsilon^0(|\mathfrak{M}|)) \leqq$$
$$\leqq l(d_\varepsilon) + C_\varepsilon |\mathfrak{M}|^C \leqq C'_\varepsilon |\mathfrak{M}|^C,$$

where C is a constant independent of $\mathfrak{M}$ and ε, C_ε a constant independent of $\mathfrak{M}$ but not of ε.

Theorem 4.10 now follows if we replace ε by $\varepsilon/2$ and denote $C'_{\varepsilon/2}$ by C_ε.

In order to prove Theorem 4.10′, we need several more lemmas.

LEMMA 19. *Let s and k be arbitrary natural numbers. Then one can effectively construct an input word $b_s(k)$ of length $s]mkm^s \ln 2k[$ with the following property: for any automaton graph G, if $D_{\langle G, q_i\rangle}(s) \geqq k$ for any vertex q_i of G, then $\bar{D}_{\langle G, q_i\rangle}(b_s(k)) \geqq k$ for any vertex q_i of G.*

Proof. It will clearly suffice to show that one can effectively determine an input word $b_s(k)$ of length $s]2mkm^s \ln 2k[$ with the following property: for any automaton graph G such that $D_{\langle G, q_i\rangle}(l) \geqq k$ $(i = 0, 1, 2, \ldots)$ and

any $k-1$ designated vertices (including q_0) of G, one of the nondesignated vertices is always accessible from q_0 by the word $b_s(k)$ (by vertices accessible by a word b we mean all vertices lying on the path defined by b). Associate the graph G including $k-1$ designated vertices with an auxiliary graph G', obtained from G by deleting all nondesignated vertices and joining all edges issuing from designated vertices and incident on nondesignated vertices (call these forbidden edges) to an arbitrary designated vertex. It is obvious that a nondesignated vertex is accessible from q_0 by a word x in the graph G' only if some forbidden edge is accessible from q_0 in G by the same word. Since $D_{\langle G,q_i\rangle}(s) \geqq k$ for any vertex q_i of G, it follows that, for any vertex q_i of G', there exists an input word of length at most s by which at least one forbidden edge is accessible from q_i in G'. Reasoning in exactly the same way as for Lemma 1, it is easily seen that the number of input words of length ls by which no forbidden edges are accessible from q_0 in the graph G' is at most $(m^s-1)^l$. On the other hand, since $|G'| < k$, the number of different graphs of type G' (different as regards the choice of forbidden edges and disregarding the indexing of vertices other than q_0) is less than $k^{mk}2^{mk}$, where k^{mk} is the number of nonidentical automaton graphs with k vertices and 2^{mk} the number of ways in which one can choose forbidden edges in a graph G' with k vertices. It follows that the number of different input words of length ls by which no forbidden edge is accessible from q_0 in at least one graph of type G' (or, equivalently, by which no nondesignated vertex is accessible from q_0 in at least one graph G of the above type with $k-1$ fixed designated vertices) is at most $k^{mk}2^{mk}(m^s-1)^l$. Reasoning again as in Lemma 1, we see that if $ls = s]mkm^s \ln 2k[$ there must be at least one input word not possessing the above property. This word is clearly effectively constructible, and it is the required word $b_s(k)$. This proves the lemma.

Let C_1, C_2, C_3 be arbitrary fixed strongly effective positive constants.

LEMMA 20. *For any natural number k, one can effectively construct an input word $b(k)$ of length at most*

$$]({1}/{C_2})\log_m k[\cdot]m^2 k^{1+1/C_2}\ln 2k[$$

which has the following property: for any absolute (C_1, C_2)-graph G, if

$$k \leqq ({1}/{m^{C_2}})\,|G|^{C_1C_2},$$

then

$$\bar{D}_{\langle G,q_i\rangle}(b(k)) \geqq k$$

(q_i is an arbitrary vertex of the graph G).

Proof. Let G be an absolute (C_1, C_2)-graph and $k \leqq (1/m^{C_2}) \mid G \mid^{C_1 C_2}$. Then $]\,(1/C_2)\log_m k[< C_1 \log_m \mid G \mid$. Since G is an absolute (C_1, C_2)-graph, whenever $l \leqq C_1 \log_m \mid G \mid$ we have $D_{\langle G, q_i \rangle}(l) \geqq m^{C_2 l}$. It follows that

$$D_{\langle G, q_i \rangle}(\,]\,(1/C_2)\log_m k[\,) \geqq m^{C_2](1/C_2)\log_m k[} \geqq k.$$

Now apply Lemma 19 with $s =]\,(1/C_2)\log_m k[$—the result is the required word $b(k)$. This proves the lemma.

LEMMA 21. *For any natural number k, one can effectively construct an input word $d'(k)$ of length at most $(C_3 \log_m k)\,]2mk^{1+C_3} \ln nk[$ which residually identifies all absolute (C_1, C_2, C_3)-automata having at most k states.*

This lemma follows directly from Lemma 2 and the fact that absolute (C_1, C_2, C_3)-automata with at most k states have absolute degree of reconstructibility $B^*(\mathfrak{M})$ not exceeding $C_3 \log_m k$.

Set

$$f(k) = b(1)d'(1)b(2)d'(2)\ldots b(k)d'(k).$$

It is obvious that the function $f(k)$ satisfies the conditions imposed on word control functions, i.e., it is computable, defined on natural numbers, its values are input words, and $f(k)$ is an initial segment of $f(k+1)$.

LEMMA 22. *Let d be an arbitrary input word. Then*

1) *for any absolute (C_1, C_2)-graph G and any natural number*

$$k \leqq (1/m^{C_2}) \mid G \mid^{C_1 C_2}$$

we have the inequality

$$\bar{D}_{\langle G, q_i \rangle}(df(k)) \geqq k;$$

2) *the input word $df(k)$ residually identifies all absolute (C_1, C_2, C_3)-automata with at most k states.*

The truth of the first assertion of this lemma is a direct consequence of Lemma 20 and the definition of $f(k)$. The second assertion follows from Lemma 21, the definition of $f(k)$, and the fact that by varying the initial state of an absolute (C_1, C_2, C_3)-automaton one obtains another (C_1, C_2, C_3)-automaton.

Let $\Omega_{g,\chi}$ be a (C_1, C_2, C_3)-algorithm which, when applied to an automaton $\mathfrak{M}$, gives rise to the following situation: there exists an automaton $\mathfrak{A}$ with s states, which, if taken as the hypothesis generated at Step $i-1$, makes the algorithm $\Omega_{g,\chi}$ halt at Step i and produce an incorrect outcome, i.e., an

automaton which realizes an operator different from $T(\mathfrak{M}, q_0 g(s))$. We shall then say that $\Omega_{g,\chi}$ applied to $\mathfrak{M}$ *admits an s-error.*

Let $Q^{\chi}_{C_1,C_2,C_3}(G, g(s), s)$ denote the set of all absolute (C_1, C_2, C_3)-automata in $\tilde{G}$ on which the (C_1, C_2, C_3)-algorithm $\Omega_{g,\chi}$ admits an s-error. Obviously,

$$Q^{\chi}_{C_1,C_2,C_3}(\tilde{G}, g(s), s) \subseteq \tilde{G}_{\bar{F}(g(s)) \leqq s}.$$

It is also easy to prove the following analogue of Lemma 18:

LEMMA 23. *Let the control function $g(s)$ and boundary function $\chi(s)$ be such that for any absolute (C_1, C_2)-graph G and any natural number s*

$$\frac{|Q^{\chi}_{C_1,C_2,C_3}(\tilde{G}, g(s), s)|}{|\tilde{G}|} \leqq \frac{\varepsilon}{(s+1)^2}.$$

Then the (C_1, C_2, C_3)-algorithm $\Omega_{g,\chi}$ identifies absolute (C_1, C_2, C_3)-automata uniformly with frequency $1 - \varepsilon$.

LEMMA 24. *Let d be an arbitrary input word, $\phi(s)$ a computable nondecreasing arithmetical function, G an absolute (C_1, C_2)-graph and k a natural number such that*

$$k \leqq (1/m^{C_2}) |G^{C_1 C_2}|.$$

Then for any natural number s such that $\phi(s) \geqq k$,

$$\frac{|Q^{\chi}_{C_1,C_2,C_3}(\tilde{G}, df(\phi(s)), s)|}{|\tilde{G}|} \leqq \frac{(ns)^{ms}}{n^{k-1}}.$$

Proof. Under the assumptions of the lemma, it follows from the definition of the function $f(k)$ that if $\phi(s) \geqq k$, then $\bar{D}_{\langle G, q_i\rangle}(df(\phi(s))) \geqq \bar{D}_{\langle G, q_i\rangle}(df(k))$. Hence, by Lemma 22,

$$\bar{D}_{\langle G, q_0\rangle}(df(\phi(s))) \geqq k,$$

and so, by Lemma 14,

$$\frac{|\tilde{G}_{\bar{F}(df(\phi(s)) \leqq s}|}{|\tilde{G}|} \leqq \frac{(ns)^{ms}}{n^{k-1}}.$$

Since

$$Q^{\chi}_{C_1,C_2,C_3}(\tilde{G}, df(\phi(s)), s) \subseteq \tilde{G}_{\bar{F}(df(\phi(s))) \leqq s},$$

this implies the statement of our lemma.

LEMMA 25. *Let $\psi_\varepsilon(s)$ be a computable nondecreasing arithmetical function such that for any natural number s*

$$\frac{(ns)^{ms}}{n^{\psi_\varepsilon(s)-1}} \leqq \frac{\varepsilon}{(s+1)^2}.$$

Let

$$\chi_\varepsilon(s) = \max\{\psi_\varepsilon(s),](m^{C_2}\psi_\varepsilon(s))^{1/C_1C_2}[\}$$

and

$$g_\varepsilon(s) = f(\chi_\varepsilon(s)).$$

Then for any input word d the (C_1, C_2, C_3)-algorithm $\Omega_{dg_\varepsilon,\chi_\varepsilon}$ identifies absolute (C_1, C_2, C_3)-automata uniformly with frequency $1-\varepsilon$.

Proof. Let $\psi_\varepsilon(s)$, $\chi_\varepsilon(s)$, and $g_\varepsilon(s)$ satisfy the assumptions of the lemma. Consider an arbitrary absolute (C_1, C_2)-graph G, and let s be an arbitrary fixed number. We shall distinguish two cases:

1) $\psi_\varepsilon(s) \leqq (1/m^{C_2})\,|\,G\,|^{C_1C_2}$; then, by Lemma 24,

$$\frac{|\,Q^{\chi_\varepsilon}_{C_1,C_2,C_3}(\tilde{G}, dg_\varepsilon(s), s)\,|}{|\,\tilde{G}\,|} \leqq \frac{(ns)^{ms}}{n^{\psi_\varepsilon(s)-1}} \leqq \frac{\varepsilon}{(s+1)^2},$$

2) $\psi_\varepsilon(s) > (1/m^{C_2})\,|\,G\,|^{C_1C_2}$; then

$$\chi_\varepsilon(s) = \max\{\psi_\varepsilon(s),](m^{C_2}\psi_\varepsilon(s))^{1/C_1C_2}[\} > |\,G\,|,$$

and the second assertion of Lemma 22 implies that the word $dg_\varepsilon(s) = df(\chi_\varepsilon(s))$ residually distinguishes all absolute (C_1, C_2, C_3)-automata with at most $|\,G\,|$ states. Thus any (C_1, C_2, C_3)-automaton $\mathfrak{M}$ derived from the graph G has the following property: first, among the automata indistinguishable from $\mathfrak{M}$ by the word $dg_\varepsilon(s)$ there exists an absolute (C_1, C_2, C_3)-automaton $\mathfrak{B}$ with at most $|\,G\,|$ states, therefore at most $\chi_\varepsilon(s)$ states; second, the automaton $\langle \mathfrak{B}, q_0 dg_\varepsilon(s)\rangle$ realizes the same operator as $\langle \mathfrak{M}, q_0 dg_\varepsilon(s)\rangle$ (since the word $dg_\varepsilon(s) = df(\chi_\varepsilon(s))$ residually identifies all absolute (C_1, C_2, C_3)-automata having at most $|\,G\,|$ states). But this means precisely that the algorithm $\Omega_{dg_\varepsilon,\chi_\varepsilon}$ applied to $\mathfrak{M}$ does not admit s-errors. Hence, in this case the set $Q^{\chi_\varepsilon}_{C_1,C_2,C_3}(\tilde{G}, dg_\varepsilon(s), s)$ is empty.

We have thus shown that for any natural number s

$$\frac{|\,Q^{\chi_\varepsilon}_{C_1,C_2,C_3}(\tilde{G}\, dg_\varepsilon(s), s)\,|}{|\,\tilde{G}\,|} \leqq \frac{\varepsilon}{(s+1)^2}.$$

Together with Lemma 23, this implies the statement of our lemma.

To prove Theorem 4.10′, we now need a computable nondecreasing

arithmetical function $\psi_\varepsilon(s)$ which satisfies the assumptions of Lemma 25. One possible choice for $\psi_\varepsilon(s)$ is the function

$$\psi_\varepsilon(s) =]ms \log_n ns + 2 \log_n(s+1) + 1[+]\log_n(1/\varepsilon)[.$$

Then, according to Lemma 25, the (C_1, C_2, C_3)-algorithm $\Omega_{dg_\varepsilon, \chi_\varepsilon}$, with d an arbitrary input word,

$$g_\varepsilon(s) = f(\chi_\varepsilon(s)) \quad \text{and} \quad \chi_\varepsilon(s) = \max\{\psi_\varepsilon(s),](m^{C_2}\psi_\varepsilon(s))^{1/C_1C_2}[\},$$

will identify absolute (C_1, C_2, C_3)-automata uniformly with frequency $1 - \varepsilon$. This proves the first assertion of Theorem 4.10′.

We now turn to the second assertion of Theorem 4.10′. Let us estimate $l(g_\varepsilon(s))$ for natural numbers s. Since

$$f(k) = b(1)d'(1)\ldots b(k)d'(k),$$

$$l(b(i)) \leqq](1/C_2)\log_m i[\cdot]m^2 i^{1+1/C_2} \ln 2i[,$$

$$l(d'(i)) \leqq (C_3 \log_m i)]2mi^{1+C_3} \ln ni[,$$

it is obvious that there exist constants C' and C'' such that for all natural numbers k

$$l(f(k)) \leqq C'k^{C''}.$$

Now replace k by the function

$$\chi_\varepsilon(s) = \max\{\psi_\varepsilon(s),](m^{C_2}\psi_\varepsilon(s))^{1/C_1C_2}[\},$$

where

$$\psi_\varepsilon(s) =]ms \log_n ns + 2 \log_n(s+1) + 1[+]\log_n(1/\varepsilon)[.$$

The result is

$$l(f(\chi_\varepsilon(s))) \leqq C_\varepsilon s^C,$$

where C is a constant independent of s and ε, C_ε a constant independent of s but not of ε. Since $g_\varepsilon(s) = f(\chi_\varepsilon(s))$, this implies the second assertion of our theorem, completing the proof of Theorem 4.10′.

Supplementary material, problems

I. The signaling function $\Omega^*(\mathfrak{M})$ of an algorithm was defined above as the maximal length of the input words with which the algorithm Ω tests the automaton $\mathfrak{M}$. However, for multiple algorithms a more precise indicator

of the complexity of identification would be the total length of the input words with which the algorithm Ω tests $\mathfrak{M}$ (if Ω tests $\mathfrak{M}$ with words x and xz, the total length of these words is defined to be the length of xz alone). We denote this total length by $\Omega^{**}(\mathfrak{M})$ and call it the *exact signaling function* of Ω. Obviously,

$$\Omega^{**}(\mathfrak{M}) \leqq \Omega^{*}(\mathfrak{M}) \cdot m^{\Omega^{*}(\mathfrak{M})}.$$

Thus Theorems 4.3 and 4.8 imply the following bounds for the exact signaling function:

1. *There is a multiple unconditional algorithm Ω which initially identifies almost all relative black boxes and whose exact signaling function satisfies the inequality*

$$\Omega^{**}(\mathfrak{M}) \leqq K_{\mathfrak{M}}^{C},$$

where C is a constant.

2. *There is a multiple algorithm Σ which initially identifies absolute black boxes with frequency $1 - \varepsilon$ and whose exact signaling function satisfies the inequality*

$$\Sigma^{**}(\mathfrak{M}) \leqq |\mathfrak{M}|^{C},$$

where C is a constant independent of $\mathfrak{M}$.

It is easy to see that, up to the constant C, this estimate for the exact signaling function is the best possible.

Similarly, Theorem 4.7 implies the following assertion:

There is a multiple algorithm Σ which initially identifies absolute black boxes uniformly with frequency $1 - \varepsilon$ and whose exact signaling function satisfies the inequality

$$\Sigma^{**}(\mathfrak{M}) \leqq C^{|\mathfrak{M}|},$$

where C is a constant independent of $\mathfrak{M}$.

It seems likely that this estimate can be substantially improved. It has been conjectured that there exists a multiple algorithm Ω which initially identifies absolute black boxes uniformly with frequency $1 - \varepsilon$ and whose exact signaling function satisfies the inequality $\Omega^{**}(\mathfrak{M}) \leqq |\mathfrak{M}|^{C}$, where C is a constant independent of $\mathfrak{M}$.†

† *Added in proof*: This conjecture has recently been proved by M. P. Vasilevskii. Moreover, Barzdin' has been able to prove that for any $\varepsilon > 0$ there exists a simple algorithm Ω which residually identifies absolute black boxes uniformly with frequency $1 - \varepsilon$, and such that $\Omega^{*}(\mathfrak{M}) \leqq C_{\varepsilon}|\mathfrak{M}|^{C}$ (thereby sharpening Theorem 4.9).

II. Hitherto we have considered only automata with indexed states. Our frequency computations involve the number of *nonidentical* automata, and this concept was defined with regard to automaton with indexed states. Basing our arguments on this type of computation, we defined such statements as "property E holds for almost all automata," "property E holds with frequency $1 - \varepsilon$," and so on. The set of pairwise nonidentical automata was denoted by $\mathscr{L}$. Henceforth, when referring to these concepts of "almost all," "with frequency $1 - \varepsilon$" and so on, we shall specify "almost all automata of class $\mathscr{L}$," "automata with frequency $1 - \varepsilon$ in class $\mathscr{L}$," and so on.

Obviously, two automata previously considered to be nonidentical may in fact be isomorphic, say equivalent. However, in frequency considerations of properties of automata it may prove advantageous to identify all pairwise isomorphic automata or all pairwise equivalent automata (i.e., to count such automata only once in frequency computations). We then replace $\mathscr{L}$ by the following sets (both input and output alphabets are assumed fixed):

1) $\mathscr{R}$: the set of all pairwise nonisomorphic initialized automata;*

2) $\mathscr{S}$: the set of all pairwise nonisomorphic initialized automata all of whose states are pairwise distinguishable;

3) $\mathscr{U}$: the set of all pairwise nonequivalent initialized automata, or, which is the same (see Section II.3), the set of all pairwise nonisomorphic reduced initialized automata.

Similarly, the set $\tilde{G}$ is replaced by the following sets:

1) $\tilde{G}_{\mathscr{R}}$: the set of all pairwise nonisomorphic initialized automata in $\tilde{G}$ (recall that $\tilde{G}$ consists of initialized automata, with initial state q_0);

2) $\tilde{G}_{\mathscr{S}}$: the set of all pairwise nonisomorphic initialized automata in $\tilde{G}$, all of whose states are pairwise distinguishable;

3) $\tilde{G}_{\mathscr{U}}$: the set of all pairwise nonequivalent initialized automata in $\tilde{G}$.

As usual, the notation $\mathscr{R}(k)$, $\mathscr{S}(k)$, etc., will be used for the appropriate subsets of automata having k states. The notation $\mathscr{R}^E(k)$, $\tilde{G}^E_{\mathscr{R}}$, etc., will be used for the appropriate subsets of automata possessing property E.

The frequency characteristics considered previously are also applicable to the classes $\mathscr{R}$, $\mathscr{S}$, $\mathscr{U}$. For example, we shall say that the automata of class $\mathscr{R}$ possess property E with frequency $1 - \varepsilon$ if, for any natural k,

$$\frac{|\mathscr{R}^E(k)|}{|\mathscr{R}(k)|} \geqq 1 - \varepsilon.$$

* Two initialized automata are said to be isomorphic if they are isomorphic in the usual sense (i.e., as noninitialized automata) and their initial states correspond to each other under this isomorphism.

Similarly, we shall say that the automata of class $\mathscr{R}$ possess property E uniformly with frequency $1 - \varepsilon$ if, for any automaton graph G,

$$\frac{|\tilde{G}^E_{\mathscr{R}}|}{|\tilde{G}_{\mathscr{R}}|} \geqq 1 - \varepsilon.$$

We shall now check the validity of the theorems proved above when the class $\mathscr{L}$ is replaced by any of the classes $\mathscr{R}$, $\mathscr{S}$, $\mathscr{U}$. We shall show that in certain cases they indeed remain valid (in the remaining cases the question is open).

First consider the class $\mathscr{R}$.

Let $\mathfrak{M}$ be an automaton with k states. It is easy to see that the number of pairwise nonidentical initialized automata with indexed states which can be derived from initialized automaton $\mathfrak{M}$ (without altering the initial state) by renumbering the states is at most $(k-1)!$

Thus, for each initialized automaton $\mathfrak{M} \in \mathscr{R}(k)$ there are at most $(k-1)!$ automata in $\mathscr{L}(k)$ isomorphic to $\mathfrak{M}$ (as initialized automata). Therefore

$$|\mathscr{R}(k)| \geqq \frac{|\mathscr{L}(k)|}{(k-1)!}.$$

One easily checks that this inequality can be generalized in the following sense: Let E be some property depending only on the operator realized by the automaton; then

$$|\mathscr{R}^E(k)| \geqq \frac{|\mathscr{L}^E(k)|}{(k-1)!}.$$

Let $\overline{\mathscr{R}}$ be the set of all pairwise nonisomorphic noninitialized automata. It was proved in [36] that, for $m \geqq 3$ and $n \geqq 2$,*

$$|\overline{\mathscr{R}}(k)| \sim \frac{(nk)^{mk}}{k!} = \frac{|\mathscr{L}(k)|}{k!}.$$

Since any noninitialized automaton with k states corresponds to at most k pairwise nonisomorphic initialized automata, it follows that

$$|\mathscr{R}(k)| \leqq k\,|\overline{\mathscr{R}}(k)|,$$

and so the above asymptotic equality implies the asymptotic inequality

$$|\mathscr{R}(k)| \lesssim \frac{|\mathscr{L}(k)|}{(k-1)!}.$$

* As before, m and n denote the cardinalities of the input and output alphabets, respectively.

Comparison of these lower and upper bounds on $|\mathscr{R}(k)|$ shows that

$$|\mathscr{R}(k)| \sim \frac{\mathscr{L}(k)}{(k-1)!}$$

asymptotically. Hence, using the inequality

$$|\mathscr{R}^E(k)| \gtrsim \frac{|\mathscr{L}^E(k)|}{(k-1)!}$$

we immediately derive the following important property:

A. *Let* $m \geqq 3$ *and* $n \geqq 2$. *If*

$$\frac{|\mathscr{L}^E(k)|}{|\mathscr{L}(k)|} \gtrsim 1 - \varepsilon,$$

then

$$\frac{|\mathscr{R}^E(k)|}{|\mathscr{R}(k)|} \gtrsim 1 - \varepsilon.$$

Consider the case $m \geqq 3$ and $n \geqq 2$. Using property A, it is not hard to show that the validity of Theorems 4.3, 4.4, 4.8, 4.10 for class $\mathscr{L}$ implies their validity for class $\mathscr{R}$ as well.

To illustrate, consider Theorem 4.8. It states that there exists a multiple iterative algorithm Σ which initially identifies automata with frequency $1 - \varepsilon$, i.e., for any natural number k,

$$\frac{|\mathscr{L}^\Sigma(k)|}{|\mathscr{L}(k)|} \geqq 1 - \varepsilon.$$

Moreover, as can be seen from the proof of Theorem 4.8, the control function $\phi_\varepsilon(s)$ of this algorithm satisfies the inequality

$$\phi_\varepsilon(s) \leqq C \log_n s + C_\varepsilon.$$

Using property A, we see that the class $\mathscr{R}$ satisfies the asymptotic inequality

$$\frac{|\mathscr{R}^\Sigma(k)|}{|\mathscr{R}(k)|} \gtrsim 1 - \varepsilon.$$

This means, in particular, that there exists a number K_ε such that for all $k \geqq K_\varepsilon$ we have the (exact, not asymptotic) inequality

$$\frac{|\mathscr{R}^\Sigma(k)|}{|\mathscr{R}(k)|} \geqq 1 - 2\varepsilon.$$

Consider the algorithm Σ with control function

$$\phi'_\varepsilon(s) = \phi_{\varepsilon/2}(s) + 2K_{\varepsilon/2} \leqq C \log_n s + C_{\varepsilon/2} + 2K_{\varepsilon/2} = C \log_n s + C'_\varepsilon.$$

Now it is obvious that for the algorithm Σ'

$$\frac{|\mathscr{R}^{\Sigma'}(k)|}{|\mathscr{R}(k)|} \geqq 1 - \varepsilon$$

for *all* natural numbers k. Hence follows Theorem 4.8 for class $\mathscr{R}$.

An analogous proof shows that Theorem 4.10 remains valid when the class $\mathscr{L}$ is replaced by $\mathscr{R}$ (except that here one must use the following additional property of the control function $g_\varepsilon(s)$ of Theorem 4.10′: Absolute (C_1, C_2, C_3)-automata are identified uniformly with frequency $1 - \varepsilon$ not only by the algorithm $\Omega_{\vartheta_\varepsilon, \chi_\varepsilon}$ but also by $\Omega_{d\vartheta_\varepsilon, \chi_\varepsilon}$, where d is an arbitrary input word).

The problem of whether Theorems 4.3, 4.4, 4.8 and 4.10 hold for the class Ω when $m = 2$ is unsolved.*

Now consider uniform identification. Let E be some property which depends only on the operator realized by the automaton question. It is not difficult to see that

$$\frac{|\tilde{G}^E|}{|\tilde{G}|} = \frac{|\tilde{G}^E_{\mathscr{R}}|}{|\tilde{G}_{\mathscr{R}}|}.$$

This equality and the fact that Theorems 4.7 and 4.9 are valid for $\mathscr{L}$ directly imply that these theorems are also valid for class $\mathscr{R}$.

Now consider the class $\mathscr{S}$. Let $\mathscr{S}_{\mathscr{L}}$ ($\bar{\mathscr{S}}$) denote the set of all pairwise nonidentical initialized automata (pairwise nonisomorphic noninitialized automata) all of whose states are pairwise distinguishable. It is not difficult to verify that for every initialized automaton $\mathfrak{M} \in \mathscr{S}(k)$ there are exactly $(k-1)!$ automata in $\mathscr{S}_{\mathscr{L}}(k)$ isomorphic to $\mathfrak{M}$ (as initialized automata). Similarly, for every noninitialized automaton $\mathfrak{M} \in \bar{\mathscr{S}}(k)$ there are exactly $k!$ automata in $\mathscr{S}_{\mathscr{L}}(k)$ isomorphic to $\mathfrak{M}$ (as noninitialized automata). Therefore, in particular,

$$\frac{|\mathscr{S}^E(k)|}{|\mathscr{S}(k)|} = \frac{|\mathscr{S}^E_{\mathscr{L}}(k)|}{|\mathscr{S}_{\mathscr{L}}(k)|} \quad \text{and} \quad |\mathscr{S}_{\mathscr{L}}(k)| = k!\,|\bar{\mathscr{S}}(k)|.$$

* *Added in proof.* A positive solution has recently been given by Korshunov, who has shown that Property A also holds when $m = 2$.

Korshunov [35] has proved that

a) if $m \geqq 3$ and $n \geqq 2$, then

$$|\bar{\mathscr{S}}(k)| \sim \frac{(nk)^{mk}}{k!} = \frac{|\mathscr{L}(k)|}{k!};$$

b) if $m = 2$ and $n \geqq 2$, then

$$|\bar{\mathscr{S}}(k)| \sim e^{-1/2n^2} \frac{(nk)^{mk}}{k!} > \frac{1}{2} \frac{|\mathscr{L}(k)|}{k!}.$$

Together with the preceding remarks, this implies that for any $m \geqq 2, n \geqq 2$,

$$|\mathscr{S}_{\mathscr{L}}(k)| \gtrsim \tfrac{1}{2}|\mathscr{L}(k)|$$

and so

$$\frac{|\mathscr{S}^E(k)|}{|\mathscr{S}(k)|} = \frac{|\mathscr{S}^E_{\mathscr{L}}(k)|}{|\mathscr{S}_{\mathscr{L}}(k)|} < 2 \frac{|\mathscr{L}^E(k)|}{|\mathscr{L}(k)|}.$$

Thus, if

$$\frac{|\mathscr{L}^E(k)|}{|\mathscr{L}(k)|} < \varepsilon,$$

then

$$\frac{|\mathscr{S}^E(k)|}{|\mathscr{S}(k)|} < 2\varepsilon.$$

Hence, replacing property E by the complementary property E' (which holds if and only if E does not), we get the following proposition:

B. *If*

$$\frac{|\mathscr{L}^{E'}(k)|}{|\mathscr{L}(k)|} \gtrsim 1 - \varepsilon,$$

then

$$\frac{|\mathscr{S}^{E'}(k)|}{|\mathscr{S}(k)|} \gtrsim 1 - 2\varepsilon.$$

Using Proposition B and arguments analogous to those employed for the class $\mathscr{R}$, it is easily seen that the validity of Theorems 4.3, 4.4, 4.8, 4.10 for class $\mathscr{L}$ implies their validity for class $\mathscr{S}$ as well.

The question of whether Theorems 4.6, 4.7 and 4.9 are valid for class $\mathscr{S}$ is still open.

Also open is the question as to whether Theorems 4.3, 4.4, 4.6, 4.7, 4.8, 4.9 and 4.10 are valid for the class $\mathscr{U}$.

III. Another question arising in this context is the following: Can one construct a partition $\{\mathscr{L}_\lambda\}$ of the class $\mathscr{L}$ into finite subclasses such that the analogue of Theorem 4.6 for the concept of frequency based on this partition is no longer true (i.e., one cannot identify absolute black boxes with arbitrary preassigned frequency)? It is not hard to show that this is possible; an appropriate partition is the following:

$$\{\mathscr{L}_\lambda\}: \quad \mathscr{L}_\lambda = \mathscr{A}_\lambda \cup \mathscr{B}_\lambda,$$

where $\mathscr{A}_\lambda$ is the set of all automata with λ states which do not belong to $\mathscr{L}_1 \cup \ldots \cup \mathscr{L}_{\lambda-1}$; $\mathscr{B}_\lambda$ is the set consisting of $\lambda(|\mathscr{A}_\lambda| + 1)$ nonequivalent automata not belonging to $\mathscr{L}_1 \cup \ldots \cup \mathscr{L}_{\lambda-1}$ which are not distinguishable by input words of length λ from the initial hypothesis $\mathfrak{A}_0$ of Figure 35.

It is obvious that for any algorithm Ω there is a number λ_Ω such that for all $\lambda \geqq \lambda_\Omega$ we have $|\mathscr{B}_\lambda^\Omega| \leqq 1$, and, consequently,

$$\frac{|\mathscr{L}_\lambda^\Omega|}{|\mathscr{L}_\lambda|} \leqq \frac{|\mathscr{A}_\lambda| + 1}{\lambda(|\mathscr{A}_\lambda| + 1)} = \frac{1}{\lambda}$$

($\mathscr{B}_\lambda^\Omega$ and $\mathscr{L}_\lambda^\Omega$ are the sets of automata which the algorithm Ω identifies as absolute black boxes). This means that for the above partition $\{\mathscr{L}_\lambda\}$ there is an algorithm which identifies absolute black boxes with frequency $1 - \varepsilon$ for $\varepsilon < 1$.

IV. The so-called terminal-state identification problem has been investigated in the literature in fairly great detail. Given a reduced automaton whose diagram is known, but not its initial state (one might call this automaton a partial black box), it is required to construct a simple experiment by which one can determine the state of the automaton at the end of the experiment (though the initial state may remain unknown). In our terminology, we can reformulate this problem as follows: to construct a simple algorithm over partial black boxes which residually identifies any of the latter. The basic problem here is to construct an experiment of minimal length. It has been studied in [108], [92], [32], [98] and other papers. The strongest estimate in this respect is due to Hibbard [98]. He has shown that for any noninitialized reduced automaton with k states there exists a uniform experiment of length $k(k-1)/2$ by which one can determine the state of the automaton at the end of the experiment (for any choice of initial state). In our terminology: There exists a simple unconditional algorithm Ω which residually identifies any partial black box $\mathfrak{M}$ and has a signaling function such that $\Omega^*(\mathfrak{M}) \leqq \frac{1}{2}|\mathfrak{M}|(|\mathfrak{M}| - 1)$. Hibbard has also shown that, as a

function of the number of states k, this estimate cannot be improved (even by considering nonuniform experiments).

Can Hibbard's estimate be substantially improved if one considers not *all* reduced automata but *almost all* reduced automata? Korshunov [38] has shown that here one need consider only uniform experiments of length at most $5 \log_n k$.

Notes

The contents of this chapter are closely linked up with the theory of experiments, whose foundations were laid by Moore [108]. Some results in this field were obtained indepently of Moore by Trakhtenbrot [56]. Further improvements of the estimates may be found in Ginsburg [92], Karatsuba [32], Hibbard [98], Borodyanskii [22], Muchnik [49], and others.

The problem of identification is considered here within the framework of the general theory of synthesis. This approach may already be found in [7] and [55].

In essence, Theorem 4.1 may be found in [108] and [56]. The paper [108] also contains a weak version of Theorem 4.2. It differs from our version in that it contains a coarser estimate for the length of the appropriate experiment (in our terminology: a coarser estimate for the signaling function of the simple algorithm which identifies relative black boxes). In the form given in Theorem 4.2, the estimate was first established by Muchnik [49] (though by a different method). Theorem 4.3 is in effect implicit in Korshunov [37]. Theorem 4.4 apparently makes its first appearance in this book.

The above-mentioned papers and theorems deal with relative black boxes. A systematic investigation of the identification problem for absolute black boxes apparently appears in this book for the first time. Theorems 4.5. 4.6, 4.7 and 4.8 are due to Barzdin', Theorems 4.9 and 4.10 to Barzdin' and Vasilevskii

CHAPTER V

STATISTICAL ESTIMATES FOR PARAMETERS AND SPECTRA OF AUTOMATA

In this chapter we shall establish the statistical estimates of parameters and spectra used in Chapter IV. Our theorems will indicate the frequency of occurrence of automata properties, the latter being expressed in terms of inequalities for various parameters and spectra. In accordance with the layout of Section IV.3, this will be done separately for uniform frequency computations (fixed automaton graph) and for the more general case (fixed number of states). In Sections V.1 and V.2 we apply the first approach to the study of the distinguishability and saturation spectra. In the remaining sections the second approach is applied to the accessibility spectrum, the degree of accessibility and to the degree of reconstructibility.

Again, as in Chapter IV, the automata considered in this chapter will be automata (or automaton graphs) with numbered states, for fixed input and output alphabets $X = \{x_1, \ldots, x_m\}$, $Y = \{y_1, \ldots, y_n\}$, where $m = \text{const} \geqq \geqq 2$ and $n = \text{const} \geqq 2$.

V.1. Uniform statistical estimate of degree of distinguishability

Direct estimation of the frequencies involves certain difficulties, and we therefore proceed as follows.

The construction of automata from an automaton graph G will be envisaged as a stochastic procedure: each edge of the graph G is assigned an output label, i.e., a letter in the output alphabet $Y = \{y_1, \ldots, y_n\}$, in random fashion (giving all letters of the output alphabet Y equal probabilities $1/n$).

We shall find a connection between the frequencies and the probabilities with which our procedure yields automata of any desired type. This will enable us to replace frequencies by probabilities. Let E be some property (e.g., "the automaton has degree of distinguishability 10"). Let $\tilde{G}^E$ denote the set of all automata in $\tilde{G}$ that possess property E, and $p(\tilde{G}^E)$ the probability that the stochastic procedure described above derives from G an automaton with property E (i.e., an automaton in $\tilde{G}^E$).

LEMMA 1.

$$\frac{|\tilde{G}^E|}{|\tilde{G}|} = p(\tilde{G}^E).$$

Proof. It is readily seen that all automata in $\tilde{G}$ are equiprobable outcomes of the above stochastic procedure. This implies the lemma.

We now prove a few lemmas about distinguishability of states in automata derived from the automaton graph G.

Consider an input word $x = x(1)\dots x(t)$ and the paths issuing from vertices q_1 and q_2 of G that this word determines (Figure 44):

$$(q_1 = q'(1))d'(1)q'(2)d'(2)\dots q'(i)d'(i)\dots d'(t)q'(t+1),$$

$$(q_2 = q''(1))d''(1)q''(2)d''(2)\dots q''(i)d''(i)\dots d''(t)q''(t+1)$$

(q' and q'' are vertices, d' and d'' edges). If $d'(j) = d''(j)$ for some j, then $q'(j+1) = q''(j+1)$ (i.e., these states are merged), and so, from that point on (for $i > j$), $d'(i) = d''(i)$. Now let $d'(i) \neq d''(i)$ for some i (so that $d'(j) \neq d''(j)$ for all $j < i$). If one of the edges $d'(i)$, $d''(i)$ has not appeared before, i.e., is different from $d'(j)$, $d''(j)$ for $j < i$, we shall say that the *letter* $x(i)$ *has a primary occurrence* in x. The number of primary occurrences of letters is an important parameter of the input word x.

Let $P(G,x)$ denote the probability that states q_1 and q_2 of the automaton generated by our stochastic procedure from the graph G are indistinguishable by the input word x.

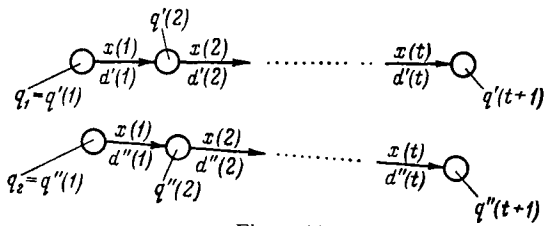


Figure 44

LEMMA 2. *Assume that at least ω letters have primary occurrences in input word x. Then*

$$P(G,x) \leqq (1/n)^{\omega},$$

where n is the cardinality of the output alphabet Y.

Proof. Given the word $x = x(1)\dots x(t)$. The stochastic procedure generating an automaton from the graph G splits into the following $t+1$ steps:

Step S_1. Assign random output labels from Y to the edges $d'(1)$ and $d''(1)$.

Step S_i $(i \leqq t)$. If the edge $d'(i)$ (or $d''(i)$) has not yet been assigned an output label, assign it a random output label from Y.

Step S_{t+1}. Assign random output labels from Y to all edges not yet labeled.

Let the letter $x(i)$ have a primary occurrence in x. Then at least one of the edges $d'(i)$, $d''(i)$, say $d'(i)$, is different from $d'(j)$, $d''(j)$ for $j < i$, and so the edge $d'(i)$ has not yet been assigned an output label at any step $S_j (j < i)$. Hence the step at which the edge $d'(i)$ is assigned a random output label from Y is precisely S_i. Note, moreover, that $d'(i) \neq d''(i)$. It follows that the probability that the step S_i (and consequently the entire stochastic procedure) will assign the edges $d'(i)$ and $d''(i)$ identical output labels is independent of the outcomes of previous steps and therefore equal to $1/n$ (where n is the cardinality of the alphabet Y). This proves the lemma.

We have defined a primary occurrence of a letter in a word with respect to vertices q_1 and q_2. Later we shall need the general case, replacing the vertices q_1 and q_2 by arbitrary vertices q_α and q_β. Instead of saying that the letter has a "primary occurrence," we shall then say that it has a (q_α, q_β)-*primary occurrence*. It is clear that Lemma 2 remains valid when "primary" is replaced by "(q_α, q_β)-primary" and $P(G,x)$ by $P(G,x,q_\alpha,q_\beta)$, where $P(G,x,q_\alpha,q_\beta)$ is the probability that the stochastic procedure will generate an automaton in which q_α and q_β are indistinguishable by the word x. In other words:

LEMMA 2′. *Assume that at least* ω *letters have* (q_α, q_β)*-primary occurrences in the input word* x. *Then*

$$P(G,x,q_\alpha,q_\beta) \leqq (1/n)^\omega.$$

We need a few more definitions. Consider the input word $x = x(1) \ldots x(t)$ and the paths that it defines issuing from the vertices q_1 and q_2 of the graph G. There obviously exists $i_0 \leqq t$ (Figure 45) such that $d'(i) \neq d''(i)$ for all $i \leqq i_0$ and $d'(i) = d''(i)$ for all $i > i_0 \neq t$. We call the initial segment $x(1) \ldots x(i_0)$ of the word x *the essential segment of the word* x and denote it by $\tilde{x}$. Let $W = \{x_{(1)}, \ldots, x_{(s)}, \ldots, x_{(u)}\}$ be a sequence of input words, where $x_{(s)} = x_{(s)}(1) \ldots x_{(s)}(i) \ldots x_{(s)}(t_s)$ is the s-th word in the sequence. $x_{(s)}$ defines two paths:

$$(q_1 = q'_{(s)}(1)) d'_{(s)}(1) \ldots q'_{(s)}(i) d'_{(s)}(i) \ldots d'_{(s)}(t_s) q'_{(s)}(t+1),$$

$$(q_2 = q''_{(s)}(1)) d''_{(s)}(1) \ldots q''_{(s)}(i) d''_{(s)}(i) \ldots d''_{(s)}(t_s) q''_{(s)}(t_s+1).$$

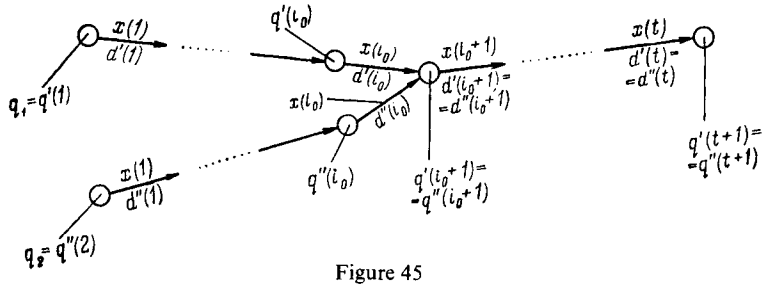


Figure 45

Now assume that $d'_{(s)}(i) \neq d''_{(s)}(i)$. If one of the edges $d'_{(s)}(i), d''_{(s)}(i)$ does not appear in any of the paths defined by the essential segments $\tilde{x}_{(1)}, \ldots, \tilde{x}_{(s-1)}$ of the preceding words of the sequence W and, moreover, is different from $d'_{(s)}(j), d''_{(s)}(j)$ for $j < i$, we shall say that *the letter* $x_{(s)}(i)$ *has a primary occurrence in the sequence* W. As an example, consider the graph of Figure 46 and the sequence of input words $W = \{x_{(1)}, x_{(2)}\}$, where

$$x_{(1)} = x_{(1)}(1)x_{(1)}(2)x_{(1)}(3) = x_2x_2x_2,$$

$$x_{(2)} = x_{(2)}(1)x_{(2)}(2) = x_1x_2.$$

It is not hard to see that the letters $x_{(1)}(1), x_{(1)}(2), x_{(2)}(1)$ have primary occurrences in this sequence.

The number of primary occurrences of letters is an important parameter of a sequence of input words.

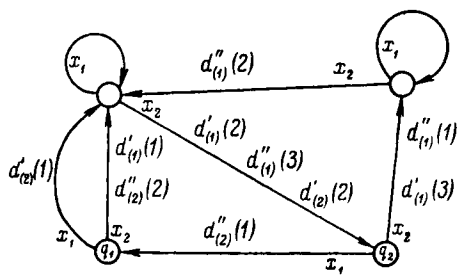


Figure 46

Let $P(G, W)$ denote the probability that states q_1, q_2 of the automaton generated by our stochastic procedure from the graph G are indistinguishable by input words from the sequence W.

Later we shall need the following generalization of Lemma 2:

LEMMA 3. *Assume that the sequence W of input words contains at least ω words with primary occurrences. Then*

$$P(G,W) \leqq (1/n)^{\omega}.$$

Proof. Let $W = \{x_{(1)}, x_{(2)}, \ldots, x_{(n)}\}$ be a sequence of input words and $\tilde{x}_{(1)} = x_{(1)}(1) \ldots x_{(1)}(i_1)$, $\tilde{x}_{(2)} = x_{(2)}(1) \ldots x_{(2)}(i_2), \ldots$ their essential segments. Consider the letters belonging to these segments, and split the stochastic procedure into steps $S_{11}, S_{12}, \ldots, S_{1i_1}, S_{21}, S_{22}, \ldots, S_{2i_2}, \ldots$, where step $S_{\alpha\beta}$ is as follows: if the edge $d'_{(\alpha)}(\beta)$ (or $d''_{(\alpha)}(\beta)$) corresponding to the letter $x_{\alpha}(\beta)$ has not been assigned an output label at previous steps, it is assigned a random output label from Y. It is not difficult to see that with this subdivision of the stochastic procedure we can repeat the reasoning used to prove Lemma 2; this completes the proof of Lemma 3.

We need one more auxiliary concept. Let $x = x(1) \ldots x(t)$ be a word and l_1, l_2 the paths that it defines issuing from the vertices q_1, q_2 of the graph G (or the automaton $\mathfrak{M}$ derived from G) (Figure 45).

As already stated, there exists $i_0 \leqq t$ such that $d'(i) \neq d''(i)$ for all $i \leqq i_0$ and $d'(i) = d''(i)$ for all $i > i_0 \neq t$.

The paths

$$(q_1 = q'(1))d'(1) \ldots d'(i_0)q'(i_0 + 1),$$

$$(q_2 = q''(1))d''(1) \ldots d''(i_0)q''(i_0 + 1),$$

(that is to say, the unmerged parts of the paths l_1 and l_2) are called the *essential paths* of the word x in the graph G (the automaton $\mathfrak{M}$).

We shall now establish some properties of essential paths. Obviously, in order to determine whether states q_1 and q_2 of the automaton derived from the graph G are distinguishable by a word x we need only consider essential paths. In the sequel we shall need a stronger assertion: Let $\mathfrak{M}$ be an automaton derived from G. Delete from this automaton arbitrary vertices (i.e., states) other than $q'(1), \ldots, q'(i_0), q''(1), \ldots, q''(i_0)$ (the terminal vertices $q'(i_0 + 1)$ and $q''(i_0 + 1)$ of the essential paths of x may also be deleted). Delete the edges issuing from the deleted vertices. Join those of the remaining edges leading to deleted vertices to the vertex q_1. The (input and output) labels of the remaining edges are not altered. The result is an automaton $\mathfrak{M}'$. We shall refer to this construction of an automaton $\mathfrak{M}'$ from $\mathfrak{M}$ as a *construction preserving the essential paths of* x. The justification for this designation is that x generates the same essential paths in $\mathfrak{M}'$ and $\mathfrak{M}$, except that the states $q'(i_0 + 1)$ and $q''(i_0 + 1)$ may be replaced by q_1 (since deletion of $q'(i_0 + 1)$ and $q''(i_0 + 1)$ is permitted in the construction of $\mathfrak{M}'$; in this

case the edges $d'(i_0)$ and $d''(i_0)$ are necessarily joined to q_1). It follows that the automaton $\mathfrak{M}'$ has the important property that states q_1 and q_2 are distinguishable by a word x in $\mathfrak{M}$ if and only if they are distinguishable by x in $\mathfrak{M}'$.

We now consider the connection between the number of edges in the essential paths of the word x and the number of primary occurrences of letters in x. Suppose that for some i one of the edges $d'(i)$, $d''(i)$ of the essential paths of x is different from all preceding edges of these paths; then, clearly, the letter $x(i)$ has a primary occurrence in x. Therefore, if a is the number of different edges in the essential paths of the word x and b the number of primary occurrences of letters in x, then, obviously,

$$b \geqq \tfrac{1}{2}a. \tag{1}$$

Now consider all input words of length r (where r is any natural number) and the corresponding essential paths in the graph G. Denote the set of edges of which these paths consist by $U_G(r)$. It is clear that $U_G(1) \subseteq U_G(2) \subseteq \ldots \subseteq U_G(r) \subseteq \ldots$. The sequence must therefore level off at some r (which we denote by R_G):

$$U_G(R_G) = U_G(R_G + 1) = \ldots$$

Obviously,

$$|U_G(r)| \leqq |U_G(R_G)|$$

for any natural number r. In the sequel, the number $|U_G(r)|$ will play an essential role, as an intermediate parameter in estimating the degree of distinguishability of the automata derived from G. Our immediate problem is to estimate $|U_G(r)|$.

Arrange all the input words of length r arbitrarily in a sequence, fixed from now on. Let $\omega_G(r)$ denote the number of primary occurrences of letters in this sequence. By the same arguments as before, one can show that inequality (1) remains valid if the single word x is replaced by all words of length r. In other words,

$$\omega_G(r) \geqq \tfrac{1}{2}|U_G(r)|. \tag{2}$$

Let $\overline{U}_G(r)$ denote the set of vertices in G from which issue the edges in the set $U_G(r)$. Since each edge can lead to one vertex only and at least one edge of the set $U_G(r)$ issues from each vertex $q \in \overline{U}_G(r)$, we have

$$|\overline{U}_G(r)| \leqq |U_G(r)|. \tag{3}$$

In the general case, the set $\overline{U}_G(r)$ may contain vertices q_i other than q_1 and q_2. Obviously, each such vertex q_i is the endpoint of at least one edge from the set $U_G(r-1)$. Since each edge can lead to one vertex only,

$$|\overline{U}_G(r)| \leqq |U_G(r-1)| + 2. \tag{4}$$

LEMMA 4. *If states q_1 and q_2 of an automaton $\mathfrak{M}$ derived from the graph G are distinguishable by input words of length r, they are also distinguishable by input words of length $|\overline{U}_G(r)| - 1$.*

Proof. Starting from the automaton $\mathfrak{M}$, construct another automaton $\mathfrak{M}'$, with state set $\overline{U}_G(r)$, as follows. Delete from the automaton $\mathfrak{M}$ all vertices not belonging to $\overline{U}_G(r)$, together with the edges issuing from them. Join any of the remaining edges which have previously led to a deleted vertex to the vertex $q_1 \in \overline{U}_G(r)$; the (input and output) labels of the remaining edges are not altered.

It is obvious that this construction of $\mathfrak{M}'$ preserves the essential paths of any word x of length r. Therefore, states q_1 and q_2 of the automaton $\mathfrak{M}'$ are distinguishable by input words of length r if and only if states q_1 and q_2 of $\mathfrak{M}$ are distinguishable by input words of this length. But the automaton $\mathfrak{M}'$ has $|\overline{U}_G(r)|$ states, and by Theorem 2.14 its degree of distinguishability is at most $|\overline{U}_G(r)| - 1$. Thus, if states q_1 and q_2 of the automaton $\mathfrak{M}'$ are distinguishable by input words of length r, they are also distinguishable by input words of length $|\overline{U}_G(r)| - 1$. This proves the lemma.

COROLLARY. *If states q_1 and q_2 of an automaton $\mathfrak{M}$ derived from a graph G are distinguishable, they are distinguishable by input words of length $|U_G(R_G)| - 1$.*

To prove this corollary it suffices to note that

a) $|\overline{U}_G(r)| \leqq |U_G(r)|$ by (3);

b) $|U_G(r)| \leqq |U_G(R_G)|$.

LEMMA 5. *For any natural number $r < R_G$,*

$$|U_G(r)| \geqq r.$$

Proof. Assume that this is false, so that $|\overline{U}_G(r')| < r'$ for some $r' < R_G$. Since $r' < R_G$, there exists $r_0 \geqq r'$ such that $U_G(r') = U_G(r'+1) = \ldots = U_G(r_0)$ while $U_G(r_0) \neq U_G(r_0+1)$. Then $|U_G(r_0)| = |U_G(r')| < r' \leqq r_0$. Since $U_G(r_0) \neq U_G(r_0+1)$, there exists an edge (say d_0) which is

in $U_G(r_0 + 1)$ but not in $U_G(r_0)$. The edge d_0 obviously has the following properties:

1. For any input word $x = x(1)\dots x(t)$ of length less than $r_0 + 1$, the edge d_0 either does not belong to the paths from q_1 and q_2 defined by x, or belongs only to the merged part of these paths (i.e., if $d'(i) \neq d''(i)$, then $d'(i) \neq d_0$ and $d''(i) \neq d_0$).

2. There exists an input word $\bar{x} = \bar{x}(1)\dots\bar{x}(r_0 + 1)$ of length $r_0 + 1$ such that if

$$(q_1 = \bar{q}'(1))\bar{d}'(1)\dots\bar{d}'(r_0 + 1)\bar{q}'(r_0 + 2),$$

$$(q_2 = \bar{q}''(1))\bar{d}''(1)\dots\bar{d}''(r_0 + 1)\bar{q}''(r_0 + 2)$$

are the paths that it defines then, first, $\bar{d}'(r_0 + 1) \neq \bar{d}''(r_0 + 1)$ and, second, d_0 is either $\bar{d}'(r_0 + 1)$ or $\bar{d}''(r_0 + 1)$.

We now construct an automaton as follows: assign to all edges of the graph G except d_0 the same output label, say y_1; to the edge d_0 assign a different output label, say y_2. This automaton, call it $\mathfrak{M}_0$, has the property that states q_1 and q_2 are indistinguishable by input words of length r_0 but distinguishable by input words of length $r_0 + 1$. Apply Lemma 4 to the automaton $\mathfrak{M}_0$. It follows that $|\overline{U}_G(r_0 + 1)| - 1 > r_0$ (since otherwise states q_1 and q_2 would be distinguishable by input words of length r_0). By inequality (4),

$$|\overline{U}_G(r_0 + 1)| \leqq |U_G(r_0)| + 2.$$

Consequently,

$$|U_G(r_0)| + 2 - 1 = |U_G(r_0)| + 1 > r_0.$$

Since $|U_G(r_0)|$ and r_0 are natural numbers, this inequality means that $|U_G(r_0)| \geqq r_0$. But this contradicts the inequality $|U_G(r_0)| < r_0$, proving the lemma.

As in Section IV.7, let $\tilde{G}_{\rho > r}$ denote the set of all automata in $\tilde{G}$ with degree of distinguishability greater than r. Obviously if $r_1 \geqq r_2$ then $\tilde{G}_{\rho > r_1} \subseteq \subseteq \tilde{G}_{\rho > r_2}$.

THEOREM 5.1. *For any automaton graph G and any natural number r,*

$$\frac{|\tilde{G}_{\rho > r}|}{|\tilde{G}|} < |G|^2 \left(\frac{1}{n}\right)^{r/2}$$

($|G|$ is the number of vertices in the graph G and n the cardinality of the output alphabet Y).

Proof. Let $p(\tilde{G}_{\rho>r})$ be the probability that the stochastic procedure for construction of automata from G generates an automaton with degree of distinguishability greater than r. By Lemma 1 (reduction to probabilities), $|\tilde{G}_{\rho>r}|/|G| = p(\tilde{G}_{\rho>r})$. The theorem will therefore be proved if we can show that for all natural numbers r

$$p(\tilde{G}_{\rho>r}) < |G|^2 \left(\frac{1}{n}\right)^{r/2}.$$

Let $p(\tilde{G}_{\rho>r}, q_i, q_j)$ be the probability that our stochastic procedure generates an automaton whose states q_i and q_j are indistinguishable by input words of length r but distinguishable by longer input words. Obviously,

$$p(\tilde{G}_{\rho>r}) \leqq \sum_{\substack{i,j \\ i\neq j, i<|G|, j<|G|}} p(\tilde{G}_{\rho>r}, q_i, q_j).$$

We shall prove that

$$p(\tilde{G}_{\rho>r}, q_1, q_2) \leqq \left(\frac{1}{n}\right)^{r/2}. \tag{5}$$

There are two cases:

1) $r < R_G$,

2) $r \geqq R_G$.

Consider Case 1. Applying Lemma 5, we see that $|U_G(r)| \geqq r$. Arrange the input words of length r in some arbitrary fixed sequence (call it W_r). Let $\omega_G(r)$ be the number of primary occurrences of letters in this sequence. By inequality (2),

$$\omega_G(r) \geqq \tfrac{1}{2}|U_G(r)|,$$

and so $\omega_G(r) \geqq r/2$. Apply Lemma 3 to the sequence W_r. Then

$$P(G, W_r) \leqq \left(\frac{1}{n}\right)^{\omega_G(r)} \leqq \left(\frac{1}{n}\right)^{r/2},$$

where $P(G, W_r)$ is the probability that our stochastic procedure generates an automaton in which q_1 and q_2 are indistinguishable by input words of the sequence W_r, i.e., by input words of length r. It is readily seen that $p(G_{\rho>r}, q_1, q_2) \leqq P(G, W_r)$. Therefore $p(G_{\rho>r}, q_1, q_2) \leqq (1/n)^{r/2}$, and this proves inequality (5) for Case 1.

Now consider Case 2. Here $U_G(r) = U_G(R_G)$, and we distinguish two subcases:

a) $|U_G(r)| = |U_G(R_G)| \geqq r$;

b) $|U_G(r)| = |U_G(R_G)| < r$.

Subcase a) is treated in analogy to Case 1, except that there is no need to use Lemma 5, since the inequality $|U_G(r)| \geqq r$ is given.

Consider subcase b). It follows from the Corollary to Lemma 4 that, in any automaton derived from the graph G, if states q_1 and q_2 are distinguishable, then they are distinguishable by input words of length $|U_G(R_G)| - 1 < r - 1$. This means that $p(\tilde{G}_{\rho>r}, q_1, q_2) = 0$, which proves inequality (5).

Thus, in all cases $p(\tilde{G}_{\rho>r}, q_1, q_2)$ satisfies the estimate given by inequality (5).

It is clear that the same estimate holds for $p(\tilde{G}_{\rho>r}, q_i, q_j)$ when q_i and q_j are arbitrary states (since this merely involves renumbering the states of the graph G). Now the number of pairs (i,j) with $i \neq j$, $i \leqq |G|, j \leqq |G|$, is less than $|G|^2$. This implies that

$$p(\tilde{G}_{\rho>r}) \leqq \sum_{\substack{i,j \\ i\neq j, i\leqq|G|, j\leqq|G|}} p(\tilde{G}_{\rho>r}, q_i, q_j) < |G|^2 \left(\frac{1}{n}\right)^{r/2}.$$

This completes the proof of the theorem.

Theorem 5.1 has a corollary (Theorem 5.2) which is of independent interest.

Following Section IV.3, let us say that uniformly almost all automata with k states possess a property E if

$$\min_{|G|=k} \frac{|\tilde{G}^E|}{|\tilde{G}|} \to 1 \quad \text{as} \quad k \to \infty,$$

where $|G|$, as usual, is the number of vertices in the graph G and $\tilde{G}^E$ the set of automata in $\tilde{G}$ that possess property E.

THEOREM 5.2 (ON THE UNIFORM DEGREE OF DISTINGUISHABILITY). *Uniformly almost all automata $\mathfrak{M}$ with k states have degree of distinguishability $\rho(\mathfrak{M})$ at most $C \log_n k$, where C is a constant.*

Proof. Let G be an arbitrary automaton graph with k vertices. By Theorem 5.1,

$$\frac{|\tilde{G}_{\rho>r}|}{|\tilde{G}|} < k^2 \left(\frac{1}{n}\right)^{r/2}.$$

Set $r =]5 \log_n k[$; then

$$\frac{|\tilde{G}_{\rho>]5\log_n k[}|}{|\tilde{G}|} < \left(\frac{1}{k}\right)^{1/2}.$$

Hence,

$$\min_{|G| = k} \frac{|\tilde{G}_{\rho >]5 \log_n k[}|}{|\tilde{G}|} \to 0 \quad \text{as} \quad k \to \infty.$$

This means that uniformly almost all automata with k states have degree of distinguishability $\rho(\mathfrak{M})$ at most $]5 \log_n k[$.

It is not difficult to see that the order of magnitude of this estimate for the degree of distinguishability cannot be improved. In other words, there exists a positive constant C' such that the statement "uniformly almost all automata $\mathfrak{M}$ with k states have degree of distinguishability $\rho(\mathfrak{M}) < C' \log_n k$" is false. To see this it suffices to consider, for example, the graph G_k illustrated in Figure 42. For sufficiently large C_0, its vertices $q_0, q_{C_0 \log_n k}, q_{2C_0 \log_n k}, \ldots, q_{iC_0 \log_n k}, \ldots$ have the following property: if $p(k)$ denotes the probability that the stochastic procedure generating automata from G_k yields an automaton in which the states $q_0, q_{C_0 \log_n k}, \ldots, q_{iC_0 \log_n k}, \ldots$ are distinguishable, then $p(k) \to 1$ as $k \to \infty$. Since the number of these states is at least $k/C_0 \log_n k$, it follows (in view of the peculiar properties of G_k) that the minimal length of input words by which they are all distinguishable must be at least $(1/\log_n mn) \log_n (k/C_0 \log_n k)$, and so at least $C' \log_n k$.

However, if we drop the word "uniformly" from the statement of Theorem 5.2, the estimate for the degree of distinguishability may be considerably improved. Korshunov [37] has proved the following theorem:

Almost all automata with k states have degree of distinguishability asymptotically equal to $\log_m \log_n k$.

Recall that by Theorem 2.14 any automaton with reduced weight μ has degree of distinguishability at least $\log_m \log_n \mu - 1$.

V.2. Uniform statistical estimate of the saturation spectrum

In this section we shall estimate the relative frequency of automata $\mathfrak{M}$ in $\tilde{G}$ whose saturation spectrum satisfies the inequality $F_{\langle \mathfrak{M}, q_0 \rangle}(l) \leqq s$. As in Section IV.6, let $\tilde{G}_{F(l) \leqq s}$ denote the set of all automata $\mathfrak{M}$ in $\tilde{G}$ such that $F_{\langle \mathfrak{M}, q_0 \rangle}(l) \leqq s$.

We define the *height of an automaton graph* G as the minimal number h_G such that any vertex q_j accessible from q_0 is accessible by an input word of length at most h_G.

THEOREM 5.3. *For any automaton graph G of height* h_G *and any natural numbers* s *and* ψ *such that* $s + \psi \leqq h_G$, *we have*

$$\frac{|\tilde{G}_{F(s+\psi)\leqq s}|}{|\tilde{G}|} \leqq \frac{s(s+1)}{2}\left(\frac{1}{n}\right)^{\psi},$$

where n is the cardinality of the output alphabet Y.

Proof. Let $p(\tilde{G}_{F(l)\leqq s})$ be the probability that the stochastic procedure for construction of an automaton from G generates an automaton $\mathfrak{M}$ such that $F_{\langle \mathfrak{M},q_0\rangle}(l) \leqq s$. By Lemma 1,

$$\frac{|\tilde{G}_{F(l)\leqq s}|}{|\tilde{G}|} = p(\tilde{G}_{F(l)\leqq s}).$$

It will thus suffice to prove that if s and ψ satisfy the assumptions of the theorem then

$$p(\tilde{G}_{F(s+\psi)\leqq s}) \leqq \frac{s(s+1)}{2}\left(\frac{1}{n}\right)^{\psi}.$$

By assumption, $s+\psi \leqq h_G$. Therefore, there exists a word $x^0 = x^0(1)x^0(2)\dots x^0(s+\psi)$ such that the vertices

$$q_0,\ q_0x^0(1),\ q_0x^0(1)x^0(2),\dots,q_0x^0(1)\dots x^0(s+\psi)$$

lie at heights $0,1,2,\dots,s+\psi$, respectively (Figure 47). To abbreviate the notation we shall denote these vertices by $q_0^0,q_1^0,q_2^0,\dots,q_{s+\psi}^0$. We are mainly interested in the vertices $q_0^0,q_1^0,\dots,q_s^0$. Let i and j be fixed numbers, $0 \leqq i < j \leqq s$. Consider the pair of vertices q_i^0, q_j^0 and the word* $x^0(j+1)\dots x^0(s)$. This word takes the vertex q_j^0 to q_s^0. Let q_{ij} be the vertex to which this word takes q_i^0. Since q_i^0 lies lower than q_j^0 (i.e., $i<j$), it is not hard to see that q_{ij} lies lower than q_s^0. Consider the paths leading from the vertices q_{ij} and q_s^0 defined by the word $x^0(s+1)\dots x^0(s+\psi)$. It is obvious that for any $1 \leqq r \leqq \psi$ the vertex $q_{ij}x^0(s+1)\dots x^0(s+r)$ lies lower than the vertex $q_s^0x^0(s+1)\dots x^0(s+r) = q_{s+r}^0$. Therefore, the edge incident on the vertex q_{s+r}^0 is distinct from all preceding edges in these paths. Hence the letter $x^0(s+r)$ has a (q_{ij},q_s^0)-primary occurrence in this word. We have thus verified that all the letters of the word $x^0(s+1)\dots x^0(s+\psi)$ have (q_{ij},q_s^0)-primary occurrences there. As before, let $P(G,x,q_\alpha,q_\beta)$ denote the probability that the stochastic process for construction of an automaton

* If $j=s$, the word $x^0(j+1)\dots x^0(s)$ is empty.

from G yields an automaton in which the states q_α and q_β are indistinguishable by the word x. Using Lemma 2', we see that

$$P(G, x^0(s+1)\dots x^0(s+\psi), q_{ij}, q_s^0) \leqq \left(\frac{1}{n}\right)^\psi.$$

Let $\mathfrak{M}$ be an automaton derived from the graph G and $V_{\mathfrak{M}}$ the complete tree, of height $s+\psi$ and root q_*, compatible with $\mathfrak{M}$. With each state $q_\alpha^0 = q_0 x^0(1)\dots x^0(\alpha)$ $(\alpha = 0, 1, \dots, s)$ of the automaton $\mathfrak{M}$, associate the vertex $q_\alpha^* = q_* x^0(1)\dots x^0(\alpha)$ of the tree $V_{\mathfrak{M}}$. It is easy to verify that if states q_{ij} and q_s^0 of the automaton $\mathfrak{M}$ are distinguishable by the word $x^0(s+1)\dots x^0(s+\psi)$, then the vertices q_i^* and q_j^* of the tree $V_{\mathfrak{M}}$ are distinguishable. Indeed, if q_{ij} and q_s^0 are distinguishable by the word $x^0(s+1)\dots x^0(s+\psi)$, then states q_i^0 and q_j^0 are distinguishable by the word $x^0(j+1)\dots x^0(s+1)\dots x^0(s+\psi)$, and so the vertices q_i^* and q_j^* of the tree $V_{\mathfrak{M}}$ are also distinguishable. Let $p(q_i^*, q_j^*)$ denote the probability that our stochastic procedure generates an automaton $\mathfrak{M}$ in whose tree $V_{\mathfrak{M}}$ of height $s+\psi$ the vertices q_i^*, q_j^* are indistinguishable, and p^* the probability of generating an automaton for which at least two vertices q_i^*, q_j^* $(0 \leqq i \leqq j \leqq s)$ are indistinguishable. It follows from the foregoing arguments that

$$p(q_i^*, q_j^*) \leqq P(G, x^0(s+1)\dots x^0(s+\psi), q_{ij}, q_s^0) \leqq \left(\frac{1}{n}\right)^\psi$$

and so

$$p^* \leqq \sum_{\substack{i,j \\ 0 \leqq i < j \leqq s}} p(q_i^*, q_j^*) \leqq \frac{s(s+1)}{2}\left(\frac{1}{n}\right)^\psi.$$

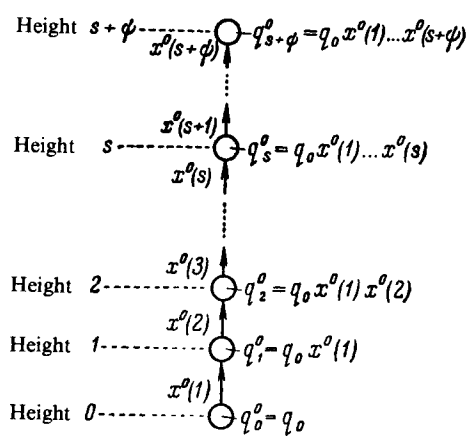


Figure 47

Note that the number of vertices $q_0^*, q_1^*, \ldots, q_s^*$ is $s + 1$, which is of course greater than s. Therefore, if any two vertices $q_i^*, q_j^* (0 \leqq i < j \leqq s)$ of the tree $V_{\mathfrak{M}}$ are distinguishable, the tree contains more than s distinguishable vertices, and therefore $F_{\langle \mathfrak{M}, q_0 \rangle}(s + \psi) > s$. Therefore,

$$p(\tilde{G}_{F(s+\psi) \leqq s}) \leqq p^* \leqq \frac{s(s+1)}{2} \left(\frac{1}{n}\right)^{\psi}.$$

This completes the proof.

V.3. Stochastic procedure generating automaton graphs

From this section through Section V.8 we shall be dealing with statistical properties of automaton graphs. Specifically, we wish to study the frequencies with which automaton graphs possess certain properties related to accessibility spectra. It is very difficult to estimate these frequencies directly. We shall therefore envisage the construction of automaton graphs as a stochastic procedure. The next step will be to establish the relation between the frequencies and the probabilities with which the procedure generates the required automaton graphs. We shall then be able to replace frequencies by probabilities.

The present section will describe a stochastic model for the generation of automaton graphs, and will also determine the relation between frequencies and probabilities.

Note that, as before, we are dealing exclusively with automaton graphs with numbered vertices $q_0, q_1, \ldots$ and fixed input alphabet $X = \{x_1, \ldots, x_m\}$, where $m = \text{const} \geqq 2$.

Our "raw material" will be k vertices $q_0, q_1, \ldots, q_{k-1}$, from each of which issue m edges with free endpoints, one labeled x_1, the second $x_2, \ldots$, the last labeled x_m (Figure 48).

Consider one of these edges; suppose that we choose a vertex at random and connect to it the free end of this edge (assuming that the vertices are equiprobable, so that each is chosen with probability $1/k$). We shall refer to this procedure as the random connection of the edge to the vertices $q_0, q_1, \ldots, q_{k-1}$. Now, if all the edges issuing from the vertices $q_0, \ldots, q_{k-1}$ are connected at random to these vertices, the result is a certain (random) automaton graph with k vertices. The random connection of all edges may be imagined in various ways; for example, one might stipulate that all connections be made simultaneously. Alternatively, the stochastic procedure

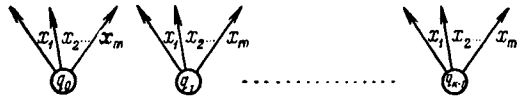


Figure 48

for generation of an automaton graph might be as follows (and this approach will be adopted from now on).

Step 1. Choose a vertex q_0 (henceforth termed the *vertex of level* 0), and connect the edges issuing from it (the *edges of level* 0) at random to the vertices $q_0, q_1, \ldots, q_{k-1}$. Call all the vertices other than q_0 (with the edges issuing from them), to which edges of level 0 are connected, *vertices of level* 1 (or the *level constructed at step* 1). The object consisting of the first two levels is called the 1-*base*, and the edges issuing from the vertices of level 1 are called the edges of level 1. If it turns out that all edges of level 0 have been connected to q_0, then all the edges issuing from the vertices $q_1, \ldots, q_{k-1}$ are connected at random to the vertices $q_0, q_1, \ldots, q_{k-1}$, and the procedure halts.

Step h ($h = 2,3,\ldots$). Assume that an $(h-1)$-base has been constructed. This means that all edges of the first $h-2$ levels have already been connected to suitable vertices, while the edges of level $h-1$ (and also the edges issuing from vertices not included in the $(h-1)$-base) have free ends. Connect all edges of level $h-1$ at random to the vertices $q_0, \ldots, q_{k-1}$. Call the vertices (together with the edges issuing from them) to which the edges of level $h-1$ are connected, and which do not appear in the first $h-1$ levels, *vertices of level* h (or the *level constructed at step* h). The vertices of levels $0,1,\ldots,h$, together with the edges issuing from them, form the *h-base*, and the edges issuing from vertices of level h are the *edges of level h*. If it turns out that all edges of level $h-1$ have been connected to vertices of the first $h-1$ levels (i.e., level h is empty), then all edges issuing from vertices outside the $(h-1)$-base are connected at random to the vertices $q_0, \ldots, q_{k-1}$, and the procedure halts.

This completes the description of the process for generation of an automaton graph with k vertices. Henceforth we shall refer to it as the *stochastic procedure for generating automaton graphs*. We now present a simple, but important lemma, analogous to Lemma 1, which establishes the connection between frequencies and probabilities.

Let E be some property of automaton graphs. Let $\mathscr{T}(k)$ denote the set of all pairwise nonidentical automaton graphs with k vertices, and $\mathscr{T}^E(k)$ the set of all pairwise nonidentical automaton graphs in $\mathscr{T}(k)$ which possess

property E. Let $p_E(k)$ be the probability that the above stochastic procedure generates an automaton graph with property E.

LEMMA 6 (REDUCTION TO PROBABILITIES).

$$\frac{|\mathscr{T}^E(k)|}{|\mathscr{T}(k)|} = p_E(k).$$

Proof. The lemma follows from the easily verified fact that in our stochastic procedure all automaton graphs with k vertices are generated with equal probabilities; in other words, all graphs of the set $\mathscr{T}(k)$ are equiprobable. Note that here we are making essential use of the fact that $\mathscr{T}(k)$ is the set of all pairwise nonidentical automaton graphs. Were $\mathscr{T}(k)$ the set of, say, nonisomorphic graphs, Lemma 6 would be false, as is easily seen (see also Supplementary Material to Chapter IV).

V.4. Statistical estimate of the accessibility spectrum for automaton graphs

The aim of this section is to prove the following theorem:

THEOREM 5.4. *There exist positive constants C_1 and C_2 such that almost all automaton graphs G with k vertices have the following property: For any vertex q_i of the graph G and $l \leqq C_1 \log_m k$, the accessibility spectrum $D_{\langle G,q_i\rangle}(l)$ satisfies the inequality*

$$D_{\langle G,q_i\rangle}(l) \geqq m^{C_2 l}.$$

The theorem will be proved by an examination of the stochastic procedure generating automaton graphs with k vertices.

Let $p(k,C_1,C_2)$ be the probability that the stochastic procedure generates an automaton graph not satisfying the statement of the theorem, i.e., there exists a vertex $q_i \in G$ such that

$$D_{\langle G,q_i\rangle}(l) \geqq m^{C_2 l}$$

is false for some $l \leqq C_1 \log_m k$. By Lemma 6, Theorem 5.4 will be proved if we can show that there exist $C_1 > 0$ and $C_2 > 0$ such that

$$p(k,C_1,C_2) \to 0 \text{ as } k \to \infty.$$

Let $p_0(k,C_1,C_2)$ be the probability that our stochastic procedure generates an automaton graph whose accessibility spectrum has the property: $D_{\langle G,q_0\rangle}(l) \geqq m^{C_2 l}$ is false for some $l \leqq C_1 \log_m k$.

Obviously

$$p(k,C_1,C_2) \leqq kp_0(k,C_1,C_2).$$

Therefore, the theorem will be proved if we show that there exist $C_1 > 0$, $C_2 > 0$ and $\varepsilon > 0$ such that, for sufficiently large k,*

$$p_0(k,C_1,C_2) \leqq \frac{1}{k^{1+\varepsilon}}.$$

If the h-base of an automaton graph G can be converted into a tree by deleting exactly i edges (naturally, edges issuing from the first $h-1$ levels), we shall say that the *h-base of G differs from a tree at i points.*

Assume that the h-base of a graph G differs from a tree at most at one point, i.e., it is either a tree or can be converted into one by deleting a single edge. It is then easily seen that for any $l \leqq h$

$$D_{\langle G,q_0\rangle}(l) \geqq 1 + (m-1) + (m-1)m + \ldots + (m-1)m^{l-1},$$

and so

$$D_{\langle G,q_0\rangle}(l) \geqq m^l.$$

Together with the preceding arguments, this implies that Theorem 5.4 will follow from

LEMMA 7. *Let $p(k)$ be the probability that the stochastic procedure generates an automaton graph whose $]\frac{1}{6}\log_m k[$-base differs from a tree at most at one point. Then, for sufficiently large k,*

$$p(k) \leqq k^{-8/7}$$

Proof. Let us assume that throughout the stochastic procedure the edges (including those within the same level) are connected one by one rather than simultaneously. This implies, of course, that edges of level 0 come first, then those of level 1, and so on. Suppose that in the course of this procedure an edge d is connected to a vertex to which another edge was connected at an earlier stage of the procedure (Figure 49). We shall then say that *the edge d distorts the tree.* It is clear that $p(k)$ is precisely the probability that more than one edge of the first $]\frac{1}{6}\log_m k[- 1$ levels distort the tree during the procedure.

* Here and below, the phrase "$A(k)$ is valid for sufficiently large k" means that there exists k_0 such that $A(k)$ is valid for all $k \geqq k_0$. Similarly, "$A_{\alpha\beta}(k)$ is true for sufficiently large k depending on α but not on β" means that for any α there exists k_α such that $A_{\alpha\beta}(k)$ is true for any β and all $k \geqq k_\alpha$.

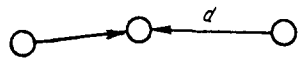


Figure 49

Consider an arbitrary edge d of level $h \leqq]\frac{1}{6}\log_m k[- 1$. Obviously, when our stochastic procedure dictates that this edge be connected at random to the vertices $q_0, q_1, \ldots, q_{k-1}$, the number of "used" vertices (vertices to which an edge is already connected) must be less than the maximal possible number of vertices of the first $h + 1$ levels, i.e., less than $1 + m + m^2 + \ldots + m^{h+1}$. But $1 + m + m^2 + \ldots + m^{h+1} < m^{h+2} \leqq m^2 l^{1/6}$. Thus the probability that the edge d can distort the tree is less than $m^2 k^{1/6}/k = m^2 k^{-5/6}$. This means that the probability that more than one edge in the first $]\frac{1}{6}\log_m k[- 1$ levels will distort the tree, i.e., $p(k)$, is less than $\sum_{r=2}^{n} b(r,n,p)$, where n is the maximal possible number of edges in the first $]\frac{1}{6}\log_m k[- 1$ levels, $p = m^2 k^{-5/6}$ and $b(r,n,p)$ is the probability of r successful outcomes and $n - r$ failures in n Bernoulli trials with probability p for success and $q = 1 - p$ for failure. Let us estimate $\sum_{r=2}^{n} b(r,n,p)$. Obviously, $n < m^3 k^{1/6}$. Therefore the expectation pn is less than $m^2 k^{-5/6} \times m^3 k^{1/6}$, and so, for sufficiently large k, it is less than 2.

Now the quantity

$$b(r,n,p) = C_n^r p^r q^{n-r}$$

satisfies the inequality $b(r,n,p) > b(r+1,n,p)$ for any $r \geqq pn$. Therefore, in our case, $b(2,n,p) > b(3,n,p) > b(4,n,p) > \ldots$, and so

$$\sum_{r=2}^{n} b(r,n,p) < nb(2,n,p) < m^3 k^{1/6} C^2_{m^3 k^{1/6}} (m^2 k^{-5/6})^2 (1 - m^2 k^{5/6})^{m^3 k^{1/6} - 2} < m^3 k^{1/6} C^2_{m^3 k^{1/6}} (m^2 k^{-5/6})^2 < m^{13} k^{-7/6}.$$

For sufficiently large k, the last expression is bounded above by $k^{-8/7}$. This completes the proof of the lemma.

V.5. Statistical estimate of the diameter. Statement of the fundamental result

The *diameter of an automaton graph* G is the smallest number d_G such that if one of any two vertices q_i and q_j in G is accessible from the other, it is so accessible by an input word of length at most d_G.

It is easy to see that there are automaton graphs with k vertices whose diameters are $k - 1$. Conversely, it is also easily seen that there are automaton graphs with k vertices which have far smaller diameters. Our goal is now to determine the diameter of almost all automaton graphs. We reiterate that we are dealing with automaton graphs over a fixed input alphabet $X = \{x_1, \ldots, x_m\}$, where $m = \text{const} \geqq 2$.

Our fundamental result is

THEOREM 5.5. *Almost all automaton graphs with k vertices have diameter at most* $C \log_m k$, *where C is a constant which depends on m and approaches unity with increasing m.*

We immediately reduce the proof of this theorem to that of a lemma concerning the height of automaton graphs.

Consider the stochastic procedure generating automaton graphs with k vertices. Let $D(k)$ denote the random variable whose values are the diameters of the automaton graphs generated by the procedure.

It follows from Lemma 6 that Theorem 5.5 will be proved if we show that

$$P\{D(k) > C \log_m k\} \to 0 \text{ as } k \to \infty,$$

where C is a constant depending on m which approaches unity with increasing m.

The height of the automaton graph generated by our procedure is a random variable which we denote by $H(k)$. Recall that the height of an automaton graph G is the smallest number h_G such that any vertex q_j accessible from q_0 is accessible by an input word of length at most h_G. Thus the variable $H(k)$ is precisely the number of levels constructed in the course of the stochastic procedure (not counting level 0).

It is easily seen that

$$P\{D(k) > l\} < kP\{H(k) > l\}.$$

Thus, the proof of Theorem 5.5 reduces to verification of the following lemma:

FUNDAMENTAL LEMMA. *There exists a constant C which depends on m and approaches unity with increasing m, such that for sufficiently large k*

$$P\{H(k) > C \log_m k\} < \frac{1}{k^{1+\varepsilon}},$$

where ε *is a positive constant.*

We shall prove the fundamental lemma in Section V.7.

V.6. Auxiliary propositions from probability theory

In this section we prove certain estimates concerning Bernoulli trials and random allocations of numbered balls to numbered boxes. These estimates (see Lemmas 8, 9, 12) will be needed to prove the fundamental lemma.

Consider a sequence of n Bernoulli trials with probability p for success and $q = 1 - p$ for failure. Let $X(n,p)$ denote the random variable defined as the number of successful outcomes in this sequence.

LEMMA 8. *For any $p > 0$, any natural number n and any λ, $0 \leqq \lambda \leqq pn$,*

$$P\{X(n,p) \leqq \lambda\} < e^2 n \left(1 - \frac{pn - \lambda}{2pn}\right)^{pn - \lambda - 1}.$$

Proof. We first prove the following proposition.

A. *For any $p > 0$ and any natural numbers n and α, such that $\alpha \leqq pn$,*

$$P\{X(n,p) \leqq pn - \alpha\} < en \left(1 - \frac{\alpha}{2pn}\right)^{\alpha}.$$

Let $b(r,n,p)$ be the probability of r successful outcomes and $n - r$ failures in n Bernoulli trials with probability p for success and $q = 1 - p$ for failure. Then

$$P\{X(n,p) \leqq \lambda\} = \sum_{r=0}^{\lambda} b(r,n,p).$$

We know that for fixed n and p the quantity

$$b(r,n,p) = C_n^r p^r q^{n-r}$$

satisfies the inequality $b(r-1,n,p) < b(r,n,p)$ for any natural number $r \leqq pn$. Hence, since α is a natural number,

$$P\{X(n,p) \leqq pn - \alpha\} = P\{X(n,p) \leqq [pn - \alpha]\} = P\{X(n,p) \leqq [pn] - \alpha\} <$$

$$< nb([pn] - \alpha, n, p) = nC_n^{[pn]-\alpha} p^{[pn]-\alpha} q^{n-[pn]+\alpha} = nb([pn], n, p) \times$$

$$\times \frac{([pn] - \alpha + 1)([pn] - \alpha + 2) \dots [pn]}{(n - [pn + \alpha)(n - [pn] + \alpha - 1) \dots (n - [pn] + 1)} \left(\frac{q}{p}\right)^{\alpha} <$$

$$< \frac{n(pn - \alpha + 1)(pn - \alpha + 2) \dots pn}{(n - pn + \alpha)(n - pn + \alpha - 1) \dots (n - pn + 1)} \left(\frac{q}{p}\right)^{\alpha} <$$

$$< \frac{n(pn-\alpha+1)(pn-\alpha+2)\dots pn}{(n-pn)^{\alpha}} \left(\frac{q}{p}\right)^{\alpha} =$$

$$= \frac{n(pn-\alpha+1)(pn-\alpha+2)\dots pn}{(pn)^{\alpha}}.$$

It follows from the inequalities

$$(x-\beta)x < (x-\beta+1)(x-1) < \dots < \left(x-\frac{\beta}{2}\right)\left(x-\frac{\beta}{2}\right)$$

that

$$(pn-\alpha+1)(pn-\alpha+2)\dots pn < (pn-\tfrac{1}{2}(\alpha-1))^{\alpha}.$$

Therefore,

$$P\{X(n,p) \leqq pn-\alpha\} < \frac{n(pn-\frac{1}{2}(\alpha-1)^{\alpha}}{(pn)^{\alpha}} =$$

$$= n\left(1-\frac{\alpha-1}{2pn}\right)^{\alpha} = n\left(1-\frac{\alpha}{2pn}\right)^{\alpha}\left(1+\frac{1}{2pn-\alpha}\right)^{\alpha}$$

Since $\alpha \leqq pn$, we have

$$\left(1+\frac{1}{2pn-\alpha}\right)^{\alpha} \leqq \left(1+\frac{1}{pn}\right)^{pn} < e,$$

and so

$$P\{X(n,p) \leqq np-\alpha\} < en\left(1-\frac{\alpha}{2pn}\right)^{\alpha}.$$

This proves Proposition A.

Proposition A was proved under the assumption that α is a natural number. We now consider the case of an arbitrary positive number $\alpha \geqq 1$. Then

$$P\{X(n,p) \leqq pn-\alpha\} \leqq P\{X(n,p) \leqq pn-[\alpha]\}$$

and, by Proposition A,

$$P\{X(n,p) \leqq pn-[\alpha]\} < en\left(1-\frac{[\alpha]}{2pn}\right)^{[\alpha]}.$$

Now the inequalities

$$\left(1-\frac{[\alpha]}{2pn}\right)^{\alpha} < \left(1-\frac{\alpha-1}{2pn}\right)^{\alpha-1}$$

and

$$\left(1 - \frac{\alpha - 1}{2pn}\right)^{\alpha - 1} < e\left(1 - \frac{\alpha}{2pn}\right)^{\alpha - 1},$$

give

$$P\{X(n,p) \leqq pn - \alpha\} < e^2 n \left(1 - \frac{\alpha}{2pn}\right)^{\alpha - 1}.$$

Thus, for any $\lambda, 0 \leqq \lambda \leqq pn - 1$,

$$P\{X(n,p) \leqq \lambda\} = P\{X(n,p) \leqq pn - (pn - \lambda)\} < e^2 n \left(1 - \frac{pn - \lambda}{2pn}\right)^{pn - \lambda - 1}$$

It is easily seen that this inequality also holds for $pn - 1 < \lambda \leqq pn$.

This proves the lemma.

LEMMA 9. *For any $p > 0$, any natural number n and any λ, $pn < \lambda \leqq n$,*

$$P\{X(n,p) \geqq \lambda\} < n \Big/ \left(\frac{\lambda}{pn}\right)^{(\lambda - pn - 3)/2}.$$

Proof. As in the proof of Lemma 8, we first prove a weaker assertion:

B. *For any $p > 0$ and any natural numbers n and α such that $\alpha \leqq n - pn$,*

$$P\{X(n,p) \geqq pn + \alpha\} < n \Big/ \left(1 + \frac{\alpha}{pn}\right)^{\alpha/2}.$$

The proof of this proposition is analogous to that of Proposition A, except that here we use the fact that $b(r,n,p) > b(r+1,n,p)$ for any $r \geqq pn$. We have

$$P\{X(n,p) \geqq pn + \alpha\} = P\{X(n,p) \geqq]pn + \alpha[\} = P\{X(n,p) \geqq]pn[+ \alpha\} <$$

$$< nb(]pn[+ \alpha, n, p) = nC_n^{]pn[+ \alpha} p^{]pn[+ \alpha} q^{n -]pn[- \alpha} = nb(]pn[, n, p) \times$$

$$\times \frac{(n -]pn[)(n -]pn[- 1)\dots(n -]pn[- \alpha + 1)}{(]pn[+ 1)(]pn[+ 2)\dots(]pn[+ \alpha)} \left(\frac{p}{q}\right)^{\alpha} <$$

$$< \frac{n(n - pn)^{\alpha}}{(pn + 1)(pn + 2)\dots(pn + \alpha)} \left(\frac{p}{q}\right)^{\alpha} = \frac{n(pn)^{\alpha}}{(pn + 1)(pn + 2)\dots(pn + \alpha)}.$$

Since

$$(pn + 1)(pn + 2)\dots(pn + \alpha) \geqq ((pn + 1)(pn + \alpha))^{\alpha/2} > (pn)^{\alpha/2}(pn + \alpha)^{\alpha/2},$$

it follows that

$$P\{X(n,p) \geqq pn + \alpha\} < \frac{n(pn)^{\alpha}}{(pn)^{\alpha/2}(pn + \alpha)^{\alpha/2}} = \frac{n}{(1 + \alpha/pn)^{\alpha/2}}.$$

This proves Proposition B.

Proposition B was proved under the assumption that α is a natural number. We now let α be an arbitrary positive number not less than 3. Then

$$P\{X(n,p) \geqq pn + \alpha\} \leqq P\{X(n,p) \geqq pn + [\alpha]\} <$$

$$< n \Big/ \left(1 + \frac{[\alpha]}{pn}\right)^{[\alpha]/2} < n \Big/ \left(1 + \frac{\alpha - 1}{pn}\right)^{(\alpha-1)/2} =$$

$$= n \Big/ \left(\frac{pn + \alpha}{pn} - \frac{1}{pn}\right)^{(\alpha-1)/2} =$$

$$= n \Big/ \left(\frac{pn + \alpha}{pn}\right)^{(\alpha-1)/2} \left(1 - \frac{1}{pn + \alpha}\right)^{(\alpha-1)/2} \leqq$$

$$\leqq n \Big/ \left(\frac{pn + \alpha}{pn}\right)^{(\alpha-1)/2} \left(1 - \frac{\alpha - 1}{2} \cdot \frac{1}{pn + \alpha}\right) <$$

$$< n \Big/ \left(\frac{pn + \alpha}{pn}\right)^{(\alpha-1)/2} \frac{pn}{pn + \alpha} = n \Big/ \left(\frac{pn + \alpha}{pn}\right)^{(\alpha-3)/2}.$$

It follows that for any $\lambda, pn + 3 \leqq \lambda \leqq n$,

$$P\{X(n,p) \geqq \lambda\} = P\{X(n,p) \geqq pn + (\lambda - pn)\} < n \Big/ \left(\frac{\lambda}{pn}\right)^{(\lambda - pn - 3)/2}.$$

It is easily seen that this inequality also holds for any $\lambda, pn < \lambda < pn + 3$. This proves the lemma.

We now consider the random allocation of n numbered balls to k numbered boxes; ω of the boxes are marked. One can imagine this done by throwing n unnumbered balls into k numbered boxes, ω of which are marked. It is assumed that any ball falls into any box with the same probability. Let $Y_k^{\omega}(n)$ denote the random variable defined as the number of unmarked boxes which receive a ball.

This process is sometimes conveniently represented by a sequence of n trials $\xi_1, \xi_2, \ldots, \xi_n$, where ξ_i is the throw of the i-th ball. Let $\tilde{\xi}_i$ denote the successful outcome of the i-th trial, where success means that the i-th ball

falls into an empty unmarked box (i.e., an unmarked box into which no ball has fallen in the preceding trials $\xi_1, \ldots, \xi_{i-1}$). It is clear that the probability $P(\tilde{\xi}_i)$ of success in the i-th trial depends on the number of successful outcomes in the preceding trials. On the other hand, it is easy to see that for any i

$$\frac{k-\omega-n}{k} < P\{\tilde{\xi}_i\} \leqq \frac{k-\omega}{k}. \tag{6}$$

The random variable $Y_k^\omega(n)$ may also be represented as the number of successful outcomes in the sequence of trials $\xi_1, \xi_2, \ldots, \xi_n$.

Later we shall deal primarily with the case in which the number ω of marked boxes is 0. The random variable $Y_k^0(n)$ will then be denoted by $Y_k(n)$.

Before stating the next lemma, we indicate a certain relation between the random variable $Y_k^\omega(n)$ and the Bernoulli random variable $X(n,p)$:

$$P\left\{X\left(n, \frac{k-\omega}{k}\right) \leqq \lambda\right\} \leqq \leqq P\{Y_k^\omega(n) \leqq \lambda\} < P\left\{X\left(n, \frac{k-\omega-n}{k}\right) \leqq \lambda\right\}, \tag{7}$$

$$P\left\{X\left(n, \frac{k-\omega}{k}\right) \geqq \lambda\right\} \geqq \geqq P\{Y_k^\omega(n) \geqq \lambda\} > P\left\{X\left(n, \frac{k-\omega-n}{k}\right) \geqq \lambda\right\}. \tag{8}$$

These inequalities follow from inequality (6).

LEMMA 10. *For any natural number k, any natural number ω or $\omega = 0$, and any $\nu \geqq 0$, $\delta \geqq 1$ such that $k - \omega - \delta > 0$,*

$$P\left\{Y_k^\omega\left(\left[\frac{(1+\nu)\delta}{k-\omega-\delta}k\right]\right) \leqq \delta\right\} < e^3\left[\frac{(1+\nu)\delta}{k-\omega-\delta}k\right]\left(1-\frac{\nu}{2(1+\nu)}\right)^{\nu\delta-2}.$$

Proof. We first show that

$$P\{Y_k^\omega(n) \leqq \delta\} \leqq P\left\{X\left(n, \frac{k-\omega-\delta}{k}\right) \leqq \delta\right\}. \tag{9}$$

To this end, consider the sequence of trials $\xi_1, \ldots, \xi_n$ appearing in the definition of $Y_k^\omega(n)$, with a different definition of successful outcome:

a) if the number of successful outcomes in the preceding trials is at most δ, the successful outcome $\tilde{\xi}_i$ is the same as before, i.e., the i-th ball falls into an empty unmarked box;

b) if the number of successful outcomes in the preceding trials is greater than δ, the event $\tilde{\xi}_i$ is defined to be any certain event.

We now define a new random variable $Y_k^{\omega,\delta}(n)$ as the number of successful outcomes, by the new definition, in the sequence of trials $\xi_1, \ldots, \xi_n$. It is obvious that for any $\lambda \leqq \delta$

$$P\{Y_k^{\omega}(n) = \lambda\} = P\{Y_k^{\omega,\delta}(n) = \lambda\}$$

and consequently

$$P\{Y_k^{\omega}(n) \leqq \delta\} = P\{Y_k^{\omega,\delta}(n) \leqq \delta\}.$$

On the other hand, it is easy to establish the required relation between the random variable $Y_k^{\omega,\delta}(n)$ and the Bernoulli variable $X(n,p)$. In fact, although the probability $P(\tilde{\xi}_i)$ of successful outcome does depend on the outcomes of the preceding trials,

$$P\{\tilde{\xi}_i\} \geqq \frac{k - \omega - \delta}{k}.$$

Therefore,

$$P\{Y_k^{\omega,\delta}(n) \leqq \delta\} \leqq P\left\{X\left(n, \frac{k - \omega - \delta}{k}\right) \leqq \delta\right\}$$

so that inequality (9) holds.

Let s be an arbitrary real number (not necessarily natural), at least 1, and let $0 \leqq \delta \leqq p[s]$. By Lemma 8,

$$P\{X([s],p) \leqq \delta\} < e^2[s]\left(1 - \frac{p[s] - \delta}{2p[s]}\right)^{p[s]-\delta-1}.$$

Let $ps - \delta > 2$. Then

$$e^2[s]\left(1 - \frac{p[s] - \delta}{2p[s]}\right)^{p[s]-\delta-1} < e^2[s]\left(1 - \frac{ps - \delta}{2ps} + \frac{1}{2s}\right)^{ps-\delta-2} \leqq$$

$$\leqq e^2[s]\left(1 - \frac{ps - \delta}{2ps}\right)^{ps-\delta-2}\left(1 + \frac{1}{ps + \delta}\right)^{ps-\delta-2} <$$

$$< e^2[s]\left(1 - \frac{ps - \delta}{2ps}\right)^{ps-\delta-2} e^{(ps-\delta-2)/(ps+\delta)} < e^3[s]\left(1 - \frac{ps - \delta}{2ps}\right)^{ps-\delta-2}$$

and so

$$P\{X([s],p) \leqq \delta\} < e^3[s]\left(1 - \frac{ps - \delta}{2ps}\right)^{ps-\delta-2}.$$

It is easily seen that this inequality is also valid when $0 \leqq ps - \delta \leqq 2$.

We have thus proved that this inequality holds for any real $s \geqq 1$ and $\delta, 0 \leqq \delta \leqq ps$.

Let k, ω, δ and v satisfy the assumptions of the lemma. Set

$$p = \frac{k - \omega - \delta}{k} \quad \text{and} \quad s = \frac{(1 + v)\delta}{k - \omega - \delta} k.$$

It is not difficult to see that $s \geqq 1$ and $\delta \leqq ps$. Therefore, using the above estimate for $P\{X([s], p) \leqq \delta\}$, we get

$$P\left\{X\left(\left[\frac{(1 + v)\delta}{k - \omega - \delta} k\right], \frac{k - \omega - \delta}{k}\right) \leqq \delta\right\} <$$

$$< e^3 \left[\frac{(1 + v)\delta}{k - \omega - \delta} k\right] \left(1 - \frac{v}{2(1 + v)}\right)^{v\delta - 2}.$$

This inequality and (9) imply the statement of the lemma.

LEMMA 11. *For any natural numbers k and μ, nonnegative number v and x such that $0 < x < k\mu/(\mu + 1) - \mu$,*

$$P\left\{Y_k\left(\left[-(1 + v)k \ln\left(1 - \frac{\mu + 1}{\mu k}(x + \mu)\right)\right]\right) < x\right\} <$$

$$< e^3 \left[-(1 - v)k \ln\left(1 - \frac{\mu + 1}{\mu k}(x + \mu)\right)\right] \left(1 - \frac{v}{2(1 + v)}\right)^{vx/\mu - 2}.$$

Proof. We first prove the following proposition:

C. *For any natural numbers k, μ, δ such that $(\mu + 1)\delta < k$, and any nonnegative number v,*

$$P\left\{Y_k\left(\left[-(1 + v)k \ln\left(1 - \frac{\mu + 1}{k}\delta\right)\right]\right) < \mu\delta\right\} <$$

$$< e^3 \left[-(1 + v)k \ln\left(1 - \frac{\mu + 1}{k}\delta\right)\right] \left(1 - \frac{v}{2(1 + v)}\right)^{v\delta - 2}.$$

Let k, μ, δ, v satisfy the assumptions of Proposition C. Consider arbitrary natural numbers $n_1, n_2, \ldots, n_\mu$, and denote their sum by n_0. We first prove

an inequality:

$$P\{Y_k(n_0) < \mu\delta\} < P\{Y_k^0(n_1) < \delta\} + \\ + P\{Y_k^{\delta}(n_2) < \delta\} + \ldots + P\{Y_k^{(i-1)\delta}(n_i) < \delta\} + \ldots \\ \ldots + P\{Y_k^{(\mu-1)\delta}(n_\mu) < \delta\}. \quad (10)$$

Suppose that n_0 balls are thrown at random into k boxes. This may be done as follows: first throw n_1 balls, then n_2, then n_3, and so on. Let A_i denote the event: After the first $n_1 + n_2 + \ldots + n_{i-1}$ balls have been thrown, the number of occupied boxes is at least $(i-1)\delta$, and after another n_i balls are thrown, the number of occupied boxes is less than $i\delta$. Clearly, if n_0 balls have been thrown and the number of occupied boxes turns out to be less than $\mu\delta$, i.e., $Y_k(n_0) < \mu\delta$, this means that at least one of the events $A_1, A_2, \ldots, A_\mu$ has occurred. Therefore,

$$P\{Y_k(n_0) < \mu\delta\} < P\{A_1\} + P\{A_2\} + \ldots + P\{A_\mu\}.$$

Now, the occurrence of the event A_i implies that if n_i balls are thrown into the boxes, of which $x_i = (i-1)\delta + \varepsilon$ are already occupied (where ε is some natural number), there will be less than $\delta - \varepsilon$ occupied boxes.

Thus, the probability of A_i cannot exceed that of the event $Y_k^{(i-1)\delta+\varepsilon}(n_i) < \delta - \varepsilon$. But, as is easily seen, the probability of $Y_k^{(i-1)\delta+\varepsilon}(n_i) < \delta - \varepsilon$ cannot exceed that of the event $Y_k^{(i-1)\delta}(n_i) < \delta$. Hence $P\{A_i\} < P\{Y_k^{(i-1)\delta}(n_i) < \delta\}$, and this proves inequality (10).

Now set

$$n_1 = \left[\frac{(1+v)\delta}{k-\delta}k\right], \ldots, n_i = \left[\frac{(1+v)\delta}{k-i\delta}k\right], \ldots, n_\mu = \left[\frac{(1+v)\delta}{k-\mu\delta}k\right].$$

Then

$$n_0 = \sum_{i=1}^{\mu} n_i = \sum_{i=1}^{\mu}\left[\frac{(1+v)\delta}{k-i\delta}k\right] \leqq \left[\sum_{i=1}^{\mu}\frac{(1+v)\delta}{k-i\delta}k\right].$$

Since $(\mu+1)\delta < k$, we have $k - i\delta > 0$ for any $1 < i \leqq \mu + 1$. Therefore

$$\sum_{i=1}^{\mu}\frac{(1+v)\delta}{k-i\delta}k < \int_1^{\mu+1}\frac{(1+v)\delta}{k-i\delta}k\,di = \\ = -(1+v)k\ln(k-i\delta)\Big|_1^{\mu+1} < -(1+v)k\ln\left(1-\frac{\mu+1}{k}\delta\right).$$

Since $P\{Y_k(n) < x\} \leqq P\{Y_k(n_0) < x\}$ for $n \geqq n_0$, it follows from inequality (10) that

$$P\left\{Y_k\left(\left[-(1+\nu)k\ln\left(1-\frac{\mu+1}{k}\delta\right)\right]\right) < \mu\delta\right\} \leqq P\{Y_k(n_0) < \mu\delta\} <$$

$$< \sum_{i=1}^{\mu} P\{Y_k^{(i-1)\delta}(n_i) < \delta\} = \sum_{i=1}^{\mu} P\left\{Y_k^{(i-1)\delta}\left(\left[\frac{(1+\nu)\delta}{k-i\delta}k\right]\right) < \delta\right\}.$$

Since $k - i\delta > 0$, the probabilities

$$P\left\{Y_k^{(i-1)\delta}\left(\left[\frac{(1+\nu)\delta}{k-i\delta}k\right]\right) < \delta\right\}$$

may be estimated by an application of Lemma 10. The result is

$$\sum_{i=1}^{\mu} P\left\{Y_k^{(i-1)\delta}\left(\left[\frac{(1+\nu)\delta}{k-i\delta}k\right]\right) < \delta\right\} <$$

$$< \sum_{i=1}^{\mu} e^3\left[\frac{(1+\nu)\delta}{k-i\delta}k\right]\left(1-\frac{\nu}{2(1+\nu)}\right)^{\nu\delta-2} <$$

$$< e^3\left[-(1+\nu)k\ln\left(1-\frac{\mu+1}{k}\delta\right)\right]\left(1-\frac{\nu}{2(1+\nu)}\right)^{\nu\delta-2}.$$

This completes the proof of Proposition C.

We are now ready to prove Lemma 11. Let k, μ, x and ν satisfy the assumptions of Lemma 11. Consider the smallest x' such that $x' > x$ and x'/μ is a natural number; obviously, $x' < x + \mu$. It is easy to see that when $\delta = x'/\mu$ the numbers k, μ, δ, ν satisfy the assumptions of Proposition C. Therefore,

$$P\left\{Y_k\left(\left[-(1+\nu)k\ln\left(1-\frac{\mu+1}{\mu k}x'\right)\right]\right) < x'\right\} <$$

$$< e^3\left[-(1+\nu)k\ln\left(1-\frac{\mu+1}{\mu k}x'\right)\right]\left(1-\frac{\nu}{2(1+\nu)}\right)^{\nu x'/\mu-2} \leqq$$

$$\leqq e^3\left[-(1+\nu)k\ln\ 1-\left(\frac{\mu+1}{\mu k}(x+\mu)\right)\right]\left(1-\frac{\nu}{2(1+\nu)}\right)^{\nu x/\mu-2}.$$

On the other hand, since $x' < x + \mu$,

$$P\left\{Y_k\left(\left[-(1+\nu)k\ln\left(1-\frac{\mu+1}{\mu k}x'\right)\right]\right) < x'\right\} \geqq$$

$$\geqq P\left\{Y_k\left(\left[-(1+\nu)k\ln\left(1-\frac{\mu+1}{\mu k}(x+\mu)\right)\right]\right) < x\right\}.$$

This proves Lemma 11.

LEMMA 12. *Let d and ν be positive numbers and μ a natural number, all arbitrary but fixed. Then, for any sufficiently large natural numbers k and z such that $z \leqq dk$,*

$$P\left\{Y_k(z) < \left(1-\frac{1}{\mu}\right)k\left(1-e^{-z/(1+\nu)k}\right)\right\} < e^3 z\left(1-\frac{\nu}{2(1+\nu)}\right)^{C_{\nu,\mu,d}z},$$

where $C_{\nu,\mu,d}$ is a positive constant depending on ν, μ and d.

Proof. Set

$$z = \left[-(1+\nu)k\ln\left(1-\frac{\mu+1}{\mu k}(x+\mu)\right)\right].$$

Then

$$z \leqq \left[-(1+\nu)k\ln\left(1-\frac{\mu+1}{\mu k}(x+\mu)\right)\right].$$

Isolating x, we get

$$x \geqq \frac{\mu}{\mu+1}k(1-e^{-z/(1+\nu)k}) - \mu.$$

Denote

$$x(z) = \frac{\mu}{\mu+1}k(1-e^{-z/(1+\nu)k}) - \mu.$$

It is not hard to see that for sufficiently large k and z the quantity $x(z)$ satisfies the assumptions of Lemma 11. Substituting z for

$$\left[-(1+\nu)k\ln\left(1-\frac{\mu+1}{\mu k}(x+\mu)\right)\right]$$

and $x(z)$ for x in the statement of Lemma 11, we see that, for sufficiently large k and z,

$$P\left\{Y_k(z) < \frac{\mu}{\mu+1}k(1-e^{-z/(1+\nu)k}) - \mu\right\} <$$

$$< e^3 z\left(1 - \frac{\nu}{2(1+\nu)}\right)^{(\nu/\mu)\{(k\mu/(\mu+1))(1-\exp(-z/(1+\nu)k))-\mu\}-2}$$

Now let $z \leqq dk$. Then, using the fact that $\lim_{s\to 0}\frac{1-e^{-s}}{s} = 1$, it is not difficult to see that for sufficiently large z the right-hand side of the above inequality is smaller than

$$e^3 z\left(1 - \frac{\nu}{2(1+\nu)}\right)^{C_{\nu,\mu,d}z},$$

where $C_{\nu,\mu,d}$ is a positive constant depending on ν, μ and d.

To complete the proof, note that for sufficiently large k and z

$$\frac{\mu}{\mu+1}k\,(1-e^{-z/(1+\nu)k}) - \mu > \left(1-\frac{1}{\mu}\right)k\,(1-e^{-z/(1+\nu)k}).$$

Therefore,

$$P\left\{Y_k(z) < \left(1-\frac{1}{\mu}\right)k\left(1-e^{-z/(1+\nu)k}\right)\right\} \leqq$$

$$\leqq P\left\{Y_k(z) < \frac{\mu}{\mu+1}k\left(1-e^{-z/(1+\nu)k} - \mu\right)\right\} < e^3 z\left(1 - \frac{\nu}{2(1+\nu)}\right)^{C_{\nu,\mu,d}z}.$$

This proves the lemma.

In conclusion, we indicate the intuitive meaning of Lemma 12. It is not difficult to see that for any fixed positive ν the upper bound provided by Lemma 12 for the probability

$$P\left\{Y_k(z) < \left(1-\frac{1}{\mu}\right)k\left(1-e^{-z/(1+\nu)}\right)\right\},$$

that is,

$$e^3 z\left(1 - \frac{\nu}{2(1+\nu)}\right)^{C_{\nu,\mu,d}z}$$

approaches 0 as z approaches infinity. This means that when k and z, $z \leqq dk$, approach infinity, and z balls are thrown at random into k boxes, the number of occupied boxes is at least

$$\left(1 - \frac{1}{\mu}\right) k(1 - e^{-z/(1+\nu)k})$$

with probability 1.

V.7. Proof of the Fundamental Lemma

The stochastic procedure described in Section V.3 for generation of automaton graphs with k vertices will be split into six stages and each of these considered separately.

Before going on to consider these stages, we adopt some conventions. Throughout the sequel, the term stochastic procedure will always refer to the stochastic procedure generating automaton graphs with exactly k vertices. The number of vertices of level h and the number of vertices in the h-base, as given by the stochastic procedure, will be denoted by t_h and D_h, respectively. We shall also say that the h-th level has growth factor a (at most a, at least a) if $t_h = at_{h-1}$ ($t_h \leqq at_{h-1}$, $t_h \geqq at_{h-1}$).

Stage 1 begins at the beginning of the stochastic process and ends (provided the stochastic procedure has not yet ended) at a step h_1 such that $D_{h_1-1} < k^{1/6}$ but $D_{h_1} \geqq k^{1/6}$.

Obviously, the probability that the first stage yields an h-base which differs from a tree at more than one point is not greater than the probability $p(k)$ that the first $]\frac{1}{6}\log_m k[$ steps yield an h-base with a similar property. By Lemma 7, for sufficiently large k the probability $p(k)$ satisfies the inequality $p(k) \leqq k^{-8/7}$. This implies

LEMMA 13. *Let $p_1(k)$ be the probability of the following composite event:* 1) *for all $h \leqq h_1$,*

$$D_h \geqq m^h;$$

2) *the number t_{h_1} of vertices of the last level constructed at the first stage satisfies the inequality*

$$t_{h_1} \geqq \frac{m-1}{m} k^{1/6} \geqq \tfrac{1}{2} k^{1/6}.$$

Then $p_1(k) > 1 - k^{-8/7}$ for sufficiently large k.

Stage 2 begins immediately after Stage 1 and ends (provided the stochastic procedure has not yet ended) at a step h_2 such that $D_{h_2-1} < k/m^4$ but $D_{h_2} \geqq k/m^4$.

Let $t_{h-1} \geqq \frac{1}{2}k^{1/6}$, $D_{h-1} < k/m^4$ and let C be a constant, say $C = 3$. We claim that under these assumptions the h-th level satisfies the inequality

$$P\left\{t_h \geqq m\left(1 - \frac{m+2}{m^4}\right)t_{h-1}\right\} > 1 - \frac{1}{k^C} \tag{11}$$

for sufficiently large k depending on m and C but not on h.

The stochastic procedure generating automaton graphs may be represented by an experiment in which mk balls are thrown into k boxes. More precisely, the free ends of the edges may be regarded as balls, the vertices $q_0, q_1, \ldots, q_{k-1}$ as boxes; a ball falling into an unoccupied box means that the corresponding free end of an edge has been connected to a vertex to which no other edge has yet been connected. Similarly, construction of the h-th level may be represented by an experiment in which mt_{h-1} balls are thrown into k boxes, of which D_{h-1} are marked (the marked boxes correspond to the vertices to which free ends of edges have already been connected). In this interpretation, t_h will correspond to the number of occupied unmarked boxes. Therefore, for fixed D_{h-1} and t_{h-1},

$$P\{t_h = \lambda\} = P\{Y_k^{D_{h-1}}(mt_{h-1}) = \lambda\}$$

and so

$$P\left\{t_h \geqq m\left(1 - \frac{m+2}{m^4}\right)t_{h-1}\right\} = P\left\{Y_k^{D_{h-1}}(mt_{h-1}) \geqq m\left(1 - \frac{m+2}{m^4}\right)t_{h-1}\right\}.$$

Let us estimate the right-hand side of this equality. To this end, we use inequality (8), plus the fact that $D_{h-1} < k/m^4$ and so $t_{h-1} < D_{h-1} < k/m^4$. It follows that

$$\begin{aligned}
P\left\{Y_k^{D_{h-1}}(mt_{h-1}) \geqq m\left(1 - \frac{m+2}{m^4}\right)t_{h-1}\right\} &> \\
> P\left\{X\left(mt_{h-1}, 1 - \frac{D_{h-1} + mt_{h-1}}{k}\right) \geqq m\left(1 - \frac{m+2}{m^4}\right)t_{h-1}\right\} &> \\
> P\left\{X\left(mt_{h-1}, 1 - \frac{m+1}{m^4}\right) \geqq m\left(1 - \frac{m+2}{m^4}\right)t_{h-1}\right\} &= \\
= 1 - P\left\{X\left(mt_{h-1}, 1 - \frac{m+1}{m^4}\right) < m\left(1 - \frac{m+2}{m^4}\right)t_{h-1}\right\}.
\end{aligned}$$

Using Lemma 8, we get

$$P\left\{X\left(mt_{h-1}, 1 - \frac{m+1}{m^4}\right) < m\left(1 - \frac{m+2}{m^4}\right)t_{h-1}\right\} <$$

$$< e^2 mt_{h-1}\left(1 - \frac{1/m^4}{2(1-(m+1)/m^4)}\right)^{(1/m^4)mt_{h-1}-1} =$$

$$= e^2 mt_{h-1}\left(1 - \frac{1}{2(m^4 - m - 1)}\right)^{(1/m^3)t_{h-1}-1}.$$

Since $t_{h-1} \geqq \frac{1}{2}k^{1/6}$, it follows that for any fixed C and for sufficiently large k depending on m and C, the last expression is smaller than $1/k^C$. This proves inequality (11).

Let $h_1 + 1, h_1 + 2, \ldots, h_2$ be the levels constructed at Stage 2. The number of these levels is clearly smaller than k.

Now let k be sufficiently large, say, so large that inequality (11) holds for the C in question.

Suppose that the second assumption of Lemma 13 is valid for Stage 1. Then level h_1 satisfies the assumptions under which inequality (11) holds, and so the $(h_1 + 1)$-th level has growth factor at least $m(1 - (m + 2)/m^4)$ with probability greater than $1 - 1/k^C$. It is also easy to see, on the basis of the above assumption on Stage 1 and using inequality (11), that if the levels $h_1 + 1, h_1 + 2, \ldots, h_1 + i - 1$ (with $h_1 + i - 1 < h_2$) have growth factor at least $m(1 - (m + 2)/m^4)$, then the $(h_1 + i)$-th level will also have growth factor at least $m(1 - (m + 2)/m^4)$, with probability greater than $1 - 1/k^C$. Therefore, if Stage 1 satisfies the second assumption of Lemma 13, then all levels constructed at Stage 2 have growth factor at least $m(1 - (m + 2)/m^4)$ with probability

$$p_2(k) > \left(1 - \frac{1}{k^C}\right)^{h_2 - h_1} > \left(1 - \frac{1}{k^C}\right)^k > 1 - \frac{1}{k^{C-1}}.$$

Together with Lemma 13, this implies the following

LEMMA 14. *Let $p_{1,2}(k)$ denote the probability of the following composite event:*

1) *for any $h \leqq h_2$,*

$$D_h > \left(m\left(1 - \frac{m+2}{m^4}\right)\right)^h;$$

2) *the number* $H_{1,2}$ *of levels constructed at Stages* 1 *and* 2 *satisfies the inequality*

$$H_{1,2} < \log_{m(1-(m+2)/m^4)} k = \frac{1}{1 + \log_m(1 - (m+2)/m^4)} \log_m k.$$

Then, for sufficiently large k,

$$p_{1,2}(k) \geqq p_1(k) \cdot p_2(k) > \left(1 - k^{-8/7}\right)\left(1 - \frac{1}{k^{c-1}}\right) \geqq 1 - k^{-9/8}.$$

Stage 3 begins immediately after Stage 2 and ends (provided the stochastic procedure has not yet ended) at a step h_3 such that

$$D_{h_3-1} < \left(1 - \frac{1}{2^m}\right)k, \qquad \text{but} \qquad D_{h_3} \geqq \left(1 - \frac{1}{2^m}\right)k.$$

Throughout our study of this stage, we shall assume that at each step $h (h = 1, 2, 3, \dots)$ the edges of the $(h-1)$-th level are connected as follows: one first connects (at random) the free ends of the edges issuing from one vertex, then those of the edges issuing from a second vertex, a third vertex, and so on. In other words, each step h is divided into substeps, in each of which only the edges (more precisely, free ends of edges) issuing from a single vertex are connected (there are exactly m such edges). We shall so arrange the substeps that the substep of step 1 comes first, followed by the substeps of step 2, then of step 3, and so on. Let $S(r)$ denote the number of occupied vertices (i.e., vertices to which the free ends of some edges have been connected) after the first r substeps. Obviously,

$$P\{S(r) = \lambda\} = P\{Y_k(mr) = \lambda\}.$$

We shall now study the quantity $S'(r) = S(r) - r$. Intuitively, if step h in some concrete realization of the stochastic procedure ends at the r-th substep, the number of vertices of level h constructed at this step is precisely $S'(r)$.

It follows from the definition of Stage 3 that the substeps to which it corresponds have r-values such that

$$\frac{k}{m^5} < r < \left(1 - \frac{1}{2^m}\right)k. \tag{12}$$

Therefore, if we prove that the probability of the event $S'(r) \geqq \lambda$ is greater than p for all r satisfying (12), it will follow that all levels constructed at Stage 3 have at least λ vertices, with probability greater than p.

Let us estimate the probability $P\{S'(r) \geqq \lambda\}$: we shall show that there is a positive constant C_m (which generally depends on m) such that, for sufficiently large k and any fixed r satisfying (12),

$$P\{S'(r) \geqq C_m k\} > 1 - e^3 m \left(1 - \frac{1}{2^m}\right) k \cdot 0.9996^{Ck/m^4}, \tag{13}$$

where C is a positive constant.

Accordingly, we have

$$P\{S'(r) \geqq C_m k\} = P\{S(r) \geqq r + C_m k\} =$$

$$= P\{Y_k(mr) \geqq r + C_m k\} = 1 - P\{Y_k(mr) < r + C_m k\}.$$

Now set $\mu = 100 \cdot 2^m$, $\nu = 0.001$ and $d = m(1 - 1/2^m)$ in Lemma 12. Let k and z be sufficiently large and $z \leqq m(1 - 1/2^m)k$. Then

$$P\left\{ Y_k(z) < \left(1 - \frac{1}{100 \cdot 2^m}\right) k \ (1 - e^{-0.999\, z/k}) \right\} < e^3 z \cdot 0.9996^{Cz},$$

where C is a positive constant.

Now assume that we have found a positive constant C_m such that, for any r satisfying (12),

$$r + C_m k \leqq \left(1 - \frac{1}{100 \cdot 2^m}\right) k \left(1 - e^{-0.999\, mr/k}\right). \tag{14}$$

Then, obviously, if r satisfies inequality (12),

$$P\{Y_k(mr) < r + C_m k\} \leqq$$

$$\leqq P\left\{Y_k(mr) < \left(1 - \frac{1}{100 \cdot 2^m}\right) k \ (1 - e^{-0.999\, mr/k}) \right\} < e^3 mr \cdot 0.9996^{Cmr} <$$

$$< e^3 m \left(1 - \frac{1}{2^m}\right) k \cdot 0.9996^{Ck/m^4}$$

and, consequently, inequality (13) holds.

Thus, in order to prove inequality (13) it will suffice to prove the following proposition:

A. *There exists a positive constant C_m which satisfies inequality* (14), *whenever r satisfies inequality* (12).

Using the standard methods of the calculus, one readily shows that the expression

$$\left(1 - \frac{1}{100 \cdot 2^m}\right) k \left(1 - e^{-0.999\, mr/k}\right) - r,$$

viewed as a function of r over the interval $k/m^5 \leqq r \leqq (1 - 1/2^m)k$, has a minimum at either $r = k/m^5$ or $r = (1 - 1/2^m)k$. Therefore, Proposition A will be proved if we can show that there exists a positive constant C_m such that

1) $$\left(1 - \frac{1}{100 \cdot 2^m}\right) k \left(1 - e^{-0.999\, mk/km^5}\right) - \frac{k}{m^5} \geqq C_m k,$$

that is,

$$\left(1 - \frac{1}{100 \cdot 2^m}\right)\left(1 - e^{-0.999\, /m^4}\right) - \frac{1}{m^5} \geqq C_m;$$

2) $$\left(1 - \frac{1}{100 \cdot 2^m}\right) k \left(1 - e^{-0.999\,(m/k)(1 - 1/2^m)k}\right) - \left(1 - \frac{1}{2^m}\right) k \geqq C_m k,$$

that is,

$$\frac{1}{2^m} - \left(1 - \frac{1}{100 \cdot 2^m}\right) e^{-0.999m(1 - 1/2^m)} - \frac{1}{100 \cdot 2^m} \geqq C_m.$$

It is not difficult to show (using, e.g., the power-series expansion of the function e^{-x}) that for any fixed $m \geqq 2$ the left-hand sides of these inequalities are positive. Hence, for any fixed m (in our case we always have $m \geqq 2$), there exists a positive constant C_m satisfying these inequalities.

This proves Proposition A. and hence inequality (13)

Let r be a number satisfying (12). It follows from (13) that, for any C', for sufficiently large k depending on m and C' but not on r,

$$P\{S'(r) \geqq C_m k\} > 1 - \frac{1}{k^{C'}} \quad . \tag{13'}$$

Now consider the probability that all r-values corresponding to Stage 3 satisfy the inequality $S'(r) \geqq C_m k$; this is the probability

$$P\left\{\prod_{r = k/m^5}^{(1 - 1/2^m)k} (S'(r) \geqq C_m k)\right\}.$$

Using inequality (13'), we see that for sufficiently large k,

$$P\left\{\prod_{r=k/m^5}^{(1-1/2^m)k}(S'(r) \geqq C_m k)\right\} > 1 - \frac{1}{k^{C'-1}} = 1 - \frac{1}{k^C}.$$

We have thus proved

LEMMA 15. *Let $p_3(k)$ be the probability of the following composite event:*
1) *all levels constructed at Stage* 3 *contain at least $C_m k$ vertices, where $C_m > 0$;*
2) *the number H_3 of levels constructed at Stage* 3 *satisfies the inequality*

$$H_3 < \frac{k}{C_m k} = \frac{1}{C_m}.$$

Then, for any C, $p_3(k) > 1 - 1/k^C$ for sufficiently large k.

Stage 4 begins immediately after Stage 3 and ends (provided the stochastic procedure has not yet ended) at a step h_4 such that $t_{h_4-1} > (\log_2 k)^2$ but $t_{h_4} \leqq (\log_2 k)^2$.

Let h be any step of Stage 4, so that

$$t_{h-1} > (\log_2 k)^2 \qquad \text{and} \qquad D_{h-1} \geqq \left(1 - \frac{1}{2^m}\right)k.$$

We shall show that for any C' and for all sufficiently large k depending on m and C' but not on h,

$$P\left\{t_h \leqq \frac{1.5m}{2^m} t_{h-1}\right\} > 1 - \frac{1}{k^{C'}}. \tag{15}$$

Reasoning exactly as for Stage 2, one easily sees that for fixed D_{h-1} and t_{h-1}

$$P\{t_h = \lambda\} = P\{Y_k^{D_{h-1}}(mt_{h-1}) = \lambda\}$$

and so

$$P\left\{t_h \leqq \frac{1.5m}{2^m} t_{h-1}\right\} = P\left\{Y_k^{D_{h-1}}(mt_{h-1}) \leqq \frac{1.5m}{2^m} t_{h-1}\right\}.$$

We now estimate the right-hand side of this equality. Using inequality (7) and also the fact that $D_{h-1} \geqq (1 - 1/2^m)k$, we get

$$P\left\{Y_k^{D_{h-1}}(mt_{h-1}) \leqq \frac{1.5m}{2^m} t_{h-1}\right\} \geqq P\left\{X\left(mt_{h-1}, \frac{k - D_{h-1}}{k}\right) \leqq\right.$$

$$\leqq \frac{1.5m}{2^m} t_{h-1}\Big\} \geqq P\left\{X\left(mt_{h-1}, \frac{1}{2^m}\right) \leqq \frac{1.5m}{2^m} t_{h-1}\right\} =$$

$$= 1 - P\left\{X\left(mt_{h-1}, \frac{1}{2^m}\right) > \frac{1.5m}{2^m} t_{h-1}\right\}.$$

Now, using Lemma 9, we get

$$P\left\{X\left(mt_{h-1}, \frac{1}{2^m}\right) > \frac{1.5m}{2^m} t_{h-1}\right\} < \frac{mt_{h-1}}{(1.5)^{(0.5m/2\cdot 2^m)t_{h-1}-3/2}}.$$

Consider the right-hand side of this inequality. Since $t_{h-1} > (\log_2 k)^2$, it follows that for any C', for sufficiently large k depending only on m and C', the right-hand side of this inequality is smaller than $1/k^{C'}$. This proves inequality (15).

Let k be so large that inequality (15) holds. It then follows from (15) that the h-th level, constructed at Stage 4, has growth factor at most $1.5m/2^m$, with probability $1 - 1/k^{C'}$. Together with the fact that the number of levels is at most k, this implies that all levels constructed at Stage 4 have growth factor at most $1.5m/2^m$, with probability

$$p_4'(k) > \left(1 - \frac{1}{k^{C'}}\right)^k > 1 - \frac{1}{k^{C'-1}} = 1 - \frac{1}{k^C}.$$

Now, if all the levels have a growth factor with this property, the number H_4 of levels constructed at Stage 4 is less than

$$\log_{2^m/1.5m} k = \frac{\log_2 m}{m - \log_2 1.5m} \log_m k.$$

We have thus proved

LEMMA 16. *For any C and sufficiently large k, the number H_4 of levels constructed at Stage 4 satisfies the inequality*

$$H_4 < \frac{\log_2 m}{m - \log_2 1.5m} \log_m k$$

with probability $p_4(k) > 1 - 1/k^C$.

Stage 5 begins immediately after Stage 4 and ends (provided the stochastic procedure has not yet ended) at a step h_5 such that $t_{h_5-1} > 120 \log_2 k$ but $t_{h_5} \leqq 120 \log_2 k$.

Let h be any step of Stage 5. We shall show that for sufficiently large k,

depending on m but not on h,

$$P\left\{t_h \leqq \tfrac{2}{3}\, t_{h-1}\right\} > 1 - \frac{1}{k^3}. \tag{16}$$

Reasoning exactly as for Stage 4, we get

$$P\left\{t_h \leqq \tfrac{2}{3}\, t_{h-1}\right\} > 1 - P\left\{X\left(mt_{h-1}, \frac{1}{2^m}\right) > \tfrac{2}{3}\, t_{h-1}\right\}.$$

Now, using Lemma 9 and the fact that $m \geqq 2$ and $t_{h-1} > 120 \log_2 k$, we get

$$P\left\{X\left(mt_{h-1}, \frac{1}{2^m}\right) > \tfrac{2}{3}\, t_{h-1}\right\} < \frac{mt_{h-1}}{(2 \cdot 2^m/3m)^{(1/3 - m/2\cdot 2^m)t_{h-1} - 3/2}} \leqq$$

$$\leqq \frac{mt_{h-1}}{(2 \cdot 2^2/3 \cdot 2)^{(1/3 - 2/2\cdot 2^2)t_{h-1} - 3/2}} < \frac{mt_{h-1}}{2^{(1/30)t_{h-1} - 1}} < \frac{mt_{h-1}}{2^{(1/30)120 \log_2 k - 1}} =$$

$$= \frac{mt_{h-1}}{2^{4 \log_2 k - 1}}.$$

Consequently, for sufficiently large k depending on m but not on h,

$$P\left\{X\left(mt_{h-1}, \frac{1}{2^m}\right) > \tfrac{2}{3}\, t_{h-1}\right\} < \frac{1}{k^3}.$$

This proves inequality (16).

Inequality (16) implies that all levels constructed at Stage 5 have growth factor at most 2/3 with probability $p'_5(k) > (1 - 1/k^3)^k > 1 - 1/k^2$. Since Stage 5 begins with a level h for which $t_h \leqq (\log_2 k)^2$, we see that

LEMMA 17. *For sufficiently large k, the number H_5 of levels constructed at Stage 5 satisfies the inequality*

$$H_5 < \log_{3/2} (\log_2 k)^2 < (4 \log_2 m) \log_m \log_2 k$$

with probability $p_4(k) > 1 - 1/k^2$.

Stage 6 begins immediately after Stage 5 and ends with the construction of an automaton graph.

Recall that as long as no level has appeared in which all edges are connected to vertices of previously constructed levels, the stochastic procedure proceeds level by level, i.e., level 1 is constructed first, then level 2, and so on. We call this part of the stochastic procedure its "levelwise" part; the levelwise part of Stage 6 is defined similarly. (Note that the levelwise part of the stochastic procedure includes the random connection of

edges of the last level.) The levelwise part of the procedure after step h is that part of the levelwise part corresponding to steps $h + 1, h + 2, \ldots$

Suppose that the first h_0 steps of the stochastic procedure have already been executed, and let $D_{h_0} = D, t_{h_0} = t$. Then the number of vertices of the levelwise part after step h_0 is a random variable, say $L(D, t)$. Now continue the stochastic procedure to completion. Assume that this gives $L(D, t) = L_0 > \lambda$, where λ is a fixed number. This means that the part in question of the stochastic procedure has involved random connection of free edges issuing from $t + L_0$ vertices (i.e. from the vertices of level h_0 and the following levels) to the vertices $q_0, q_1, \ldots, q_{k-1}$. Moreover, the fact that the levelwise part of the stochastic procedure has broken off when $L(D, t) = L_0$ means that during the random connection of the free ends of the edges issuing from the above $t + L_0$ vertices exactly L_0 edges were connected to vertices to which no free ends had been connected before. Let us assume that in this part of the stochastic procedure, just as in Stage 3, the random connection of edges proceeds by vertices; i.e., first all edges issuing from one vertex are connected, then all those issuing from a second vertex, a third, and so on. Moreover, this is done in such a way that the first vertices are those of level h_0, then the vertices of level $h_0 + 1$, and so on. Consider the first $t + \lambda$ vertices (in the order adopted for random connection of edges). The first t of these are obviously vertices of level h_0, while the remaining λ belong to the following levels (though not all levels, since $\lambda < L_0 = L(D, t)$). Now $L(D, t) = L_0 > \lambda$ by assumption, and so, once the edges issuing from the first $t + \lambda$ vertices are connected, there are more than λ vertices to which no free ends of other edges have ever been connected (otherwise $L(D, t)$ would be equal to λ). Now there are $mt + m\lambda$ such edges; their random connection may be represented by an experiment in which $mt + m\lambda$ balls are thrown at random into k boxes, D of which are marked (the marked boxes correspond to vertices to which free ends of edges have already been connected). Thus, the event $L(D, t) > \lambda$ has probability at most that of the event $Y_k^D(mt + m\lambda) > \lambda$:

$$P\{L(D, t) > \lambda\} \leqq P\{Y_k^D(mt + m\lambda) > \lambda\}. \tag{17}$$

Let h_5 be the step at which Stage 5 ended, so that $h_5 + 1$ is the step at which Stage 6 began. Then

$$D_{h_5} \geqq \left(1 - \frac{1}{2^m}\right) k \qquad \text{and} \qquad t_{h_5} \leqq 120 \log_2 k.$$

We now prove that there exists a positive constant C_0 such that for sufficiently large k

$$P\left\{L(D_{h_5}, t_{h_5}) > \frac{C_0}{m}\log_2 k\right\} < \frac{1}{k^2}. \tag{18}$$

By inequality (17),

$$P\left\{L(D_{h_5}, t_{h_5}) > \frac{C}{m}\log_2 k\right\} \leqq P\left\{Y^{D_{h_5}}(mt_{h_5} + C\log_2 k) > \frac{C}{m}\log_2 k\right\}.$$

Let us estimate the right-hand side of this inequality. Using inequality (8), Lemma 9 and the above estimates for D_{h_5} and t_{h_5}, we get

$$P\left\{Y_k^{D_{h_5}}(mt_{h_5} + C\log_2 k) > \frac{C}{m}\log_2 k\right\} \leqq$$

$$\leqq P\left\{X\left(mt_{h_5} + C\log_2 k, \frac{k - D_{h_5}}{k}\right) \geqq \frac{C}{m}\log_2 k\right\} \leqq$$

$$\leqq P\left\{X\left(m\cdot 120\log_2 k + C\log_2 k, \frac{1}{2^m}\right) \geqq \frac{C}{m}\log_2 k\right\} <$$

$$< \frac{(120m + C)\log_2 k}{(C\cdot 2^m/m(120m + C))^{(1/2)(C/m - (120m+C)/2^m)\log_2 k - 3/2}} =$$

$$= \frac{(120m + C)\log_2 k}{2^{(1/2)(m - \log_2(m(120m+C)/C)(C/m - (120m+C)/2^m)\log_2 k - (3/2)(m - \log_2(m(120m+C)/C))}}.$$

Consider the expression

$$\left(m - \log_2\frac{m(120m + C)}{C}\right)\left(\frac{C}{m} - \frac{120m + C}{2^m}\right) =$$

$$= C + \frac{120m + C}{2^m}\log_2\frac{m(120m + C)}{C} - m\frac{120m + C}{2^m} -$$

$$- \frac{C}{m}\log_2\frac{m(120m + C)}{C}.$$

All its terms except the first approach zero as m increases. It follows quite easily that there exists C_0 such that when $C = C_0$ and $m \geqq 2$ this expression does not exceed 5.

It follows that for $C = C_0$, when k is sufficiently large,

$$P\left\{Y_k^{D_{h_5}}(mt_{h_5} + C\log_2 k) > \frac{C}{m}\log_2 k\right\} < \frac{1}{k^2}$$

and this proves inequality (18).

Now, note that the number of levels constructed at Stage 6 cannot be greater than $L(D_{h_5}, t_{h_5})$ (since each level contains at least one vertex). Inequality (18) thus implies the following

LEMMA 18. *For sufficiently large k the number H_6 of levels constructed at Stage 6 satisfies the inequality*

$$H_6 \leqq \frac{C_0}{m} \log_2 k = \frac{C_0 \log_2 m}{m} \log_m k$$

with probability $p_6(k) > 1 - 1/k^2$.

This concludes our levelwise investigation of the stochastic procedure. Its results are summarized in Lemmas 14, 15, 16, 17 and 18. Since the height $H(k)$ of the automaton graph generated by this procedure is $H_{1,2} + H_3 + H_4 + H_5 + H_6$, these lemmas directly imply that, for sufficiently large k, the inequality

$$H(k) < \frac{1}{1 + \log_m(1 - (m+2)/m^4))} \log_m k + \frac{1}{C_m} + \frac{\log_2 m}{m - \log_2 1.5m} \log_m k + (4 \log_2 m) \log_m \log_2 k + \frac{C_0 \log_2 m}{m} \log_m k$$

holds with probability

$$p(k) > p_{1,2}(k) \cdot p_3(k) \cdot p_4(k) \cdot p_5(k) \cdot p_6(k) > 1 - k^{-10/9}.$$

Note that, for sufficiently large k,

$$\frac{1}{C_m} + (4 \log_2 m) \log_m \log_2 k < \frac{1}{m} \log_m k.$$

Set

$$C = \frac{1}{1 + \log_m(1 - (m+2)/m^4)} + \frac{\log_2 m}{m - \log_2 1.5m} + \frac{C_0 \log_2 m}{m} + \frac{1}{m}.$$

It follows from the foregoing that, for sufficiently large k,

$$P\{H(k) < C \log_m k\} > 1 - k^{-10/9}$$

and so

$$P\{H(k) > C \log_m k\} < k^{-10/9}.$$

It is not difficult to see that for any $m \geqq 2$ the constant C is well defined and approaches unity as m increases.

This completes the proof of the fundamental lemma.

V.8. Statistical estimate from below for the height of automaton graphs

In this section we shall derive a statistical estimate from below for the height; for large values of m, this estimate approaches the upper bound for the diameter given by Theorem 5.5. Since any lower bound for the height is also a lower bound for the diameter, this will imply that our upper statistical estimate for the diameter is almost best possible for large m.

THEOREM 5.6. *Almost all automaton graphs with k vertices have height greater than* $\log_m k - 2$.

Proof. It will suffice to prove that the stochastic procedure generating automaton graphs with k vertices yields automaton graphs with $H(k) > > \log_m k - 2$, with probability $p \to 1$ as $k \to \infty$.

The first assertions of Lemmas 14 and 15 imply that, with probability $p \to 1$ as $k \to \infty$, the stochastic procedure will not break off during the first three stages. But Stage 3 ends with the construction of an h-base containing at least $(1 - 1/2^m)$ vertices. Therefore, the number of levels $H(k)$ satisfies the inequality

$$1 + m + m^2 + \ldots + m^{H(k)} = \frac{m^{H(k)+1} - 1}{m - 1} \geqq \left(1 - \frac{1}{2^m}\right) k,$$

with probability $p \to 1$ as $k \to \infty$, i.e.,

$$H(k) > \log_m k - 2.$$

Q.E.D.

V.9. Statistical estimate for accessibility spectrum, degree of accessibility and degree of reconstructibility of automata

As we have noted before, automaton graphs with equal numbers of vertices give rise to equal numbers of pairwise nonidentical automata (derived from the former by assigning output labels to the edges). Our statistical estimates for accessibility spectrum and height, which were derived for automaton graphs, therefore carry over directly to automata.

Thus, Theorem 5.4 implies

THEOREM 5.7. *There exist positive constants C_1 and C_2 such that almost all automata $\mathfrak{M}$ with k states possess the following property: For any state q_i of $\mathfrak{M}$, if $l \leqq C_1 \log_m k$ the accessibility spectrum $D_{\langle \mathfrak{M}, q_i \rangle}(l)$ satisfies the*

inequality

$$D_{\langle \mathfrak{M}, q_i \rangle}(l) \geqq m^{C_2 l}.$$

Since the degree of accessibility and the absolute degree of accessibility of an automaton derived from a graph G coincide with the latter's height and diameter, respectively, Theorems 5.5 and 5.6 imply

THEOREM 5.8. *Almost all automata $\mathfrak{M}$ with k states have absolute degree of accessibility $\delta^*(\mathfrak{M})$ at most $C \log_m k$, where C is a constant that depends on m and approaches unity as m approaches infinity.*

THEOREM 5.9. *Almost all automata $\mathfrak{M}$ with k states have degree of accessibility $\delta(\mathfrak{M}, q_0)$ greater than $\log_m k - 2$.*

Recall that by Theorem 2.14 any automaton in which k states are accessible from q_0 has degree of accessibility at least $\log_m k - 1$.

Now consider the absolute degree of reconstructibility. As mentioned in Section IV.2, this parameter satisfies the inequality

$$B^*(\mathfrak{M}) \leqq \delta^*(\mathfrak{M}) + \rho(\mathfrak{M}) + 1.$$

Theorems 5.2 and 5.8 thus imply

THEOREM 5.10. *There exists a constant C_0 such that almost all automata with k states have absolute degree of reconstructibility $B^*(\mathfrak{M})$ at most $C_0 \log_m k$.*

The estimates of Theorems 5.8 and 5.10 obviously remain valid if the absolute parameters $\delta^*(\mathfrak{M})$ and $B^*(\mathfrak{M})$ are replaced by the ordinary degree of accessibility $\delta(\mathfrak{M}, q_0)$ and degree of reconstructibility $B(\mathfrak{M}, q_0)$.

Notes

Theorems 5.1, 5.2, 5.3 and 5.4 are apparently presented here for the first time (the proofs are due to Barzdin').

Statistical estimates for the diameter (Theorems 5.5, 5.6, 5.8 and 5.9) may already be found in the joint paper of Barzdin' and Korshunov [17]. A statistical estimate for the degree of distinguishability (in the nonuniform case) is given in Korshunov [37]. These estimates directly imply Theorem 5.10, on the absolute degree of reconstructibility for almost all automata.

REFERENCES

BOOKS

1. Automata (collection of articles), IL, Moscow 1956. [Russian translation, with additions, of: Shannon C. E. and J. McCarthy (editors). Automata Studies (Annals of Mathematics Studies, No. 34), Princeton University Press 1956.]
2. Aizerman M. A., L. A. Gusev, L. I. Rozonoer, I. M. Smirnov and A. A. Tal'. Logic, Automata, Algorithms, Fizmatgiz, Moscow 1963.
3. Gavrilov M. A. Theory of Relay Switching Circuits, publ. by AN SSSR, Moscow 1950.
4. Gill A. Introduction to the Theory of Finite-State Machines, McGraw-Hill Book Co., New York 1962.
5. Gladkii A. V. Lectures on Mathematical Linguistics for Students at Novosibirsk State University, Novosibirsk 1966.
6. Glushkov V. M. Synthesis of Digital Automata, Fizmatgiz, Moscow 1962.
7. Kobrinskii N. E. and B. A. Trakhtenbrot. Introduction to the Theory of Finite Automata, Fizmatgiz, Moscow 1962. [English translation: North-Holland Publ. Co., Amsterdam 1965.]
8. McKinsey J. C. C. Introduction to the Theory of Games, McGraw-Hill Book Co., New York 1952.
9. Mal'tsev A. I. Algorithms and Recursive Functions, Nauka, Moscow 1965.
10. Novikov P. S. Elements of Mathematical Logic, Fizmatgiz, Moscow 1959.
11. Trakhtenbrot B. A. Complexity of Algorithms and Computations, Novosibirsk 1967.
12. Ginsburg S. An Introduction to Mathematical Machine Theory, Addison-Wesley Company, Reading, Mass. 1962.
13. Harrison M. A. Introduction to Switching and Automata Theory, McGraw-Hill Book Co., New York 1965.
14. Hartmanis J. and E. E. Stearns. Algebraic Structure Theory of Sequential Machines, Prentice-Hall, Englewood Cliffs, N. Y. 1966.

ARTICLES

15. Barzdin' Ya. M. On identification of automata, Problemy Kibernet. (1969), No. 21, 103–114.
16. Barzdin' Ya. M. Capacity of a medium and behavior of automata, DAN SSSR (1965), **160**, No. 2, 302–305.
17. Barzdin' Ya. M. and A. D. Korshunov. On the diameter of reduced automata, Diskretnyi Analiz (1967), No. 9, 3–45.

18. Blokh A. Sh. On problems solvable by sequential machines, Problemy Kibernet. (1960), No. 3, 81–88.
19. Bodnarchuk V. G. On events representable in a finite automaton with a single state, Ukrain. Mat. Zh. (1962), **14**, No. 2, 190–191.
20. Bodnarchuk V. G. Automata and events, Ukrain. Mat. Zh. (1962), **14**, No. 4, 351–361.
21. Bodnarchuk V. G. Systems of equations in the algebra of events, Zh. Vychisl. Mat. i Mat. Fiz. (1963), **3**, No. 6, 1077–1088.
22. Borodyanskii Yu. M. Experiments with finite Moore automata, Kibernetika (1965), No. 6, 18–27.
23. Bukharaev R. G. Some equivalences in the theory of probabilistic automata, "Stochastic Methods and Cybernetics," Uchen. Zap. Kazan. Univ. (1964), **124**, No. 2, 45–65.
24. Glebskii, Yu. V. Realizable sequences in finite automata, Problemy Kibernet. (1961), No. 5, 279–282.
25. Grinberg V. S. and A. D. Korshunov. On the asymptotic behavior of the maximum weight of a finite tree, Problemy Peredachi Informatsii (1966), **2**, No. 1, 96–99.
26. Grinberg V. S. Determinization of systems of graphs and synthesis of finite automata, Sibirsk. Mat. Zh. (1966), **7**, No. 6, 1259–1267.
27. Grinberg V. S. Some new estimates in the theory of finite automata, DAN SSSR (1966), **166**, No. 5, 1066–1068.
28. Dushskii V. A. On characteristic experiments with automata, Problemy Kibernet. (1964), No. 11, 257–262.
29. Ershov Yu. L. On a conjecture of V. A. Uspenskii, Algebra i Logika (Seminar) (1962), **1**, No. 4, 45–48.
30. Kapitonova Yu. V. On isomorphism of abstract automata, I, Kibernetika (1965), No. 3, 25–28.
31. Kapitonova Yu. V. On isomorphism of abstract automata, II, Kibernetika (1965), No. 5, 10–13.
32. Karatsuba A. A. Solution of a problem in the theory of finite automata, UMN (1960), **15**, No. 3, 157–159.
33. Kozmidiadi V. A. On sets which are decidable and enumerable by automata, in: Problems of Logic, pp. 102–115, publ. by AN SSSR, Moscow 1963.
34. Korpelevich G. M. On the relationship between the concepts of decidability and enumerability for finite automata, DAN SSSR (1962), **149**, No. 5, 1023–1025.
35. Korshunov A. D. On asymptotic estimates for the number of reduced finite automata, Diskretnyi Analiz (1966), No. 6, 35–50.
36. Korshunov A. D. On asymptotic estimates for the number of finite automata, Kibernetika (1967), No. 2, 12–19.
37. Korshunov A. D. On the degree of distinguishability of finite automata, Diskretnyi Analiz (1967), No. 10, 39–59.
38. Korshunov A. D. On an upper estimate for the lengths of shortest uniform experiments for terminal-state identification of almost all automata, DAN SSSR (1969), **184**, No. 1, 28–29.
39. Kuznetsov A. V. and B. A. Trakhtenbrot. Investigation of partial-recursive operators by techniques of Baire spaces, DAN SSSR (1955), **105**, No. 5, 897–900.
40. Levenshtein V. I. On inversion of finite automata, DAN SSSR (1962), **147**, No. 6, 1300–1303.

41. Levenshtein V. I. On some encoding properties and adaptive automata for encoding of messages, Problemy Kibernet. (1964), No. 11, 63–121.
42. Lupanov O. B. On comparison of two types of finite sources, Problemy Kibernet. (1963), No. 9, 321–326.
43. Lyubich Yu. I. Estimates for the number of states arising in determinization of a nondeterministic autonomous automaton, DAN SSSR (1964), **155**, No. 1, 41–43.
44. Lyubich Yu. I. Estimates for optimal determinization of nondeterministic autonomous automata, Sibirsk. Mat. Zh. (1964), **5**, No. 2, 337–355.
45. Lyubich Yu. I. and E. M. Livshits. Estimates for the weight of a regular event over a one-letter alphabet, Sibirsk. Mat. Zh. (1965), **6**, No. 1, 122–126.
46. Lyubich Yu. I. On periodicity properties of events representable in finite automata, Ukrain. Mat. Zh. (1964), **16**, No. 3, 396–402.
47. Martynyuk V. V. Relationship between memory and some possibilities of a finite automaton, Problemy Kibernet. (1961), No. 5, 87–96.
48. Medvedev Yu. T. On a class of events representable in a finite automaton, in: [1], pp. 385–401.
49. Muchnik A. A. On the length of an experiment determining the structure of a finite strongly-connected automaton, Problemy Kibernet. (1968), No. 20, 159–171.
50. Plesnevich G. S. Estimates for mean computation time for uniform one-way iterative systems, DAN SSSR (1966), **171**, No. 3, 537–540.
51. Red'ko V. N. On a determining family of relations for the algebra of regular events, Ukrain. Mat. Zh. (1964), **16**, No. 1, 120–126.
52. Sorkin Yu. I. Algorithmic decidability of the isomorphism problem for automata, DAN SSSR (1961), **137**, No. 4, 804–806.
53. Spivak M. A. Representation of automata transformations of regular expressions, Kibernetika (1965), No. 6, 15–17.
54. Spivak M. A. Some properties of the set of experiments of an automaton, Kibernetika (1966), No. 6, 1–7.
55. Tal' A. A. A questionnaire language and abstract synthesis of minimal sequential machines, Avtomat. i Telemekh. (1964), **25**, No. 6, 946–962.
56. Trakhtenbrot B. A. On operators realizable in logical nets, DAN SSSR (1957), **112**, No. 6, 1005–1007.
57. Trakhtenbrot B. A. Synthesis of logical nets whose operators are described in terms of monadic predicates, DAN SSSR (1958), **118**, No. 4, 646–649.
58. Trakhtenbrot B. A. Some constructions in the monadic predicate calculus, DAN SSSR (1961), **138**, No. 2, 320–321.
59. Trakhtenbrot B. A. Finite automata and the monadic predicate calculus, DAN SSSR (1961), **140**, No. 2, 326–329.
60. Trakhtenbrot B. A. Finite automata and the monadic predicate calculus, Sibirsk. Mat. Zh. (1962), **3**, No. 1, 103–131.
61. Trakhtenbrot B. A. Finite automata, Proceedings of Fourth All-Union Mathematics Conference (1961), Vol. 2, pp. 93–101, Nauka, Moscow 1964.
62. Trakhtenbrot B. A. On an estimate for the weight of a finite tree, Sibirsk. Mat. Zh. (1964), **5**, No. 1, 186–191.
63. Shestakov V. I. On transformation of a monocyclic sequence into a reccurrent one, DAN SSSR (1954), **98**, 541–544.

64. Yablonskii S. V. Basic concepts of cybernetics, Problemy Kibernet. (1959), No. 2, 7–38.
65. Yanov Yu. I. On invariant operations on events, Problemy Kibernet. (1964), No. 12, 253–258.
66. Yanov Yu. I. On some subalgebras of events which have no finite complete systems of identities, Problemy Kibernet. (1966), No. 17, 255–258.
67. Yanov Yu. I. On identical transformations of regular expressions, DAN SSSR (1962), **147**, No. 2, 327–330.
68. Aufenkamp D. D. Analysis of sequential machines, IRE Trans. Electronic Computers (1958), EC-**7**, No. 4, 233–306.
69. Aufenkamp D. D. and F. E. Hohn. Analysis of sequential machines, IRE Trans. Electronic Computers (1957), EC-**6**, No. 4, 276–285.
70. Bar-Hillel Y., M. Perles and E. Shamir. On formal properties of simple phrase-structure grammars, Z. Phonetik Sprachwiss. Kommunikat. (1961), **14**, 143–172.
71. Barnes B. H. and J. M. Fitzgerald. Minimal experiments for input-independent machines, J. Assoc. Comput. Mach. (1967), **14**, No. 4, 683–686.
72. Brzozowski J. A. A survey of regular expressions and their applications, IRE Trans. Electronic Computers (1962), EC-**11**, No. 3, 324–335.
73. Büchi J. R. Weak second-order arithmetic and finite automata, Z. Math. Logik Grundlagen Math. (1960), **6**, No. 1, 66–92.
74. Büchi J. R. On a decision method in restricted second-order arithmetic, in: Proceedings of the 1960 International Congress on Logic, Methodology and Philosophy of Science, pp. 1–11, Stanford University Press 1962.
75. Büchi J. R. Decision methods in the theory of ordinals, Bull. Amer. Math. Soc. (1965), **71**, No. 5, 767–769.
76. Büchi J. R., C. C. Elgot and J. B. Wright. The non-existence of certain algorithms in finite automata theory (Abstract), Notices Amer. Math. Soc. (1958), **5**, 98.
77. Burks A. W. The logic of fixed and growing automata, Internat. Symp. Theory of Switching (Proc. 1957), Vol. 29, Part 1, 147–188.
78. Burks A. W. Computation, behaviour and structure in fixed and growing automata, Behavioral Sci. (1961), **6**, No. 1, 5–22.
79. Burks A. W. and J. B. Wright. Theory of logical nets, Proc. IRE (1953), **41**, No. 10, 1357–1365.
80. Burks A. W. and J. B. Wright. Sequence generators, graphs and formal languages, Information and Control (1963), **5**, No. 3, 204–212.
81. Carlyle J. W. Reduced forms for stochastic sequential machines, J. Math. Anal. Appl. (1963), **7**, No. 2, 167–175.
82. Chomsky N. and G. A. Miller. Finite state languages, Information and Control (1958), **11**, No. 2, 91–112.
83. Chomsky N. Formal properties of grammars, in: Handbook of Mathematical Psychology, Vol. 2, pp. 328–418, Wiley, New York 1963.
84. Church A. Applications of recursive arithmetic to the problem of circuit synthesis, in: Summaries of the Summer Institute for Symbolic Logic, pp. 3–50, Cornell University, Ithaca, N. Y. 1957.
85. Church A. Logic, arithmetic and automata, in: Proceedings of Internat. Congress of Mathematicians, pp. 23–35, 1962.

86. Cole S. N. Real-time computation by iterative arrays of finite-state machines, Doctoral dissertation presented to Harvard University, 1964.
87. Copi I. M., C. C. Elgot and J. B. Wright. Realization of events by logical nets, J. Assoc. Comput. Mach. (1958), **5**, No. 2, 181–196.
88. Dejean F. and M. P. Schützenberger. On a question of Eggan, Information and Control (1966), **9**, No. 1, 23–24.
89. Eggan L. C. Transition graphs and the starheight of regular events, Michigan Math. J. (1963), **10**, 385–397.
90. Elgot C. C. Decision problems of finite automata design and related arithmetics, Trans. Amer. Math. Soc. (1961), **98**, No. 1, 21–52.
91. Elgot C. C. and J. B. Wright. Quantifier elimination in a problem of logical design, Michigan Math. J. (1959), **6**, No. 1, 65–69.
92. Ginsburg S. On the length of the smallest uniform experiment which distinguishes the terminal states of a machine, J. Assoc. Comput. Mach. (1958), **5**, No. 3, 266–280.
93. Ginsburg S. Compatibility of states in input-independent machines, J. Assoc. Comput. Mach. (1961), **8**, No. 3, 400–403.
94. Ginsburg S. and E. H. Spanier. Bounded regular sets, Proc. Amer. Math. Soc. (1966), **117**, No. 5, 1043–1049.
95. Harrison M. A. A census of finite automata, Canad. J. Math. (1965), **17**, No. 1, 100–113.
96. Harrison M. A. A census of finite automata, Proceedings of IEEE Symposium on Switching Circuit Theory and Logical Design, pp. 44–46, New York 1964.
97. Harrison M. A. On asymptotic estimates in switching and automata theory, J. Assoc. Comput. Mach. (1966), **13**, No. 1, 151–157.
98. Hibbard T. H. Least upper bounds on minimal terminal state experiments for two classes of sequential machines, J. Assoc. Comput. Mach. (1961), **8**, No. 4, 601–612.
99. Holland J. Survey of automata theory, Memorandum of Project Michigan, University of Michigan, Ann Arbor, Mich. 1959.
100. Kleene S. C. Representation of events in nerve nets, in [1] [English original, pp. 3–41].
101. McCulloch W. and W. Pitts. A logical calculus of the ideas immanent in neuron activity, Bull. Math. Biophys. (1943), **5**, 115–133.
102. McNaughton R. Testing and generating infinite sequences by a finite automaton, Information and Control (1966), **9**, No. 5, 521–530.
103. McNaughton R. Techniques for manipulating regular expressions (Presented to Conference on Systems and Computer Science, University of Western Ontario, London, Ontario, Canada, September 1965).
104. McNaughton R. The theory of automata, a survey, in: Advances in Computers, Vol. 2, pp. 379–421, Academic Press, New York-London 1961.
105. McNaughton R. Finite-state infinite games, MIT, Project MAC, Sept. 1965.
106. McNaughton R. The loop complexity of regular events (Presented to the 1965 Symposium on Logic, Computability and Automata, Rome, New York 1965).
107. Mealy G. H. A method for synthesizing sequential circuits, Bell System Tech. J. (1955), **34**, 1045–1079.
108. Moore E. F. Gedanken-experiments on sequential machines, in [1] [English original, pp. 129–153].
109. Muller D. E. Infinite sequences and finite machines, Proceedings of IEEE Symposium on Switching Circuit Theory and Logical Design, pp. 3–16, New York 1963.

110. Myhill J. Finite automata and the representation of events, WADC Technical Report 57–624, pp. 112–137, 1957.
111. Nerode A. Linear automaton transformations, Proc. Amer. Math. Soc. (1958), **9**, 541–544.
112. Perles M., M. O. Rabin and E. Shamir. The theory of definite automata, IRE Trans. Electronic Computers (1963), EC-**12**, No. 3, 233–243.
113. Rabin M. O. Probabilistic automata, Information and Control (1963), **6**, No. 3, 230–245.
114. Rabin M. O. and D. Scott. Finite automata and their decision problems, IBM J. Res. Develop. (1959), **3**, No. 2, 114–125.
115. Radke C. E. Enumeration of strongly connected sequential machines, Information and Control (1965), **8**, No. 4, 377–389.
116. Ramsey F. P. On a problem of formal logic, Proc. London Math. Soc. (2) (1929), **30**, 264–286.
117. Raney G. W. Sequential functions, J. Assoc. Comput. Mach. (1958), **5**, No. 2, 177–180.
118. Robinson R. M. Restricted set-theoretical definitions in arithmetic, Proc. Amer. Math. Soc. (1958), **9**, No. 2, 238–242.
119. Salomaa A. Two complete axiom systems for the algebra of regular events, J. Assoc. Comput. Mach. (1966), **13**, No. 1, 158–169.
120. Salomaa A. Axiomatization of an algebra of events realizable by logical nets, Problemy Kibernet. (1966), No. 17, 237–246.
121. Salomaa A. On probabilistic automata with one input letter, Ann. Univ. Turku. Ser. A I (1965), No. 85, 3–16.
122. Shepherdson J. C. The reduction of two-way automata to one-way automata, IBM J. Res. Develop. (1959), **3**, No. 2, 198–200.
123. Starke P. H. Theory of stochastic automata, Kybernetika (1966), **2**, No. 6, 475–482.
124. Stearns R. and J. Hartmanis. Regularity-preserving modifications of regular expressions, Information and Control (1963), **6**, No. 1, 55–69.
125. Thatcher J. W. Notes on mathematical automata theory, Univ. Mich. Coll. Liter. Sci. and Arts Commun. Sci. Program, Technical Note, Ann Arbor, Mich. 1963.
126. Turing A. M. On computable numbers with an application to the Entscheidungsproblem, Proc. London Math. Soc. (2) (1937), **42**, 230–265.
127. Landweber L. H. Synthesis algorithms for sequential machines and definability in monadic second order arithmetic, Ph.D. Thesis, Purdue Univ., August 1967.
128. Büchi J. R. and L. H. Landweber. Solving sequential conditions by finite-state strategies, Trans. Amer. Math. Soc. (1969), **138**, 295–311.
129. Büchi J. R. and L. H. Landweber. Definability in the monadic second-order theory of successor, J. Symbolic Logic (1969), **34**, 166–170.
130. Landweber L. H. Decision problems for ω-automata, Math. Systems Theory (1969), **3**, 376–384.
131. Landweber L. H. Synthesis algorithms for sequential machines, Proc. IFIP Congress 68, A 146-150, Edinburgh 1968.

SUBJECT INDEX