OUKA
The University of Osaka
Institutional Knowledge Archive

Title	COMPUTATIONAL COMPLEXITY IN ONE- AND TWO-DIMENSIONAL TAPE AUTOMATA
Author(s)	森田, 憲一
Citation	大阪大学, 1978, 博士論文
Version Type	VoR
URL	https://hdl.handle.net/11094/678
rights	
Note	

The University of Osaka Institutional Knowledge Archive : OUKA

https://ir.library.osaka-u.ac.jp/

The University of Osaka

COMPUTATIONAL COMPLEXITY IN ONE- AND TWO-DIMENSIONAL TAPE AUTOMATA

by

KENICHI MORITA

DOCTORAL THESIS, OSAKA UNIVERSITY

February, 1978

PREFACE

This thesis is a study of computational complexity in various one- and two-dimensional tape automata. Here, we consider how the computational powers of these automata are characterized by an amount of memory (tape complexity) or some other related measures.

This study is motivated by the theory of a tape-bounded (one-dimensional) Turing machine proposed by Stearns et al. In chapter 1, we survey this theory and give a general introduction to the studies of computational complexity in automata theory.

One of the main purposes of this study is to introduce the notion of computational complexity to the field of two-dimensional information processing. So far, there have been several (but not many) researches which deal with the two-dimensional information processing, such as pattern recognition, from the viewpoint of automata theory. However, systematic studies of computational complexity of two-dimensional pattern processing have not been made at all. In chapter 2, a tape-bounded two-dimensional Turing machine is proposed to formalize the notion of tape complexity for the two-dimensional case, and its acceptabilities of a set of patterns are investigated. Here, the hierarchy theorem, the theorem concerning the lower bounds on tape complexity, and some other basic properties are derived.

In chapter 3, various two-dimensional tape automata are

considered. It is shown that the acceptabilities of all these automata can be measured by the tape complexity. Thus the relations of acceptabilities among them can be systematically derived from the hierarchy theorem of two-dimensional Turing machine.

In chapter 4, several automata of tape complexity log n, which accept one-dimensional languages, are investigated. The class of tape complexity log n is very interesting, because there are many concrete models in this class, and thus their computations are intuitive. Furthermore, this class is closely related to a kind of multi-dimensional tape automaton. We newly define an n-bounded multi-counter automaton and a multi-dimensional rebound automaton, and study how their language acceptabilities form a subhierarchy in the class of tape complexity log n.

In chapter 5, several classes of transducers are defined, and their computing abilities of number-theoretic functions are investigated. A tape-bounded Turing transducer, a multi-head transducer, a multi-counter transducer, and some other models are proposed. Here, we also study how their computing abilities vary with the auxiliary memory.

February, 1978

Kenichi Morita

ACKNOWLEDGEMENT

I would like to express my gratitude to the members of my doctoral committee, especially to Professor Tadao Kasami for his useful comments.

I wish to express my deep thanks especially to Dr. Kazuhiro Sugata for his valuable guidances and helpful discussions. I am also greatly indebted to Dr. Hiroshi Tamura who provided constant encouragements and useful suggestions.

I am much indebted to my collaborator Mr. Hiroshi Umeo for his useful discussions. I also thank Mr. Hiroyuki Ebi who helped this study.

Finally I wish to thank the staff and the students of Assistant Professor Tamura's laboratory for their daily discussions.

CONTENTS

GLOSSARY

Term	Meaning	Page
$A \subseteq B$	A is a subset of B	
$A \subsetneq B$	A is a proper subset of B	
$A \not\subset B$	A is not a subset of B	
$A \not\subset\!\not\supset B$	Two sets A and B are incomparable (i.e. $A \not\subset B$ and $A \not\supset B$)	
$A \cup B$	The union of sets A and B	
$A \cap B$	The intersection of sets A and B	
$\bar{A}$	The complement of a set A	
$A \times B$	Cartesian product of sets A and B	
$a \in A$	a is an element of A	
$a \notin A$	a is not an element of A	
$\mathcal{A}^f$	A class of automata $\mathcal{A}$ such that their inputs are restricted to Σ_f^{2+}	16,82
$\mathcal{A}^1$	A class of automata $\mathcal{A}$ such that the number of input symbols is restricted to one	17,98
C_Σ^x	The set of all chunks of sidelength x over a set Σ	30
ϵ	The string of length 0	5
$\emptyset$	The empty set	
$f : A \to B$	f is a mapping from a set A into a set B	
$\mathcal{F}[\mathcal{T}]$	The set of recursive functions computed by a class of transducers $\mathcal{T}$	133
$\mathcal{L}[\mathcal{A}]$	The set of languages accepted by a class of automata $\mathcal{A}$	5,15,98
$\mathcal{L}_0$	The class of type 0 (recursively enumerable) languages	98

Term	Meaning	Page
$\mathcal{L}_1$	The class of type 1 (context-sensitive) languages	98
$\mathcal{L}_2$	The class of type 2 (context-free) languages	98
$\mathcal{L}_3$	The class of type 3 (regular) languages	98
MPSA	The class of deterministic multi-scanner parallel-sequential array automata	67-69
N	The set of natural numbers	
nbc	An n-bounded counter	94
OMPSA	The class of deterministic orthogonal multi-scanner parallel-sequential array automata	69-71
O2PSA	The class of deterministic orthogonal two-scanner parallel-sequential array automata	71
1BCA	The class of deterministic one-dimensional n-bounded multi-counter automata	93-94
1MCA	The class of deterministic one-dimensional multi-counter automata	92-93
1MHA	The class of deterministic one-dimensional multi-head automata	91
1MMA	The class of deterministic one-dimensional multi-marker automata	91
1TM(L(n))	The class of deterministic L(n) tape-bounded one-dimensional Turing machines	3-5,90-91
1WFST	The class of deterministic one-way finite-state transducers	143
1WMCT	The class of deterministic one-way multi-counter transducers	143-144
1WMHT	The class of deterministic one-way multi-head transducers	142-143
PSA	The class of deterministic parallel-sequential array automata	65-67

Term	Meaning	Page
Q1WMHT	The class of quasi-deterministic one-way multi-head transducers	150-151
RA	The class of deterministic multi-dimensional rebound automata	95-96
$\widehat{RA}(2)$	The class of deterministic two-dimensional rebound automata with variant input tapes	111-112
$\mathbf{R}_+$	The set of nonnegative real numbers	
Σ^*	The set of all strings over Σ	5
Σ^+	The set of all nonempty strings over Σ	5
Σ^{2+}	The set of all rectangular arrays over Σ	15
Σ_f^{2+}	The set of rectangular arrays over Σ whose shapes are restricted by a shape function f	16
TT(L(n))	The class of deterministic L(n) tape-bounded Turing transducers	131-133
2FSA	The class of deterministic two-dimensional finite-state automata	62
2LBA	The class of deterministic two-dimensional linear-bounded automata	64
2MHA	The class of deterministic two-dimensional multi-head automata	62-63
2MMA	The class of deterministic two-dimensional multi-marker automata	63-64
2TM(L(m,n))	The class of deterministic L(m,n) tape-bounded two-dimensional Turing machines	13-14
2SFST	The class of deterministic two-scan finite-state transducers	147
2WFST	The class of deterministic two-way finite-state transducers	143
2WMCT	The class of deterministic two-way multi-counter transducers	143-144

Term	Meaning	Page
2WMHT	The class of deterministic two-way multi-head transducers	142-143
$W_{\Sigma}^{x,y}$	The set of all rectangular arrays of size x×y over Σ	35
[x]	The greatest integer less than or equal to x	
]x[	The least integer greater than or equal to x	

CHAPTER 1

INTRODUCTION

1.1 Computational Complexity

Since Turing [57] gave his answer to the problem "What kind of logical work can we effectively perform?", many researchers have been making their efforts to investigate the problem "How complicated is it to perform a given logical work?". The concept of "computational complexity" is a formalization of such "difficulty of logical works".

In the study of computational complexity, the "complexity measure" plays a very important role. Borodin proposed (in [1]) that the complexity measures can be categorized into two classes, i.e. the static measures and the dynamic measures. For example, the size of a program or a machine is included in the static measures. The amount of resources (e.g. the running time or the amount of auxiliary memory) required in the computation is regarded as the dynamic measure.

If some programming language or machine model is given, we can define a concrete complexity measure for it. The concepts of time and tape complexities of a Turing machine

defined by Hartmanis et al. [12,47] are such concrete measures (they are both dynamic measures). However, even if a programming language or a machine model is not explicitly given, some abstract measures of complexity can also be defined. Blum [3,4] proposed the size measure and the resource measure which are independent of machine models by an axiomatic method, and derived several interesting properties.

In the theory of automata, there also have been many researches on the computing abilities of various kinds of automata. That is to say, "What kind of machine can do what kind of logical work?" For example, it is well known that the classes of finite-state automata, nondeterministic pushdown automata, nondeterministic linear-bounded automata, and Turing machines form the "Chomsky hierarchy". This can be thought, in some sense, as the hierarchy of the structural complexity of automata. But, generally, these structural differences come from the restrictions on accessibility of the auxiliary memory. Thus, such structural differences of automata are closely related to the resource measure (especially to the amount of memory).

1.2 Tape Complexity of Turing Machine

The concept of "tape complexity" is very convenient to characterize various types of automata from the standpoint of memory requirements. This also facilitates the systematical

investigation of the computing abilities of various classes of automata. Stearns, Hartmanis and Lewis [47] introduced an L(n) tape-bounded (one-dimensional) Turing machine to formalize the notion of tape complexity, and they investigated its computing ability. Later, some results were refined or newly added by Hopcroft and Ullman [17,18]. Here, we give the definition of the L(n) tape-bounded one-dimensional Turing machine, and survey some properties of it.

1.2.1 Tape-Bounded One-Dimensional Turing Machine

Consider a Turing machine (acceptor) T illustrated in Fig.1.1.

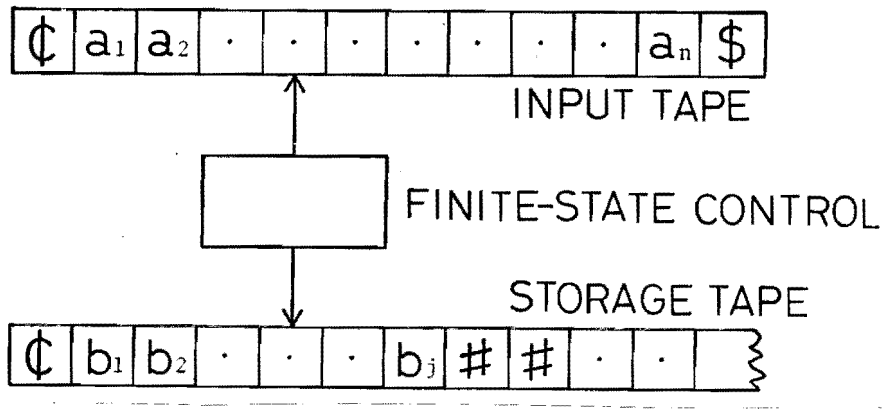

Fig. 1.1 A one-dimensional Turing machine.

T consists of an input tape, a two-way read-only input head, a storage tape, a two-way storage tape head, and a finite-state control. The input tape is a one-dimensional finite-length tape, and is divided into squares in which input

symbols are written. Special border symbols ¢ and $ are attached to the left and the right side ends of the input tape. The input head can move freely in both directions, but it cannot go beyond the border symbols. The storage tape is a one-dimensional semi-infinite tape and divided into squares. A border symbol ¢ is attached to the left side end of it. Initially, a special blank symbol # is written in every square of the storage tape. The storage tape head can read or write the storage tape symbols, moving freely in two ways. The state of the finite-state control is called an *internal state* or simply a *state*. Depending on the present internal state and the symbols read by the input head and the storage tape head, the finite-state control determines the storage tape symbol to be written, the move directions of the input head and the storage tape head, and the next internal state.

Formally a *deterministic one-dimensional Turing machine* (abbreviated to 1TM) is a 9-tuple

$$T = (K, \Sigma, \Gamma, \delta, q_0, ¢, \$, \#, F),$$

where, K is a nonempty finite set of states, Σ and Γ are nonempty finite sets of input symbols and storage tape symbols respectively, $q_0 \in K$ is an initial state, ¢ is a left border symbol of the input tape and the storage tape ($\Sigma \cap \{¢\}=\emptyset$, $\Gamma \cap \{¢\}=\emptyset$), $ is a right border symbol of the input tape ($\Sigma \cap \{\$\}=\emptyset$), # is a blank symbol of the storage tape ($\Gamma \cap \{\#\}=\emptyset$), and $F \subseteq K$ is a set of final states. δ is a mapping from a subset of $K \times (\Sigma \cup \{¢,\$\}) \times (\Gamma \cup \{¢,\#\})$ into $K \times (\Gamma \cup \{¢\}) \times \{L,R,H\}^2$,

where {L,R,H} is the set of move directions of the input head or the storage tape head. T halts for the element of $K\times(\Sigma\bigcup\{¢,\$\})\times(\Gamma\bigcup\{¢,\#\})$ on which δ is not defined.

A finite string (sequence) of the elements of Σ is called a *word* on Σ. The set of all the words on Σ is denoted by Σ^*. And the set of all the words except the null word ε (i.e. the word of length 0) is denoted by Σ^+. (Thus, $\Sigma^*=\Sigma^+\bigcup\{\varepsilon\}$.)

Let us give T a word $w\in\Sigma^+$ with border symbols ¢ and \$. Assume that T begins its movements from the initial state q_0 setting the input head and the storage tape head at the left side ends. If T eventually halts in a final state, then we say that T *accepts* the word w. If T halts in a state other than the final states, we say that T *rejects* w. (If T does not halt, T neither accepts nor rejects.) The set of all the words accepted by T is called a *language* accepted by T.

Let L(n) be a function from **N** into $\mathbf{R}_+$, where **N** and $\mathbf{R}_+$ are the sets of natural numbers and nonnegative real numbers, respectively. If T scans no more than [L(n)] squares ([x] means the greatest integer less than or equal to x) of the storage tape for every input word of length n, T is said to be a *deterministic L(n) tape-bounded one-dimensional Turing machine*. The function L(n) is called a *tape function*, and T is said to have *tape complexity* L(n). We denote the class of deterministic L(n) tape-bounded one-dimensional Turing machine by lTM(L(n)). And let $\mathcal{L}$[lTM(L(n))] denote the class of (one-dimensional) languages accepted by lTM(L(n)).

The tape function L(n) is said to be *constructible*, if there exists some $T \in 1TM(L(n))$ which uses exactly [L(n))] squares of the storage tape and eventually halts, for some input word of length n, for every n.

1.2.2 Some Properties of 1TM(L(n))

There are three important theorems concerning 1TM(L(n)). They are the tape reduction theorem, the hierarchy theorem, and the theorem of the lower bounds on tape growth.

First, the tape reduction theorem is as follows.

Theorem 1.1 (Stearns, Hartmanis and Lewis [47])

Let L(n) be a tape function of 1TM. Then, for any constant c>0,

$$\mathcal{L}[1TM(L(n))] = \mathcal{L}[1TM(c \cdot L(n))].$$

This theorem means that a constant factor of a tape function does not affect the language acceptability of 1TM(L(n)). This property comes from the fact that there is no restriction to the number of storage tape symbols.

The hierarchy theorem asserts the existence of infinite hierarchy of language acceptabilities among 1TM(L(n)). This theorem is partitioned into two cases. The first case is that the tape function L(n) grows at least proportional to log n. In this case, the diagonalization argument is applied to prove this.

Theorem 1.2 (Stearns, Hartmanis and Lewis [47])

Let $L_1(n)$ and $L_2(n)$ be constructible tape functions of 1TM. Suppose that

$$\lim_{i\to\infty} \frac{L_1(n_i)}{L_2(n_i)} = 0$$

$$\frac{L_2(n_i)}{\log n_i} > k$$

for some increasing sequence of natural numbers $\{n_i\}$, and for some constant $k>0$. Then, there exists a language L such that $L \in \mathcal{L}[1TM(L_2(n))]$ but $L \notin \mathcal{L}[1TM(L_1(n))]$.

The second case is that the tape function grows more slowly than log n. Hopcroft anf Ullman [17] proved this using the notion of the "transition matrix". The following theorem also holds for the nondeterministic Turing machine, because this argument can easily be expanded to the non-deterministic case.

Theorem 1.3 (Hopcroft and Ullman [17])

Let $L_1(n)$ and $L_2(n)$ be constructible tape functions of 1TM. Suppose that

$$\lim_{i\to\infty} \frac{L_1(n_i)}{L_2(n_i)} = 0$$

$$\frac{L_2(n_i)}{\log n_i} < \frac{1}{2}$$

for some increasing sequence of natural numbers $\{n_i\}$. Then, there exists a language L such that $L \in \mathcal{L}[1TM(L_2(n))]$ but

$L \notin \mathcal{L}[1TM(L_1(n))]$.

Combining Theorem 1.2 and 1.3, we can see that if two constructible tape functions $L_1(n)$ and $L_2(n)$ satisfies

$$\liminf_{n\to\infty} \frac{L_1(n)}{L_2(n)} = 0,$$

then there exists a language L which is accepted by some $1TM(L_2(n))$ but not accepted by any $1TM(L_1(n))$. Since there are various constructible tape functions, there exists an infinite hierarchy of language acceptabilities among 1TM.

However, the next theorem states that there exists no constructible tape function which grows more slowly than the order of log log n. Thus no hierarchy of language acceptability exists below the tape complexity log log n.

Theorem 1.4 (Hopcroft and Ullman [17])

Let L(n) be a constructible tape function of 1TM. If $\limsup_{n\to\infty} L(n) = \infty$, then

$$\limsup_{n\to\infty} \frac{L(n)}{\log \log n} > 0.$$

It is known that there is a constructible tape function which increases in proportion to log log n [10,28]. Thus, indeed, the order of log log n is the lower bound of tape growth.

1.3 Standpoint of This Research

In this thesis, we study the computing abilities of various kinds of automata from the standpoint of computational complexity. We mainly pay attention to the notion of tape complexity and some other complexity measures related to it. And we investigate how the computing abilities of automata are characterized by these measures.

The problem of computational complexity also arises in the two-dimensional information processing. Blum and Hewitt [5] first proposed two-dimensional tape automata, and investigated their pattern recognition abilities. Since then, there have been several reserches in this field, but each of these researches was the one which deals with merely particular two-dimensional automata.

The concept of tape complexity is also applicable to the case of two-dimensional pattern processing, and it plays an important role to unify the automata-theoretic approaches in this field. Here, we propose a tape-bounded two-dimensional Turing machine. It will be seen (in chapter 3) that various two-dimensional tape automata are characterized by the notion of tape complexity. Thus, many properties of these automata can be uniformly and systematically derived from the general theory of tape-bounded two-dimensional Turing machine.

The tape-bounded two-dimensional Turing machine is considered to be a natural expansion of the one-dimensional case. So, some results can be derived in a similar manner as

in the one-dimensional case. However, there also exist many properties peculiar to the two-dimensional case. For example, we shall see that the property concerning the lower bounds on tape growth is quite different from the one-dimensional case.

As seen in the previous section, the systematic study of language acceptabilities of one-dimensional tape automata from the standpoint of tape complexity have already been outlined by several researchers. Recently, very precise investigations of language acceptabilities of automata of tape complexity log n are being made. The tape complexity log n is very interesting, because it is, in some sense, the "critical" tape complexity. It is based on the fact that the tape complexity log n is a necessary and sufficient amount of memory to keep the coordinates on the input tape. There are various concrete models of automata of this complexity, and some of them are newly defined here. Furthermore, it will be seen that these automata are closely related to some class of multi-dimensional finite-state automata. In chapter 4, we consider the detailed relations among these automata, and investigate several subhierarchies of acceptabilities in the class of tape complexity log n.

So far most of the researches concerning the tape complexity were the ones which had investigated the acceptabilities of automata. In the last chapter of this thesis, we introduce a tape-bounded Turing transducer and some other models, each of which computes a recursive function. Here, we consider

how the computing ability of recursive functions is characterized by an amount of auxiliary memory.

CHAPTER 2

TWO-DIMENSIONAL TURING MACHINE

Recently, there have been several attempts to study two-dimensional information processing, such as pattern recognition, from the standpoint of automata theory. A two-dimensional Turing machine is considered as one of the most convenient mathematical model for this purpose. In this chapter, a deterministic L(m,n) tape-bounded two-dimensional Turing machine and its variants are introduced to formalize memory limited computations in two-dimensional information processing. It is a Turing machine which uses no more than L(m,n) squares of the storage tape for every two-dimensional input of size m×n. Thus the function L(m,n) is regarded as one of the measures of computational complexity connected with memory requirements. Henceforth a two-dimensional Turing machine is treated as an acceptor of a set of two-dimensional patterns, which is called a "two-dimensional language" by an analogy of the one-dimensional formal language theory. In what follows, general properties of their language acceptabilities are investigated.

2.1 Definitions

We consider a deterministic two-dimensional Turing machine (acceptor) T illustrated in Fig.2.1.

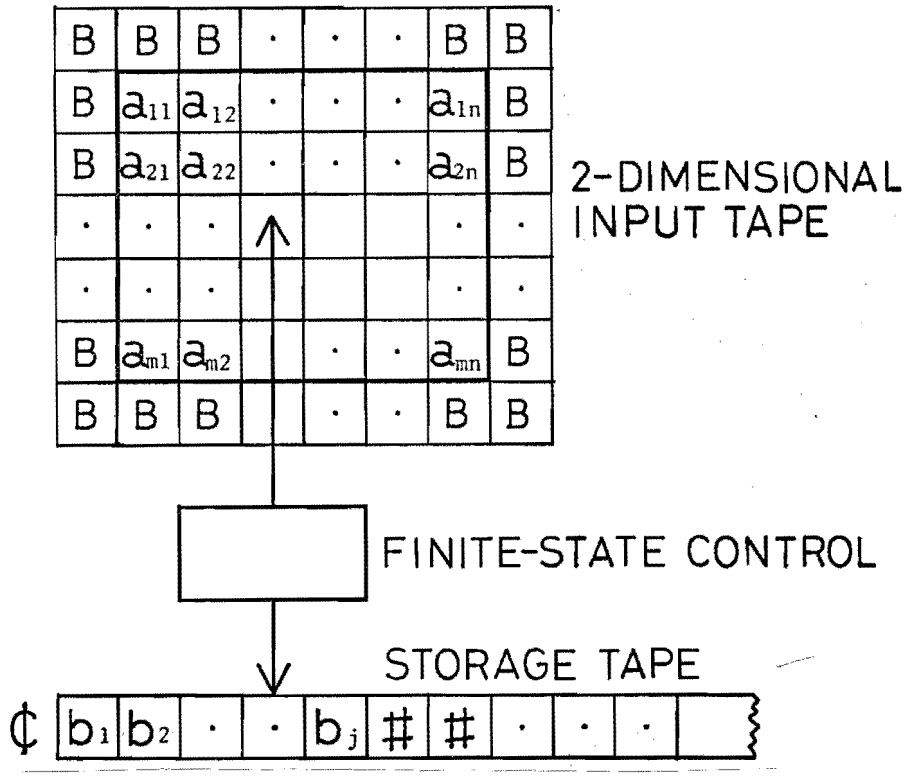

Fig. 2.1 A two-dimensional Turing machine.

The definitions concerning the two-dimensional Turing machine are analogous to those of the one-dimensional Turing machine, except that the dimension of the input tape is different.

T consists of a rectangle-shaped input tape, a read-only input head, a storage tape, a storage tape head, and a finite-state control. The input tape is divided into

m×n squares and surrounded by special border symbols. (m and n are arbitrary natural numbers.) An input symbol is written in each square of the tape. The input head can move freely in four directions (north, south, east, and west) on the input tape, but never falls off. The storage tape is a semi-infinite one-dimensional tape and divided into squares in which storage tape symbols are written. The storage tape head can read or write the storage tape symbols, moving on the tape in two ways. A special blank symbol is written in every square of the storage tape, when T starts to move.

Formally a *deterministic two-dimensional Turing machine* (abbreviated to 2TM) is a 9-tuple

$$T = (K, \Sigma, \Gamma, \delta, q_0, B, ¢, \#, F),$$

where K is a nonempty finite set of states, Σ and Γ are nonempty finite sets of input symbols and storage tape symbols, $q_0 \in K$ is an initial state, B is a border symbol of the input tape ($\Sigma \cap \{B\}=\emptyset$), ¢ and # are a border symbol and a blank symbol of the storage tape ($\Gamma \cap \{¢,\#\}=\emptyset$), and $F \subseteq K$ is a set of final states. δ is a mapping from a subset of $K\times(\Sigma \cup \{B\})\times(\Gamma \cup \{¢,\#\})$ into $K\times(\Gamma \cup \{¢\})\times\{N,S,E,W,H\}\times\{L,R,H\}$, where {N,S,E,W,H} and {L,R,H} are the sets of move directions of the input head and the storage tape head. T halts for the element of $K\times(\Sigma \cup \{B\})\times(\Gamma \cup \{¢,\#\})$ on which δ is not defined.

A rectangular array of the elements of Σ is said to be a *two-dimensional word* on Σ (or simply a *word* on Σ, if no confusion occurs). A set of all the two-dimensional words

on Σ is denoted by Σ^{2+}. (Σ^{2+} does not contain a word of size 0×0.)

Assume that a word $w \in \Sigma^{2+}$ with surrounding border symbols is given to T as an input tape. And suppose that T starts in the initial state q_0, setting the input head at the north-west corner of the input tape and the storage tape head at the left side end of the storage tape. We say that T *accepts* the two-dimensional word w, if T eventually halts in a final state. And we say that T *rejects* w, if T halts in a state other than the final states. The set of all the words accepted by T is called a *two-dimensional language* (or symply a *language*, when no confusion occurs) accepted by T.

Let L(m,n) be a function from $\mathbf{N}^2$ into $\mathbf{R}_+$. If T scans no more than [L(m,n)] squares of the storage tape for every input of size m×n, T is said to be an *L(m,n) tape-bounded two-dimensional Turing machine*. L(m,n) is called a *tape function* and we say that T has *tape complexity* of L(m,n). The class of L(m,n) tape-bounded two-dimensional Turing machines and the class of two-dimensional languages accepted by them are denoted by 2TM(L(m,n)) and $\mathcal{L}$[2TM(L(m,n))], respectively.

We say that the tape function L(m,n) is *constructible*, if there exists some $T \in$ 2TM(L(m,n)) which uses exactly [L(m,n)] squares of the storage tape and eventually halts for some input of m×n, for every m and n.

A *computational configuration* of T is a combination

of the internal state, the position of the input head, the contents of the storage tape, and the position of the storage tape head. If T is a 2TM(L(m,n)), the total number of computational configurations for an input of size m×n is at most $s \cdot m \cdot n \cdot [L(m,n)] \cdot t^{[L(m,n)]}$, where s and t are the numbers of internal states and storage tape symbols. A *storage state* of T is a combination of the internal state, the contents of the storage tape and the position of the storage tape head. Similarly, the total number of storage states for an input of size m×n is at most $s \cdot [L(m,n)] \cdot t^{[L(m,n)]}$.

Next, we define a two-dimensional Turing machine whose inputs are restricted to the ones of special shapes. Let ℓ_1 and ℓ_2 be functions from Σ^{2+} into **N**, which pick up vertical and horizontal sidelengths from a two-dimensional word on Σ (i.e. $\ell_1(w)$ and $\ell_2(w)$ are vertical and horizontal sidelengths of the word w respectively). Let f be an injection from **N** into $\mathbf{N}^2$, which is called a *shape function*. We consider a set of words Σ_f^{2+} defined by the shape function f.

$$\Sigma_f^{2+} = \{w \mid w \in \Sigma^{2+},\ \exists j\ ;\ f(j)=(m,n),\ \ell_1(w)=m,\ \ell_2(w)=n\}.$$

We call j an *index* of $w \in \Sigma_f^{2+}$, such that $f(j)=(m,n)$, $\ell_1(w)=m$, and $\ell_2(w)=n$. The class of 2TM whose inputs are restricted to Σ_f^{2+} is denoted by $2TM^f$. Let L(j) be a function from **N** into $\mathbf{R}_+$, and let T be any $2TM^f$. If T uses no more than [L(j)] squares of its storage tape for every input with the index j, we say T is an L(j) tape-bounded $2TM^f$. (So, f must be an injection.) The class of L(j) tape-bounded $2TM^f$ is

denoted by $2TM^f(L(j))$. Generally, a tape function L(j) for $2TM^f$ is a function of the index j. But as a special case of $2TM^f(L(j))$, if f is an injection such that f(j)=(j, f'(j)) (or f(j)=(f'(j), j)) for some injection f' : $\mathbf{N} \to \mathbf{N}$, then L(j) becomes a function of a vertical (or horizontal) sidelength of an input. So, in such a case, we express the tape function by L(m) (or L(n)) when no confusion occurs. Especially, if s : $\mathbf{N} \to \mathbf{N}^2$ be a shape function such that s(j)=(j,j), then $\Sigma_s^{2+}=\{w \mid w \in \Sigma^{2+},\ \ell_1(w)=\ell_2(w)\}$ is the set of all the square-shaped words on Σ. The class of 2TM whose inputs are restricted to square-shaped words is denoted by $2TM^s$. The class of L(m) tape-bounded $2TM^s$, which uses at most [L(m)] squares of the storage tape for every input of size m×n, is denoted by $2TM^s(L(m))$.

Finally, we define a class of 2TM with one input symbol. Let $\Sigma_1=\{a\}$. The class of 2TM whose inputs are restricted to $(\Sigma_1)^{2+}$ is denoted by $2TM^1$. And $2TM^1(L(m,n))$ denotes the class of 2TM which uses at most [L(m,n)] squares of the storage tape for an input of size m×n, for every m and n. Similarly, $2TM^{1f}$ and $2TM^{1s}$ denote $2TM^f$ on $(\Sigma_1)_f^{2+}$ and $2TM^s$ on $(\Sigma_1)_s^{2+}$, respectively.

2.2 Closure Properties Under Boolean Operations

In this section, we investigate some closure properties of the class of languages accepted by 2TM under Boolean

operations.

First, we consider complementation.

Theorem 2.1 Let L(m,n) be a tape function of 2TM. If $c \cdot L(m,n) \geq \log mn$[†] for some constant $c \geq 1$, then the class of languages $\mathcal{L}[2TM(L(m,n))]$ is closed under complementation.

Proof. Let T_1 be any 2TM(L(m,n)). We construct $T_2 \in 2TM(L(m,n))$ which accepts the complement of the language accepted by T_1, in the following way.

Let s and t be the numbers of internal states and storage tape symbols of T_1 respectively. First, T_2 marks (log mn)/]c[squares of the storage tape by counting the total number of squares of the input tape with a number of base $2^{]c[}$ (]x[means the least integer greater than or equal to x). Then T_2 begins to simulate T_1 step by step. In this simulation, if T_1 halts in an accepting (final) state, then T_2 halts in a rejecting state. Conversely, if T_1 halts in a rejecting state, then T_2 halts in an accepting state. At the same time, in order to check whether T_1 loops or not, T_2 counts the number of steps of T_1 with a number of base r ($r=2^{]c[+1} \cdot st$) by using the other track of the storage tape. To do this, T_2 uses (log mn)/]c[squares of the storage tape during the period T_1 is using the storage tape less than (log mn)/]c[squares.

---

† In what follows, we assume that the base of logarithm is always 2.

But if T_1 has used more than (log mn)/]c[squares, T_2 counts it by using the squares which T_1 has used so far. If the numbers of squares which T_1 has used for an given input of size m×n is ℓ ($\leq$L(m,n)), then the total number of computational configurations of T_1 is $smn\ell t^{\ell}$. Thus, if T_1 does not halt in less than $smn\ell t^{\ell}+1$ steps, we can conclude T_1 is looping. Since $r=2^{]c[+1}st$, $smn\ell t^{\ell}<r^{\ell}$ holds if $\ell>$(log mn)/]c[. Similarly $smn\ell t^{\ell}<r^{(\log\ mn)/]c[}$ holds if $\ell\leq$(log mn)/]c[. Thus in either case, T_2 can check whether T_1 loops or not, and T_2 accepts the input if T_1 loops.

Obviously, T_2 accepts the complement of the language accepted by T_1. This completes the proof. (Q.E.D.)

When the tape function L(m,n) grows more slowly than the order of log mn, it remains open whether $\mathcal{L}$[2TM(L(m,n))] is closed under complementation.

Now, we consider the operation of intersection between two languages in the same class of tape complexity.

Theorem 2.2 Let L(m,n) be a tape function of 2TM. If L_1, $L_2 \in \mathcal{L}$[2TM(L(m,n))], then $(L_1 \cap L_2) \in \mathcal{L}$[2TM(L(m,n))].

Proof. Let T_1 and T_2 be 2TM(L(m,n)) which accept L_1 and L_2 respectively. We construct $T' \in$ 2TM(L(m,n)) which accepts $(L_1 \cap L_2)$ as follows. Assume an input w is given to T'. T' first simulates T_1 with w. If T_1 halts and accepts w, then T' begins to simulate T_2 with w. T' accepts w if

and only if both T_1 and T_2 accepts it. (Q.E.D.)

From Theorem 2.1, 2.2 and de Morgan's law, the next theorem can be obtained.

Theorem 2.3 Let L(m,n) be a tape function of 2TM such that $c \cdot L(m,n) \geq \log mn$ holds for some constant $c \geq 1$. Then $\mathcal{L}[2TM(L(m,n))]$ forms a Boolean algebra.

Concerning $2TM^f$, these properties can also be shown in the same way as in Theorem 2.1 and 2.2. So we can obtain the next corollary.

Corollary 2.4 Let $f : \mathbf{N} \to \mathbf{N}^2$ be a shape function, and let L(j) be a tape function of $2TM^f$. Suppose that there exists some constant $c \geq 1$ such that $c \cdot L(j) \geq \log mn$ holds for every input $w \in \Sigma_f^{2+}$, where j is the index of w and f(j)=(m,n). Then $\mathcal{L}[2TM^f(L(j))]$ forms a Boolean algebra.

Next, we consider the closure properties of $\mathcal{L}[2TM^1(L(m,n))]$ under these operations, i.e. the case that the number of the input symbols is restricted to one. First, it is shown that $\mathcal{L}[2TM(L(m,n))]$ is closed under complementation, even if the condition is somewhat weaker than the case of Theorem 2.1. But it still remains open whether $\mathcal{L}[2TM^1(L(m,n))]$ is closed under complementation when L(m,n) grows more slowly than both the orders of log m and log n.

Theorem 2.5 Let $L(m,n)$ be a tape function of $2TM^1$ such that $c \cdot L(m,n) \geq \log m$ (or, $c \cdot L(m,n) \geq \log n$) holds for every m and n, for some constant $c \geq 1$. Then $\mathcal{L}[2TM^1(L(m,n))]$ is closed under complementation.

To prove this theorem, we show the next lemma.

Lemma 2.6 Let $L(m,n)$ be a tape function of $2TM^1$ such that $c \cdot L(m,n) \geq \log m$ (or, $c \cdot L(m,n) \geq \log n$) holds for every m and n, for some constant $c \geq 1$. Then, for every $T_1 \in 2TM^1(L(m,n))$, there exists $T_2 \in 2TM^1(L(m,n))$ which accepts the same language as T_1 and never loops for any input.

Proof. We consider only the case of $c \cdot L(m,n) \geq \log m$. First, we construct following $T_1' \in 2TM^1(L(m,n))$ from T_1. Assume an input w of size $m \times n$ is given to T_1'. T_1' first writes the number m, using $(\log m)/]c[$ squares of the storage tape. This can be done by counting the vertical sidelength of w with a number of base $2^{]c[}$. Hereafter, the input head of T_1' moves only on the first row of w. T_1' then begins to simulate T_1 step by step in the following manner. The horizontal movements of the input head of T_1 is simulated by that of T_1'. But, its vertical movements is simulated by keeping its vertical coordinate in the storage tape of T_1'. T_1' does this using a number of base $2^{]c[}$. And T_1' uses the other track of the storage tape to simulate the storage tape of T_1. T_1' accepts the input w, if and only if T_1 halts in an accepting

state. Apparently T_1' accepts the same language as T_1.

Now, we construct T_2 which simulates T_1' and halts even if T_1' loops. This can be shown in a similar manner as in the one-dimensional case (Hartmanis and Berman [15]). If an input w of size m×n is given to T_2, T_2 begins to simulate T_1', and accepts w if and only if T_1' accepts it. Simultaneously, T_2 checks whether T_1' loops or not, in the following way. Let s and t be the numbers of internal states and the storage tape symbols of T_1'. And let ℓ ($\leq L(m,n)$) be the number of squares of the storage tape which T_1' actually uses for the input w. Then the total number of storage states of T_1' is $s\ell t^{\ell}$. T_1' can loop only if the input head is moving on the first row of the input. There are two situations when T_1' loops, i.e., (1) T_1' reads the left or the right border symbol of the input tape infinitely many times, or (2) T_1' does not read the border symbol in the loop. To check whether the case (1) is occurring, T_2 counts the number of times that T_1' have read the border symbol, using the other track of the storage tape with a number of base r, where r=2st. If this number exceeds the maximum number which T_2 can count by using the storage tape of the length T_1' has used so far, then T_2 can conclude that T_1' is looping. Because, T_1' cannot read the left or the right border symbol more than $s\ell t^{\ell}$ ($<r^{\ell}$) times without looping. Next, to check the case (2), T_2 counts the number of steps since the time T_1' has read a border symbol lastly, using the other track of the storage tape with a number of base r. If this number

exceeds the maximum number that T_2 can count by using the squares T_1' has used so far, the same storage state must have appeared in this period. And if the positions of the input head are also the same, we can conclude T_1' is looping. To check this, T_2 records the storage state of T_1' at that moment, and continues the simulation of T_1'. Furthermore, T_2 begins to count the relative position of the input head of T_1', with a number of base r. If the head position becomes the same when T_1' next enters the same storage state as T_2 is keeping, and if T_1' have not read a border symbol in this period, then T_2 halts and rejects w. Otherwise T_1' will read a border symbol at some time, so T_2 continues to simulate it.

Thus T_2 accepts the same language as T_1 and always halts for every input. (Q.E.D.)

Proof of Theorem 2.5. Let T_1 be an arbitrary $2TM^1(L(m, n))$. In a similar way as in Lemma 2.5, we can construct $T_2' \in 2TM^1(L(m,n))$ which accepts an input w if and only if T_1 rejects it or loops. (Q.E.D.)

The following theorem can be proved in the same way as in Theorem 2.2.

Theorem 2.7 Let L(m,n) be a tape function of $2TM^1$. If $L_1, L_2 \in \mathcal{L}[2TM^1(L(m,n))]$, then $(L_1 \cap L_2) \in \mathcal{L}[2TM^1(L(m,n))]$.

From Theorem 2.6 and 2.7, it is seen that $\mathcal{L}[2TM^{1}(L(m,n))]$ forms a Boolean algebra under a little weaker condition than in Theorem 2.3.

Theorem 2.8 Let L(m,n) be a tape function of $2TM^{1}$ such that $c \cdot L(m,n) \geq \log m$ (or $c \cdot L(m,n) \geq \log n$) holds for every m and n, for some constant $c \geq 1$. Then $\mathcal{L}[2TM^{1}(L(m,n))]$ forms a Boolean algebra.

As for $2TM^{1f}(L(j))$, the next corollary can be derived likewise.

Corollary 2.9 Let $f : \mathbf{N} \to \mathbf{N}^2$ be a shape function and let L(j) be a tape function of $2TM^{1f}$. Suppose that there exists some constant $c \geq 1$ such that $c \cdot L(j) \geq \log m$ (or $c \cdot L(j) \geq \log n$) holds for every input $w \in \{a\}_f^{2+}$, where j is the index of w and f(j)=(m,n). Then $\mathcal{L}[2TM^{1f}(L(j))]$ forms a Boolean algebra.

2.3 Tape Reduction Theorem

In section 2.3-2.5, we will consider how the language acceptabilities of 2TM are characterized by the tape function L(m,n). In other words, the question is "What makes 2TM(L(m, n)) accept new languages if L(m,n) is varied?" To answer this, we will show the tape reduction theorem and the hierarchy theorem of 2TM.

In this section, some conditions which do not change the language acceptability of 2TM are investigated.

The next theorem shows that the language acceptability is not affected by adding a constant to the tape function.

Theorem 2.10 Let $L(m,n)$ be a tape function of 2TM. Then for any constant $c>0$, $\mathcal{L}[2TM(L(m,n))] = \mathcal{L}[2TM(L(m,n)+c)]$.

Proof. Let T_1 be any $2TM(L(m,n)+c)$. We will construct $T_2 \in 2TM(L(m,n))$ which accepts the same language as T_1. T_2 can easily do this by simulating the leftmost $]c[$ squares of the storage tape of T_1 in the finite-state control of T_2. Thus $\mathcal{L}[2TM(L(m,n))] \supseteq \mathcal{L}[2TM(L(m,n)+c)]$. And $\mathcal{L}[2TM(L(m,n))] \subseteq \mathcal{L}[2TM(L(m,n)+c)]$ is obvious, so $\mathcal{L}[2TM(L(m,n))] = \mathcal{L}[2TM(L(m,n)+c)]$ is concluded. (Q.E.D.)

Apparently this theorem also holds for $2TM^{l}(L(m,n))$, $2TM^{f}(L(j))$ and $2TM^{lf}(L(j))$.

Corollary 2.11 Let $f : \mathbf{N} \to \mathbf{N}^2$ be a shape function and let $L(m,n)$ and $L(j)$ be tape functions of $2TM^{l}$ and $2TM^{f}$ (or $2TM^{lf}$). Then for any constant $c>0$,

$$\mathcal{L}[2TM^{l}(L(m,n))] = \mathcal{L}[2TM^{l}(L(m,n)+c)],$$
$$\mathcal{L}[2TM^{f}(L(j))] = \mathcal{L}[2TM^{f}(L(j)+c)],$$
$$\mathcal{L}[2TM^{lf}(L(j))] = \mathcal{L}[2TM^{lf}(L(j)+c)].$$

The next theorem is the tape reduction theorem for 2TM.

It can be proved in a similar way as in the one-dimensional case (Theorem 1.1).

Theorem 2.12 (Tape Reduction Theorem) Let $L(m,n)$ be a tape function of 2TM. Then, for any constant $c>0$,

$$\mathcal{L}[2TM(L(m,n))] = \mathcal{L}[2TM(c\cdot L(m,n))].$$

Proof. For an arbitrary $T_1 \in 2TM(L(m,n))$, we will construct $T_2 \in 2TM(c\cdot L(m,n)+1)$ which accepts the same language as T_1 in the following way. If the number of the storage tape symbols of T_1 is t, then that of T_2 is $t^{]1/c[}$. T_2 uses each square of the storage tape to record the contents of each block, which is made up by punctuating the storage tape of T_1 every $]c[$ squares. By this, T_2 can easily simulate T_2 using at most $(L(m,n)/]1/c[)+1$ squares of the storage tape for every input of size $m\times n$. Furthermore, from Theorem 2.10, $\mathcal{L}[2TM(L(m,n))] \subseteq \mathcal{L}[2TM(c\cdot L(m,n))]$ is concluded. $\mathcal{L}[2TM(L(m,n))] \supseteq \mathcal{L}[2TM(c\cdot L(m,n))]$ can be shown likewise, and this completes the proof. (Q.E.D.)

Corollary 2.13 Let $f : \mathbf{N} \to \mathbf{N}^2$ be a shape function, and let $L(m,n)$ and $L(j)$ be tape functions of $2TM^1$ and $2TM^f$ (or $2TM^{1f}$). Then for any constant $c>0$,

$$\mathcal{L}[2TM^1(L(m,n))] = \mathcal{L}[2TM^1(c\cdot L(m,n))],$$
$$\mathcal{L}[2TM^f(L(j))] = \mathcal{L}[2TM^f(c\cdot L(j))],$$
$$\mathcal{L}[2TM^{1f}(L(j))] = \mathcal{L}[2TM^{1f}(c\cdot L(j))].$$

2.4 Hierarchy Theorem of 2TM

Next, we derive the condition that produces a difference of language acceptability between two tape complexity classes. This theorem is called the hierarchy theorem, because it can be seen that there exists an infinite hierarchy among 2TM from it. First we will show the hierarchy theorem of 2TM(L(m,n)) with two or more input symbols. Its proof is partitioned into three lemmas (Lemma 2.14, 2.16 and 2.18).

Definition. An infinite sequence of pairs of natural numbers $\{(m_i,n_i)\}$ is called a *regular sequence* if $(m_i,n_i) \neq (m_j,n_j)$ for all $i \neq j$.

Definition. We say that a mapping $h : \Sigma^{2+} \to (\Sigma')^{2+}$ is a *homomorphism*, if $h(u \cdot v)=h(u) \cdot h(v)$ and $h\begin{pmatrix} u \\ \cdot \\ v \end{pmatrix}=\begin{matrix} h(u) \\ \cdot \\ h(v) \end{matrix}$ hold for all $u,v \in \Sigma^{2+}$, where $u \cdot v$ (or $\begin{matrix} u \\ \cdot \\ v \end{matrix}$) means the horizontal (vertical) concatenation of two words u and v. Note that the horizontal (vertical) concatenation is defined only if the vertical (horizontal) sidelengths of the two words are the same.

In Lemma 2.14, we first consider the case such that the tape function L(m,n) grows at least proportional to log m + log n for some infinite set of (m,n).

Lemma 2.14 Let $L_1(m,n)$ and $L_2(m,n)$ be constructible

tape function of 2TM. Suppose that

$$\lim_{i\to\infty} \frac{L_1(m_i,n_i)}{L_2(m_i,n_i)} = 0 \qquad (2.1)$$

$$\frac{L_2(m_i,n_i)}{\log m_i + \log n_i} > k \qquad (2.2)$$

for some regular sequence $\{(m_i,n_i)\}$ and for some constant $k>0$. Then, there exists a language L such that $L \in \mathcal{L}[2TM(L_2(m,n))]$ but $L \notin \mathcal{L}[2TM(L_1(m,n))]$.

Proof. This lemma can be proved using the diagonalization technic. We will construct $T \in 2TM(L_2(m,n))$ which accepts the language L not accepted by any $2TM(L_1(m,n))$. Let T' be a 2TM which constructs the tape function $L_2(m,n)$. If the set of input symbols of T' is Σ', then that of T is $\Sigma=\Sigma'\times\{0,1\}$. Let h_1: $(\Sigma'\times\{0,1\})^{2+} \to (\Sigma')^{2+}$ and h_2: $(\Sigma'\times\{0,1\})^{2+} \to \{0,1\}^{2+}$ be homomorphisms such that $h_1((a,i))=a$ and $h_2((a,i))=i$ for all $a \in \Sigma'$ and $i \in \{0,1\}$. Let α: $\{0,1\}^{2+} \to \mathbf{N}$ be an easily computable total recursive mapping which satisfies the following condition. For every $j \in \mathbf{N}$, there exist two integers a_j and b_j, and for every (m,n) that satisfies $m>a_j$ or $n>b_j$, there exists a word $u \in \{0,1\}^{2+}$ of size $m\times n$ such that $\alpha(u)=j$. For example, the binary number which is obtained by replacing the word u in a row satisfies this condition. Now, suppose an input word $w \in \Sigma^{2+}$ is given to T. T first simulates T' with the input $h_1(w)$, until T' halts. (If T' does not halt, T also does not.) Henceforth, if T is ready to use more squares of the storage tape than T' has used, then T halts and rejects the input. Next, T computes $\alpha(h_2(w))$ and begins

to simulate $\alpha(h_2(w))$-th 2TM (which is denoted by $T_{\alpha(h_2(w))}$) with the input w. Assume that w is a word of size $m_i \times n_i$ for sufficiently large i, and $h_1(w)$ makes T' use exactly $[L(m_i,n_i)]$ squares of the storage tape. If $T_{\alpha(h_2(w))}$ is a $2TM(L_1(m,n))$, T can simulate it within $[L_2(m_i,n_i)]$ squares of the storage tape. Because, T needs $c \cdot L_1(m_i,n_i)$ squares to simulate $T_{\alpha(h_2(w))}$, where $c>0$ is a constant depending on the number of the storage tape symbols of $T_{\alpha(h_2(w))}$, and from Eq.(2.1), $c \cdot L_1(m_i,n_i) < L_2(m_i,n_i)$ holds for sufficiently large i. Furthermore, T can decide whether $T_{\alpha(h_2(w))}$ halts or not. The total number of computational configurations of $T_{\alpha(h_2(w))}$ is at most

$$x_i = sm_in_i[L_1(m_i,n_i)]t^{[L_1(m_i,n_i)]},$$

where s and t are the numbers of states and storage tape symbols of $T_{\alpha(h_2(w))}$. T counts the number of steps of $T_{\alpha(h_2(w))}$ using the other track of the storage tape with a number of base $2^{]1/k[}$. From Eq.(2.1) and (2.2),

$$\lim_{i\to\infty} \frac{x_i}{(2^{]1/k[})^{[L_2(m_i,n_i)]}} = 0.$$

Thus for sufficiently large i, if $T_{\alpha(h_2(w))}$ does not halt less than $(2^{]1/k[})^{[L_2(m_i,n_i)]}$ steps, T can conclude that $T_{\alpha(h_2(w))}$ is looping. T accepts w if and only if $T_{\alpha(h_2(w))}$ rejects w or loops.

Let L be the language accepted by T. Now suppose that the j-th 2TM, $T_j \in 2TM(L_1(m,n))$ accepts L. If we give T_j a word $w \in \Sigma^{2+}$ of size $m_i \times n_i$ which satisfies $\alpha(h_2(w))=j$ (for

i large enogh), then a contradiction occurs. Thus $L \notin \mathcal{L}[2TM(L_1(m,n))]$. (Q.E.D.)

Next, we consider the case that the tape function L(m,n) grows more slowly than both the orders of log m and log n for some infinite set of (m,n).

As a preliminary, we define (T,ℓ)-equivalent chunks. The notion of a chunk was introduced by Blum and Hewitt [5] to investigate the language acceptabilities of two-dimensional finite-state automata. Here, we expand this notion for the two-dimensional Turing machines.

Definition. A *chunk* is a square-shaped word without border symbols. Let C_Σ^x denote a set of all the chunks on Σ of sidelength x. Let $c_1, c_2 \in C_\Sigma^x$ be two chunks, and let T be a 2TM that has s states and t storage tape symbols. Now, we assume that T uses at most ℓ squares of the storage tape. Then the total number of the storage states of T is at most $s\ell t^\ell$. Therefore, T has $4xs\ell t^\ell$ ways to enter a chunk of sidelength x. For each case of $4xs\ell t^\ell$ entering ways, T can choose its behaviour at most in $(4xs\ell t^\ell+s+1)$ ways, which are $4xs\ell t^\ell$ ways of going out of the chunk, s ways of halting in the chunk and 1 way of looping. Two chunks c_1 and c_2 of the same size are said to be *(T,ℓ)-equivalent*, if T chooses the same behaviours to c_1 and c_2 for all the ways of entering. So, T cannot distinguish (T,ℓ)-equivalent chunks if T uses at most ℓ squares of the storage tape. The total number of

(T,ℓ)-equivalence classes of C_Σ^x is at most $(4xs\ell t^\ell+s+1)^{4xs\ell t^\ell}$.

Now, the following lemma can be obtained.

Lemma 2.15 Let T be an arbitrary 2TM. Let $\{(x_i,\ell_i)\}$ be a sequence of pairs of nonnegative integers that satisfies

$$\lim_{i\to\infty} \frac{\ell_i}{\log x_i} = 0 \tag{2.3}$$

$$\lim_{i\to\infty} x_i = \infty \tag{2.4}$$

and let $\{D_i\}$ $(D_i \subseteq C_\Sigma^{x_i})$ be a sequence of sets of chunks that satisfies

$$|D_i| \geq r^{x_i^2} \tag{2.5}$$

for some constant $r>1$. ($|D_i|$ means the number of elements of D_i.) Then, for every sufficiently large i, there exist two different chunks $c_i, c_i' \in D_i$ which are (T,ℓ_i)-equivalent.

Proof. Let s and t be the numbers of internal states and storage tape symbols of T, respectively. The total number of (T,ℓ_i)-equivalence classes is at most $(g_i+s+1)^{g_i}$, where $g_i = 4x_i s\ell_i t^{\ell_i}$. Now, we denote

$$f_i = \frac{(g_i+s+1)^{g_i}}{r^{x_i^2}}.$$

By Eq.(2.3) and (2.4), we can derive $\lim_{i\to\infty} f_i = 0$. So, from Eq.(2.5), the following inequality holds for sufficiently

large i.

$$|D_i| \geq r^{x_i^2} > (g_i+s+1)^{g_i}$$

Thus, there are two different chunks $c_i, c_i' \in D_i$ which are (T,ℓ_i)-equivalent. (Q.E.D.)

From Lemma 2.15, the next lemma can be obtained.

Lemma 2.16 Let $L_1(m,n)$ and $L_2(m,n)$ be constructible tape functions of 2TM. Suppose that

$$\lim_{i\to\infty} \frac{L_1(m_i,n_i)}{L_2(m_i,n_i)} = 0 \qquad (2.6)$$

$$\lim_{i\to\infty} L_2(m_i,n_i) = \infty \qquad (2.7)$$

$$\frac{L_2(m_i,n_i)}{\log m_i} < k \quad \text{and} \quad \frac{L_2(m_i,n_i)}{\log n_i} < k \qquad (2.8)$$

for some regular sequence $\{(m_i,n_i)\}$ and some constant $k>0$. Then there exists a language L such that $L \in \mathcal{L}[2TM(L_2(m,n))]$ but $L \notin \mathcal{L}[2TM(L_1(m,n))]$.

Proof. We construct $T_1 \in 2TM(L_2(m,n))$ which accepts the language L not accepted by any $2TM(L_1(m,n))$. Let T' be a two-dimensional Turing machine which constructs the tape function $L_2(m,n)$. If the set of input symbols of T' is Σ', then that of T_1 is $\Sigma=\Sigma'\times\{0,1\}$. Let h_1 and h_2 be the same mappings as in Lemma 2.14. Suppose an input word $w \in \Sigma^{2+}$ of size $m\times n$ is given to T_1. First T_1 simulates the

movements of T' with the input $h_1(w)$ until it halts. Let ℓ be the number of squares of the storage tape which T' has used. In general $\ell \leq [L_2(m,n)]$, but $\ell = [L_2(m,n)]$ holds for an suitable input of size m×n. Now we consider two chunks w_α and w_β of sidelength $x = 2^{[\ell/k]-1}$ at the north-west corner of w (Fig.2.2).

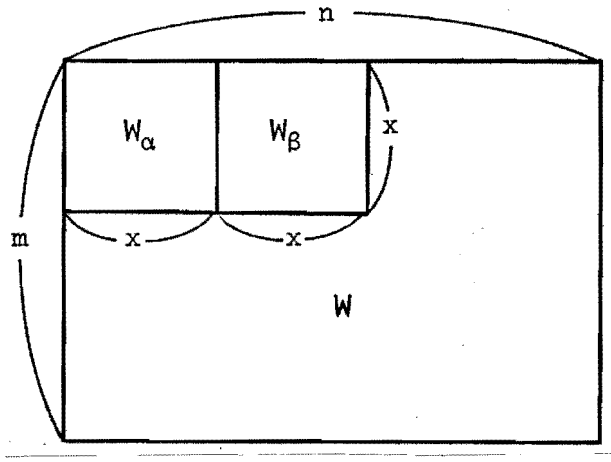

Fig. 2.2 Two chunks w_α and w_β on an input w.

If w is too small to take these chunks, then T_1 rejects w and halts. But, from Eq.(2.8), if $(m,n)=(m_i,n_i)$ for some i, these two chunks can be always taken on w. Next, T_1 examines whether $h_2(w_\alpha)=h_2(w_\beta)$ or not. T_1 can easily do this, using ℓ squares of the storage tape to keep the coordinate of each point of these chunks. If $h_2(w_\alpha)=h_2(w_\beta)$, T_1 accepts w.

Let L be the language accepted by T_1. Now we suppose that there exists $M \in 2TM(L_1(m,n))$ which accepts L. Let $\{(x_i,\ell_i)\}$ be a sequence of pairs of nonnegative integers such that

$$x_i = 2^{[[L_2(m_i,n_i)]/k]-1},$$

$$\ell_i = [L_1(m_i,n_i)].$$

Let v_i be a word on Σ' of size $m_i \times n_i$ which makes T' use exactly $[L_2(m_i,n_i)]$ squares of the storage tape. Let $v_{i\alpha}$ and $v_{i\beta}$ be two chunks of size x_i taken on v_i at the same position as in Fig.2.2. Let $\{D_i\}$ be a sequence of sets of chunks such that

$$D_i = \{c \mid c \in C_\Sigma^{x_i},\ h_1(c)=v_{i\alpha}\}.$$

Since $\{(x_i,\ell_i)\}$ and $\{D_i\}$ satisfy Eq.(2.3)-(2.5), there are two different (M,ℓ_i)-equivalent chunks $c_i,c_i' \in D_i$ for i large enough. Now, we consider two words $w_i,w_i' \in \Sigma^{2+}$ that satisfy following conditions. Let $w_{i\alpha},w_{i\beta},w_{i\alpha}',w_{i\beta}' \in C_\Sigma^{x_i}$ be chunks on w_i and w_i' taken at the same position as v_i. w_i and w_i' are the same words except the portions $w_{i\alpha}$ and $w_{i\alpha}'$. They satisfy $h_1(w_i)=h_1(w_i')=v_i$, $h_2(w_{i\beta})=h_2(w_{i\beta}')=h_2(c_i)$, $w_{i\alpha}=c_i$, $w_{i\alpha}'=c_i'$. Therefore $w_i \in L$ and $w_i' \notin L$. But on the other hand, $w_{i\alpha}$ and $w_{i\alpha}'$ are (M,ℓ_i)-equivalent, so M cannot distinguish w_i and w_i'. This is a contradiction. Thus $L \notin \mathcal{L}[2TM(L_1(m,n))]$ is concluded. (Q.E.D.)

Finally, we will prove a similar lemma for the case that the tape function L(m,n) grows more rapidly than the order of log m and more slowly than the order of log n (or vice versa) for some infinite set of (m,n). This can be proved by using the notion of $(T,\ell)^v$-equivalence (or $(T,\ell)^h$-equivalence) which is somewhat similar to (T,ℓ)-equivalence.

Definition. Let $W_{\Sigma}^{x,y}$ be a set of all the words of size x×y on Σ (without border symbols). Let u^v (or u^h) denote the word $u \in W_{\Sigma}^{x,y}$ with border symbols at the top and the bottom (or, at the right and the left) edges. u^v (u^h) is called a *vertical (horizontal) block*. Let T be a 2TM with s states and t storage tape symbols, and let $u_1, u_2 \in W_{\Sigma}^{x,y}$ be two words on Σ. T can enter or go out of the vertical block u_1^v or u_2^v through their right or left edges. So T has $2xs\ell t^{\ell}$ ways to enter or go out of them, if T uses at most ℓ squares of the storage tape. u_1 and u_2 are said to be *$(T,\ell)^v$-equivalent*, if T's behaviours to u_1^v and u_2^v are the same for all the ways of entering. (Since the notion of $(T,\ell)^h$-equivalence is similar to this, we only mention about $(T,\ell)^v$-equivalence in the following.)

Lemma 2.17 Let T be an arbitrary 2TM. Let $\{(x_i, y_i, \ell_i)\}$ be a sequence of triplets of nonnegative integers that satisfies

$$\lim_{i\to\infty} \frac{\ell_i}{\log y_i} = 0 \tag{2.9}$$

$$\lim_{i\to\infty} y_i = \infty \tag{2.10}$$

$$\frac{\log y_i}{\log x_i} > k \tag{2.11}$$

for some constant k>0. Let $\{U_i\}$ ($U_i \subseteq W_{\Sigma}^{x_i,y_i}$) be a sequence of sets of words that satisfies

$$|U_i| \geq r^{x_i y_i} \tag{2.12}$$

for some constant r>1. Then, for every sufficiently large i, there are two different words $u_i, u_i' \in U_i$ which are $(T,\ell_i)^V$-equivalent.

Proof. The total number of $(T,\ell_i)^V$-equivalence classes of $W_\Sigma^{x_i,y_i}$ is at most $(g_i+s+1)^{g_i}$, where $g_i = 2x_i s\ell_i t^{\ell_i}$. From Eq.(2.9)-(2.11), we can derive

$$\lim_{i\to\infty} \frac{(g_i+s+1)^{g_i}}{r^{x_i y_i}} = 0.$$

From Eq.(2.12), $|U_i| \geq r^{x_i y_i} > (g_i+s+1)^{g_i}$ holds for i large enough. So that there are two different $(T,\ell_i)^V$-equivalent words $u_i, u_i' \in W_\Sigma^{x_i,y_i}$ for such i. (Q.E.D.)

Lemma 2.18 Let $L_1(m,n)$ and $L_2(m,n)$ be constructible tape functions of 2TM. Suppose that

$$\lim_{i\to\infty} \frac{L_1(m_i,n_i)}{L_2(m_i,n_i)} = 0 \tag{2.13}$$

$$\lim_{i\to\infty} L_2(m_i,n_i) = \infty \tag{2.14}$$

$$\frac{L_2(m_i,n_i)}{\log m_i} > k_1 \quad \text{and} \quad \frac{L_2(m_i,n_i)}{\log n_i} < k_2 \tag{2.15}$$

$$\left[\text{or } \frac{L_2(m_i,n_i)}{\log m_i} < k_2 \quad \text{and} \quad \frac{L_2(m_i,n_i)}{\log n_i} > k_1\right]$$

for some regular sequence $\{(m_i,n_i)\}$ and some constants $0<2k_2<k_1<1$. Then there exists a language L such that $L \in \mathcal{L}[2TM(L_2(m,n))]$ but $L \notin \mathcal{L}[2TM(L_1(m,n))]$.

Proof. We construct $T_2 \in 2TM(L_2(m,n))$ which resembles T_1 in Lemma 2.16. Suppose a word $w \in (\Sigma' \times \{0,1\})^{2+}$ of size $m \times n$ is given to T_2. T_2 first simulates the movements of T' (in Lemma 2.16) with the input $h_1(w)$ until it halts. Let ℓ be the number of squares of the storage tape that T' has used. Let $y = 2^{[\ell/k_2]-1}$. If $n > 2y$, then we consider two subwords w_α and w_β of size $m \times y$ at the left side of w (Fig.2.3).

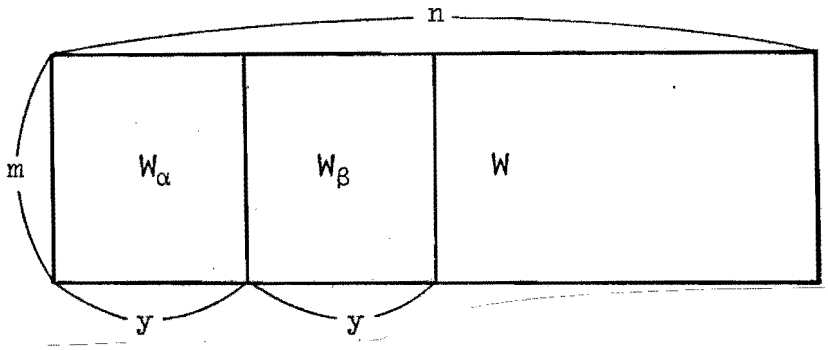

Fig. 2.3 Two subwords w_α and w_β on an input w.

T_2 examines whether $h_2(w_\alpha) = h_2(w_\beta)$, and accepts w if $h_2(w_\alpha) = h_2(w_\beta)$. Let L be the language accepted by T_2. Now we suppose that there exists $M \in 2TM(L_1(m,n))$ which accepts L. Let $\{(x_i, y_i, \ell_i)\}$ be a sequence such that

$$x_i = m_i,$$

$$y_i = 2^{[[L_2(m_i,n_i)]/k_2]-1},$$

$$\ell_i = L_1(m_i,n_i).$$

Let $v_i \in (\Sigma')^{2+}$ be a word of size $m_i \times n_i$ that makes T' use exactly $[L_2(m_i,n_i)]$ squares of the storage tape, and let $v_{i\alpha}$ and $v_{i\beta}$ be two subwords of size $x_i \times y_i$ on v_i. Since $U_i = \{u \mid u \in W_\Sigma^{x_i,y_i}, h_1(u) = v_{i\alpha}\}$, and $\{(x_i, y_i, \ell_i)\}$ satisfy (2.9)-(2.12),

there are two different $(M,\ell_i)^v$-equivalent subwords u_i, u_i' $\in U_i$ for i large enough. From these facts, we can make two words $w_i \in L$ and $w_i' \notin L$ which cannot be distinguished by M, in a similar way as in Lemma 2.16. And this concludes $L \notin \mathcal{L}[2TM(L_1(m,n))]$. (Q.E.D.)

From Lemma 2.14, 2.16, and 2.18, we can obtain the next hierarchy theorem, where there is no restriction on a growth of the tape function.

Theorem 2.19 (Hierarchy Theorem of 2TM) Let $L_1(m,n)$ and $L_2(m,n)$ be constructible tape functions of 2TM. Suppose that

$$\lim_{i\to\infty} \frac{L_1(m_i,n_i)}{L_2(m_i,n_i)} = 0$$

$$\lim_{i\to\infty} L_2(m_i,n_i) = \infty$$

for some regular sequence $\{(m_i,n_i)\}$. Then there exists a language L such that $L \in \mathcal{L}[2TM(L_2(m,n))]$ but $L \notin \mathcal{L}[2TM(L_1(m,n))]$.

Proof. Let $a_i=L_2(m_i,n_i)/(\log m_i + \log n_i)$, $b_i=L_2(m_i,n_i)/\log m_i$, $c_i=L_2(m_i,n_i)/\log n_i$. If there exists some $k>0$ such that $a_i>k$ holds for infinitely many i, we can apply Lemma 2.14 by taking a suitable subsequence of $\{(m_i,n_i)\}$. If there exists no such k, then $(1/a_i)=(1/b_i)+(1/c_i) > K$ holds for infinitely many i, for any $K>0$. Therefore, at least one of the following inequalities holds for appropreate

$K_0>0$, $K_1>1$, $K_2>2K_1$, for infinitely many i.

$(1/b_i) > K_0$ and $(1/c_i) > K_0$

$(1/b_i) < K_1$ and $(1/c_i) > K_2$

$(1/b_i) > K_2$ and $(1/c_i) < K_1$

For each case, Lemma 2.16 or 2.18 can be applied by taking a suitable subsequence of $\{(m_i,n_i)\}$. This completes the proof. (Q.E.D.)

The hierarchy theorem for $2TM^f(L(j))$ can also be proved likewise.

Corollary 2.20 Let $f : \mathbf{N} \to \mathbf{N}^2$ be a shape function, and let $L_1(j)$ and $L_2(j)$ be constructible tape functions of $2TM^f$. Suppose that

$$\lim_{i\to\infty} \frac{L_1(j_i)}{L_2(j_i)} = 0$$

for some increasing sequence of natural numbers $\{j_i\}$, and $L_2(j_i)$ is unbounded. Then there is a language L such that $L \in \mathcal{L}[2TM^f(L_2(j))]$ but $L \notin \mathcal{L}[2TM^f(L_1(j))]$.

If two constructible tape functions $L_1(m,n)$ and $L_2(m,n)$ are given, the relation between $\mathcal{L}[2TM(L_1(m,n))]$ and $\mathcal{L}[2TM(L_2(m,n))]$ must be one of the followings.

(1) $\mathcal{L}[2TM(L_1(m,n))] = \mathcal{L}[2TM(L_2(m,n))]$

(2) $\mathcal{L}[2TM(L_1(m,n))] \subsetneq \mathcal{L}[2TM(L_2(m,n))]$

(3) $\mathcal{L}[2TM(L_1(m,n))] \supsetneq \mathcal{L}[2TM(L_2(m,n))]$

(4) $\mathcal{L}[2TM(L_1(m,n))] \not\subseteq \mathcal{L}[2TM(L_2(m,n))]$ and $\mathcal{L}[2TM(L_1(m,n))] \not\supseteq \mathcal{L}[2TM(L_2(m,n))]$

($A \not\subset\not\supset B$ means that the sets A and B are incomparable, i.e. $A \not\subset B$ and $A \not\supset B$.) Now, we consider the necessary and sufficient conditions for them.

Lemma 2.21 Let $L_1(m,n)$ and $L_2(m,n)$ be constructible tape functions of 2TM. Then, only one of the following statements must be true.

(A) There exist some constants $c_1>0$, $c_1'>0$, $c_2>0$ and $c_2'>0$ that satisfy the following conditions.

$$L_1(m,n) \leq c_2 \cdot L_2(m,n) + c_2' \tag{2.16}$$

$$L_2(m,n) \leq c_1 \cdot L_1(m,n) + c_1' \tag{2.17}$$

(B) There exist some constants $c_2>0$ and $c_2'>0$, and some regular sequence $\{(m_i,n_i)\}$ that satisfy the following conditions.

$$L_1(m,n) \leq c_2 \cdot L_2(m,n) + c_2' \tag{2.18}$$

$$\lim_{i\to\infty} \frac{L_1(m_i,n_i)+1}{L_2(m_i,n_i)} = 0 \tag{2.19}$$

(C) There exist some constants $c_1>0$ and $c_1'>0$, and some regular sequence $\{(m_i,n_i)\}$ that satisfy the following conditions.

$$L_2(m,n) \leq c_1 \cdot L_1(m,n) + c_1' \tag{2.20}$$

$$\lim_{i\to\infty} \frac{L_2(m_i,n_i)+1}{L_1(m_i,n_i)} = 0 \tag{2.21}$$

(D) There exist some regular sequences $\{(m_i,n_i)\}$ and $\{(m_i',n_i')\}$ that satisfy the following conditions.

$$\lim_{i\to\infty} \frac{L_1(m_i,n_i)+1}{L_2(m_i,n_i)} = 0 \tag{2.22}$$

$$\lim_{i\to\infty} \frac{L_2(m_i',n_i')+1}{L_1(m_i',n_i')} = 0 \qquad (2.23)$$

Proof. Obviously, these statements are mutually exclusive. We consider the case that the statement (A) is untrue. If no such c_1 and c_1' exist, then $L_2(m,n) > c\cdot L_1(m,n) + c'$ holds for any $c>0$ and $c'>0$, for infinitely many (m,n). So, there exists some regular sequence $\{(m_i,n_i)\}$ such that

$$\lim_{i\to\infty} \frac{L_1(m_i,n_i)}{L_2(m_i,n_i)} = 0 \quad \text{and} \quad \lim_{i\to\infty} L_2(m_i,n_i) = \infty.$$

Thus the statement (B) becomes true. Similarly, if no such c_2 and c_2' exist, then the statement (C) becomes true. Furthermore, if no such c_1, c_1', c_2 and c_2' exist, the statement (D) holds. Accordingly, we can conclude that only one of these statements must be true. (Q.E.D.)

Theorem 2.22 Let $L_1(m,n)$ and $L_2(m,n)$ be constructible tape functions of 2TM, and (A), (B), (C) and (D) be the statements in Lemma 2.21.

(1) $\mathcal{L}[2TM(L_1(m,n))] = \mathcal{L}[2TM(L_2(m,n))]$, if and only if (A) holds.

(2) $\mathcal{L}[2TM(L_1(m,n))] \subsetneq \mathcal{L}[2TM(L_2(m,n))]$, if and only if (B) holds.

(3) $\mathcal{L}[2TM(L_1(m,n))] \supsetneq \mathcal{L}[2TM(L_2(m,n))]$, if and only if (C) holds.

(4) $\mathcal{L}[2TM(L_1(m,n))] \not\subseteq \not\supseteq \mathcal{L}[2TM(L_2(m,n))]$, if and only if (D) holds.

Proof. First, we show that (A), (B), (C) and (D) are necessary conditions for these cases.

(1) If (A) holds, then, from Theorem 2.10, 2.12 and Eq.(2.16), $\mathcal{L}[2TM(L_1(m,n))] \subseteq \mathcal{L}[2TM(c_2 \cdot L_2(m,n)+c_2')] = \mathcal{L}[2TM(L_2(m,n))]$. Similarly, from Eq.(2.17), $\mathcal{L}[2TM(L_2(m,n))] \subseteq \mathcal{L}[2TM(L_1(m,n))]$. Thus, $\mathcal{L}[2TM(L_1(m,n))] = \mathcal{L}[2TM(L_2(m,n))]$.

(2) From Eq.(2.18), $\mathcal{L}[2TM(L_1(m,n))] \subseteq \mathcal{L}[2TM(L_2(m,n))]$. And, fron Theorem 2.19 and Eq.(2.19), there exists a language L such that $L \in \mathcal{L}[2TM(L_2(m,n))]$ but $L \notin \mathcal{L}[2TM(L_1(m,n))]$. Thus, $\mathcal{L}[2TM(L_1(m,n))] \subsetneq \mathcal{L}[2TM(L_2(m,n))]$.

(3) It can be shown in a similar manner as in (2).

(4) From Theorem 2.19 and Eq.(2.22), there exists a language L such that $L \in \mathcal{L}[2TM(L_2(m,n))]$ but $L \notin \mathcal{L}[2TM(L_1(m,n))]$. But, from Eq.(2.23), there also exists a language L' such that $L' \in \mathcal{L}[2TM(L_1(m,n))]$ but $L' \notin \mathcal{L}[2TM(L_2(m,n))]$. Thus, $\mathcal{L}[2TM(L_1(m,n))] \not\subseteq \mathcal{L}[2TM(L_2(m,n))]$.

Now, it can be easily seen that (A), (B), (C) and (D) are also sufficient conditions for these cases. For instance, from Lemma 2.21, if (A) is false, then (B), (C) or (D) must be true. So, $\mathcal{L}[2TM(L_1(m,n))] = \mathcal{L}[2TM(L_2(m,n))]$ does not hold. Thus (A) becomes a sufficient condition in (1). Similarly, (B), (C) and (D) also become sufficient conditions. This completes the proof. (Q.E.D.)

Corollary 2.23 Let $f : \mathbf{N} \to \mathbf{N}^2$ be a shape function, and let $L_1(j)$ and $L_2(j)$ be constructible tape functions of $2TM^f$.

(1) $\mathcal{L}[2TM^f(L_1(j))] = \mathcal{L}[2TM^f(L_2(j))]$, if and only if there exist some constants $c_1>0$, $c_1'>0$, $c_2>0$ and $c_2'>0$ that satisfy the following conditions.

$$L_1(j) \leq c_2 \cdot L_2(j) + c_2'$$

$$L_2(j) \leq c_1 \cdot L_1(j) + c_1'$$

(2) $\mathcal{L}[2TM^f(L_1(j))] \subsetneq \mathcal{L}[2TM^f(L_2(j))]$, if and only if there exist some constants $c_2>0$, $c_2'>0$, and some increasing sequence of natural numbers $\{j_i\}$ that satisfy the following conditions.

$$L_1(j) \leq c_2 \cdot L_2(j) + c_2'$$

$$\lim_{i\to\infty} \frac{L_1(j_i)+1}{L_2(j_i)} = 0$$

(3) $\mathcal{L}[2TM^f(L_1(j))] \supsetneq \mathcal{L}[2TM^f(L_2(j))]$, if and only if there exist some constants $c_1>0$, $c_1'>0$, and some increasing sequence of natural numbers $\{j_i\}$ that satisfy the following conditions.

$$L_2(j) \leq c_1 \cdot L_1(j) + c_1'$$

$$\lim_{i\to\infty} \frac{L_2(j_i)+1}{L_1(j_i)} = 0$$

(4) $\mathcal{L}[2TM^f(L_1(j))] \not\subseteq \mathcal{L}[2TM^f(L_2(j))]$, if and only if there exist some increasing sequences of natural numbers $\{j_i\}$ and $\{j_i'\}$ that satisfy the following conditions.

$$\lim_{i\to\infty} \frac{L_1(j_i)+1}{L_2(j_i)} = 0$$

$$\lim_{i\to\infty} \frac{L_2(j_i')+1}{L_1(j_i')} = 0$$

2.5 Hierarchy Theorem of $2TM^1$

The hierarchy theorem obtained in the previous section holds only for the case of 2TM with two or more input symbols. In this section, we consider the case of one input symbol. The hierarchy theorem of $2TM^1$ can be proved by using the fact that $\mathcal{L}[2TM^1(L(m,n))]$ is closed under complementation when L(m,n) grows at least proportional to log m or log n. Thus, it also remains open when L(m,n) grows more slowly than both the orders of log m and log n. The proof of this theorem is based on the method of Hartmanis and Berman [15].

Theorem 2.24 (Hierarchy Theorem of $2TM^1$) Let $L_1(m,n)$ and $L_2(m,n)$ be constructible tape functions of $2TM^1$, and $L_1(m,n)$ satisfies $c \cdot L_1(m,n) \geq \log m$ (or $c \cdot L_1(m,n) \geq \log n$) for every m and n, and for some constant $c \geq 1$. Suppose that $L_2(m_i,n_i)$ is unbounded and

$$\lim_{i\to\infty} \frac{L_1(m_i,n_i)}{L_2(m_i,n_i)} = 0 \qquad (2.24)$$

for some recursively enumerable regular sequence $\{(m_i,n_i)\}$. Then there exists a language L such that $L \in \mathcal{L}[2TM^1(L_2(m,n))]$ but $L \notin \mathcal{L}[2TM^1(L_1(m,n))]$.

Proof. Let $\phi(i)=\max\{x_j \mid x_j=[L_2(m_j,n_j)],\ 1\leq j\leq i\}$. Since $\{(m_i,n_i)\}$ is a recursively enumerable sequence and $L_2(m_i,n_i)$ is a constructible tape function of $2TM^1$, $\phi : \mathbf{N} \to \mathbf{N}$ becomes a monotone nondecreasing total recursive function. Let $\psi : \mathbf{N} \to \mathbf{N}$ be a function defined as follows. Let $\Psi(j)$ be the

number of tape squares used to compute $\psi(j)$ by some Turing machine. (So, $\Psi(j)$ depends on the Turing machine which computes $\psi(j)$.)

$$\begin{cases} \psi(1) = \phi(1) \\ \psi(j+1) = \min\{\phi(i) \mid \psi(j)+\Psi(j) < \phi(i),\ i=1,2,\cdots\} \end{cases}$$

Thus $\psi(j)$ becomes a monotone increasing total recursive function, and $\psi(j)=\phi(i_j)$ for some increasing sequence of natural numbers $\{i_j\}$. Next, we define $g : \mathbf{N} \to \mathbf{N}$ from ψ.

$$g(x) = \max\{k \mid \psi(k)+\Psi(k) < x\}$$

Consider the partial mapping $g_A : A \to \mathbf{N}$, where

$$A = \{x \mid \exists i,\ x=[L_2(m_i,n_i)]\}.$$

(g_A is the same mapping as g, except that the domain is restricted to A.) It is easily seen that g_A is a surjection. Because, for all $\psi(j+1) \in A$ $(j=1,2,\cdots)$,

$$g_A(\psi(j+1)) = \max\{k \mid \psi(k)+\Psi(k) < \psi(j+1)\} = j.$$

We can see that there exists a one-dimensional one-tape Turing machine M which computes g(x) in the following manner. If x is given as a unary number, then M writes g(x) in a binary number using at most x squares of the tape. M can do this by computing $\psi(1)$, $\psi(2)$,$\cdots$ consecutively. Each time M computes $\psi(k)$ $(k=1,2,\cdots)$, M compares whether $\psi(k)+\Psi(k) \geq x$. (If M is ready to use more than x squares of the tape in the computation of $\psi(k)$, M can conclude $\psi(k)+\Psi(k) \geq x$ at this moment.) If M finds that k_0 is the first k that satisfies $\psi(k)+\Psi(k) \geq x$, M answers $g(x)=k_0-1$ and writes it in binary. Now, let $h : \mathbf{N} \to \mathbf{N}$ be a surjection which satisfies the following condition. For every $y \in \mathbf{N}$, there exist

infinitely many x such that h(x)=y. And h(x) can be computed by some one-dimensional one-tape Turing machine within [log n] squares of the tape, if x is given in binary. For example, it is easily seen that

$$h(x) = x - \max\{2^i \mid 2^i < x,\ i=0,1,\cdots\}$$

satisfies the above conditions.

Now, we construct $T \in 2TM^1(L_2(m,n))$ which diagonalizes all $2TM^1(L_1(m,n))$. Assume $T' \in 2TM^1(L_2(m,n))$ constructs the tape function $L_2(m,n)$. (T' always halts for any input.) Now, let us give T a word $w \in \{a\}^{2+}$ of size m×n. First T simulates T' and marks x squares of the storage tape, where $x=[L_2(m,n)]$. Then T simulates the one-dimensional one-tape Turing machine M to compute g(x) within x squares of the storage tape. Next, T computes h(g(x)), and then begins to simulate the h(g(x))-th $2TM^1$, say $T_{h(g(x))}$, with the input w within x squares of of the storage tape. T accepts w if $T_{h(g(x))}$ rejects it, and T rejects w if $T_{h(g(x))}$ accepts it (T loops if $T_{h(g(x))}$ loops). If T is ready to use more than x squares during the simulation of $T_{h(g(x))}$, then T rejects w. But, from Eq.(2.24), if $T_{h(g(x))}$ is a $2TM^1(L_1(m,n))$ and $(m,n)=(m_i,n_i)$ for some sufficiently large i, T can simulate $T_{h(g(x))}$ within x squares.

Let L be the language accepted by T. Assume that $T_j \in 2TM^1(L_1(m,n))$, the j-th $2TM^1$, accepts L. From Lemma 2.6, we may also assume, without loss of generality, T_j halts for any input. Since g_A is a surjection, there exist infinitely many $x \in A$ such that h(g(x)) = j. Thus, there also

exist infinitely many i such that an input $w \in \{a\}^{2+}$ of size $m_i \times n_i$ makes T simulate T_j with w. For sufficiently large such i, T can make an opposite answer to T_j, and this causes an contradiction. Thus $L \notin \mathcal{L}[2TM^1(L_1(m,n))]$. (Q.E.D.)

Corollary 2.25 Let $f : \mathbf{N} \to \mathbf{N}^2$ be a shape function. Let $L_1(j)$ and $L_2(j)$ be constructible tape functions of $2TM^{1f}$, and $L_1(j)$ satisfies $c \cdot L_1(j) \geq \log m$ (or $c \cdot L_1(j) \geq \log n$) for every input $w \in \{a\}^{2+}$, and for some constant $c \geq 1$, where j is the index of w and $f(j)=(m,n)$. Suppose that $L_2(j_i)$ is unbounded and

$$\lim_{i \to \infty} \frac{L_1(j_i)}{L_2(j_i)} = 0$$

for some recursively enumerable increasing sequence of natural numbers $\{j_i\}$. Then there exists a language L such that $L \in \mathcal{L}[2TM^{1f}(L_2(j))]$ but $L \notin \mathcal{L}[2TM^{1f}(L_1(j))]$.

2.6 Lower Bounds on Tape Growth

In this section, we consider lower bounds on tape growth of 2TM. As we have seen in Theorem 1.4, there exists a non-trivial lower bound log log n, in the one-dimensional case. But, we will show that there is no such lower bound, in the two-dimensional case. Actually, it will be seen that any total recursive function cannot be a lower bound of constructible tape functions.

We first consider the case of $2TM^{1f}$. For this purpose we show the next lemma.

Lemma 2.26 Let $\phi : \mathbf{N} \to \mathbf{N}$ be an arbitrary monotone nondecreasing function that satisfies $\lim_{n\to\infty} \phi(n) = \infty$. And let ψ and η be the following functions.

$$\psi(x) = \min\{n \mid \phi(n+1) > x\}$$

$$\eta(n) = \min\{x \mid \psi(x) \geq n\}$$

Then, $\phi(n) = \eta(n)$.

Proof. Since $\phi(n)$ is a monotone nondecreasing function such that $\lim_{n\to\infty} \phi(n) = \infty$, for every $n_0 \in \mathbf{N}$ there exist $n_1, n_2 \in \mathbf{N}$ which satisfy $n_1 \leq n_0 \leq n_2$ and $\phi(n_1-1) < \phi(n_1) = \phi(n_0) = \phi(n_2) < \phi(n_2+1)$. (For the sake of convenience, we assume $\phi(0)=0$.) Thus, for every n' such that $n_1 \leq n' \leq n_2$,

$$\psi(\phi(n')) = \min\{n \mid \phi(n+1) > \phi(n')\} = n_2$$

$$\psi(\phi(n_1-1)) = \min\{n \mid \phi(n+1) > \phi(n_1-1)\} = n_1-1.$$

And since $\psi(x)$ becomes monotone nondecreasing,

$$\min\{x \mid \psi(x) \geq n_0\} = \phi(n_0).$$

Accordingly, $\eta(n_0) = \phi(n_0)$. (Q.E.D.)

The following theorem states that for an arbitrary slow monotone nondecreasing total recursive function $\phi(j)$, there exists a tape function L(j) which is constructed by some $2TM^{1f}$ and grows more slowly than $\phi(j)$.

Theorem 2.27 Let $f : \mathbf{N} \to \mathbf{N}^2$ be a recursive shape

function (i.e. an effectively computable shape function) such that $\limsup_{j\to\infty}(\min(f_1(j),f_2(j))) = \infty$, where f_1 and f_2 are functions from **N** into **N** which satisfy $f(j)=(f_1(j),f_2(j))$. Let $\phi : \mathbf{N} \to \mathbf{N}$ be any monotone nondecreasing total recursive function such that $\lim_{j\to\infty} \phi(j) = \infty$. Then there exists a monotone nondecreasing tape function L(j) which is constructed by some $T \in 2TM^{lf}(L(j))$ and satisfies $L(j)<\phi(j)$ and $\lim_{j\to\infty} L(j) = \infty$.

Proof. First, let $\psi : \mathbf{N} \to \mathbf{N}$ be

$$\psi(x) = \min\{j \mid \phi(j+1)>x\}.$$

As $\phi(j)$ is a computable total function and $\lim_{j\to\infty} \phi(j) = \infty$, $\psi(x)$ is also a computable monotone nondecreasing total function and $\lim_{x\to\infty} \psi(x) = \infty$. Next let f' and $\hat{f}$ be functions from **N** into **N** defined as follows.

$$f'(j) = \min(f_1(j),f_2(j))$$

$$\hat{f}(j) = \max\{f'(i) \mid 1\leq i\leq j\}$$

Since f_1 and f_2 are computable and $\min(f_1(j),f_2(j))$ is unbounded, these functions are also computable and unbounded. By the way, it is known that a two-counter automaton† is universal, and actually, a two-counter automaton can simulate a k-counter automaton by coding k counters with a number $\prod_{i=1}^{k} p_i^{x_i}$, where x_i is the content of the i-th counter and p_i is the i-th prime (Minsky [29]). Thus we can construct a two-counter automaton M which computes the function $\hat{f}(\psi(x))$

---

† A two-counter automaton consists of a finite-state control and two counters each of which can count a nonnegative integer.

in the following manner. Let c_1 and c_2 be two counters of M, and let $|c_1|$ and $|c_2|$ denote the numbers they count. M begins the computation of $\hat{f}(\psi(x))$ from $|c_1|=2^x$, $|c_2|=0$, and halts with $|c_1|=2^x\times 3^{\hat{f}(\psi(x))}$, $|c_2|=0$. Now, we construct a somewhat different two-counter automaton M' from M. M' is an automaton that computes $\hat{f}(\psi(1))$, $\hat{f}(\psi(2))$,$\cdots$ and never halts. Let c_1' and c_2' be two counters of M'. M' starts its computation from $|c_1'|=0$, $|c_2'|=0$. First, setting $|c_1'|=2^1$, M' begins to simulate M and computes $\hat{f}(\psi(1))$. And M' finishes the simulation of M, resulting with $|c_1'|=2^1\times 3^{\hat{f}(\psi(1))}$, $|c_2'|=0$. Next, M' makes $|c_1'|=2^2$, and then starts to simulate M again to compute $\hat{f}(\psi(2))$. M' repeats this procedure infinitely many times. The *x-th series* of M' is a sequence of movements of M', begun with $|c_1'|=2^x$, $|c_2'|=0$, and finished with $|c_1'|=2^{x+1}$, $|c_2'|=0$, and $\hat{f}(\psi(x))$ is computed in this period. Let a(x) be the maximum number counted by c_1' and c_2' in the x-th series of M'. We design M' so that both counters counts a(x) in this period. Obviously, $a(x)>\hat{f}(\psi(x))$ for all x.

Now, we consider the following $2TM^{1f}$, T. Suppose an input $w\in\{a\}_f^{2+}$ of size $f_1(j)\times f_2(j)$ is given to T. T simulates the movements of M' series by series, as far as $\min(f_1(j),f_2(j))>a(x)$. T can do this by simulating the two counters of M' by the position of the input head of T. Every time M' finishes one series of movements, T marks a new square of the storage tape. Let L(j) be the tape function constructed by T. Then from Lemma 2.26,

$$L(j) = \min\{x \mid a(x)\geq f'(j)\} - 1$$

$$\leq \min\{x \mid a(x) \geq \hat{f}(j)\} - 1$$

$$\leq \min\{x \mid \hat{f}(\psi(x)) \geq \hat{f}(j)\} - 1$$

$$\leq \min\{x \mid \psi(x) \geq j\} - 1$$

$$= \phi(j) - 1.$$

Thus $L(j) < \phi(j)$ is concluded. Furthermore $L(j)$ is monotone nondecreasing and $\lim_{j\to\infty} L(j) = \infty$, because $f'(j)$ is an unbounded monotone nondecreasing total recursive function. (Q.E.D.)

Apparently the shape function $s(j)=(j,j)$ is a recursive mapping, so that the next corollary can be obtained.

Corollary 2.28 Let $\phi : \mathbf{N} \to \mathbf{N}$ be any monotone nondecreasing total recursive function such that $\lim_{m\to\infty} \phi(m) = \infty$. Then, there exists a monotone nondecreasing tape function $L(m)$ which is constructed by some $T \in 2TM^{ls}(L(m))$ and satisfies $L(m) < \phi(m)$ and $\lim_{m\to\infty} L(m) = \infty$.

For ϕ which is not a monotone nondecreasing function, we can derive the next corollary.

Corollary 2.29 Let $f : \mathbf{N} \to \mathbf{N}^2$ be a recursive shape function such that $\limsup_{j\to\infty}(\min(f_1(j), f_2(j))) = \infty$, where f_1 and f_2 are functions from $\mathbf{N}$ into $\mathbf{N}$ that satisfy $f(j)=(f_1(j), f_2(j))$. Let $\phi : \mathbf{N} \to \mathbf{N}$ be any total recursive function such that $\limsup_{j\to\infty} \phi(j) = \infty$. Then there exists a monotone nondecreasing tape function $L(j)$ which is constructed by some $T \in 2TM^{lf}(L(j))$ and satisfies $\liminf_{j\to\infty}(L(j)/\phi(j)) = 0$ and $\lim_{j\to\infty} L(j) = \infty$.

Proof. Let $\hat{\phi}(j) = \max\{\phi(k) \mid 1 \leq k \leq j\}$. $\hat{\phi}(j)$ becomes a monotone nondecreasing function and satisfies $\hat{\phi}(j_i)=\phi(j_i)$ for some increasing sequence of natural numbers $\{j_i\}$. Now, we apply Theorem 2.27 to $\hat{\phi}(j)$. But, here, by counting the number of serieses of M' on the storage tape in a binary number, T can construct the tape function L(j) such that $L(j) \leq \log(\hat{\phi}(j))$ and $\lim_{j\to\infty} L(j) = \infty$. But $L(j_i) \leq \log(\phi(j_i))$ holds for the sequence $\{j_i\}$, so that $\liminf_{j\to\infty} (L(j)/\phi(j)) = 0$ is concluded. (Q.E.D.)

Next, we consider the case of $2TM^1$ where the shape of the input is not restricted.

Definition. A function $\phi : \mathbf{N}^2 \to \mathbf{N}$ of two variables is said to be *monotone nondecreasing*, if $\phi(m,n) \leq \phi(m',n')$ holds for every (m,n) and (m',n') such that $m \leq m'$ and $n \leq n'$.

Theorem 2.30 Let $\phi : \mathbf{N}^2 \to \mathbf{N}$ be a monotone nondecreasing total recursive function such that $\lim_{n\to\infty} \phi(n,n) = \infty$. Then, there exists a monotone nondecreasing tape function L(m,n) which is constructed by some $T \in 2TM^1(L(m,n))$ and satisfies $L(m,n) < \phi(m,n)$ and $\lim_{i\to\infty} L(m_i,n_i) = \infty$ for any sequence of pairs of natural numbers $\{(m_i,n_i)\}$ such that $\lim_{i\to\infty} m_i = \infty$ and $\lim_{i\to\infty} n_i = \infty$.

Proof. Let $\phi'(m)=\phi(m,m)$. Then, from Corollary 2.28, there exists a monotone nondecreasing tape function L'(m)

which is constructed by some $T' \in 2TM^{ls}(L'(m))$ and satisfies $L'(m) < \phi'(m)$ and $\lim_{m\to\infty} L'(m) = \infty$. Now, we consider $T'' \in 2TM^{1}$ which is precisely the same as T' except that the inputs are not restricted to square-shaped ones. From Theorem 2.27, it is easily seen that if an input $w \in \{a\}^{2+}$ of size m×n is given to T", T" acts as if an input $w' \in \{a\}^{2+}_{s}$ of size min(m,n) ×min(m,n) is given, thus T" uses L'(min(m,n)) squares of the storage tape. So the tape function L(m,n) constructed by T" is as follows.

$$L(m,n) = \begin{cases} L'(m) & (\text{if } m \leqq n) \\ L'(n) & (\text{if } m > n) \end{cases}$$

Thus if $m \leqq n$, $L(m,n) = L'(m) < \phi'(m) = \phi(m,m) \leq \phi(m,n)$. In the case of m>n, $L(m,n) < \phi(m,n)$ holds as well. Furthermore, it is seen that $\lim_{i\to\infty} L(m_i,n_i) = \infty$ for any sequence $\{(m_i,n_i)\}$ such that $\lim_{i\to\infty} m_i = \infty$ and $\lim_{i\to\infty} n_i = \infty$, because $\lim_{m\to\infty} L'(m) = \infty$.

(Q.E.D.)

From above theorems, it is known that an arbitrarily slow tape function can be constructed in the range of total recursive functions by 2TM (even if the number of input symbols is only one). These results and the hierarchy theorem assures that there exists a dense hierarchy of language acceptabilities even below an arbitrarily small tape complexity. It contrasts strikingly with the fact that, in the one-dimensional case, there exists no class of tape complexity between the class of finite-state automata and the class of tape complexity log log n.

But, of course, there are many non-computable functions which grow more slowly than any constructible tape function. For example, $B^{-1}(n) = \min\{m \mid B(m) \geq n\}$ is such a function, where $B(m)$ is the busy beaver function defined by Rado [41]. Because $B(m)$ grows more rapidly than any recursive function.

2.7 Some Slowly Growing Tape Functions

As we have seen in the previous section, Theorem 2.27 and 2.30 guarantee that there exists an arbitrarily slow constructible tape function in the case of 2TM. But these theorems do not give concrete examples of slowly growing constructible tape functions. In this section, some examples of very slow constructible tape functions are shown. In what follows, we especially consider the case of $2TM^S$.

Let $W_1 \subset \{0,1\}_s^{2+}$ be a set of square-shaped words on $\{0,1\}$ defined as follows. Let $w(i,j)$ denotes the input symbol at the (i,j)-square (i.e. the square at the i-th row of the j-th column) of $w \in W_1$. If w is a word of size $m \times m$, then

$$w(i,j) = 1 \iff (i \in I_1 \wedge j \in I_2) \vee (i \in I_2 \wedge j \in I_1),$$

where $I_1=\{1,2,\cdots,m\}$, $I_2=\{2^1,2^2,\cdots,2^r\}$, $r=\max\{k \mid 2^k \leq m\}$. We can see that $2TM^S$ needs no square of the storage tape to accept W_1.

Lemma 2.31 $W_1 \in \mathcal{L}[2TM^S(0)]$.

Proof. We construct $M_1 \in 2TM^s(0)$ which accepts W_1 in the following manner. Assume an input $w \in \{0,1\}_s^{2+}$ is given to M_1

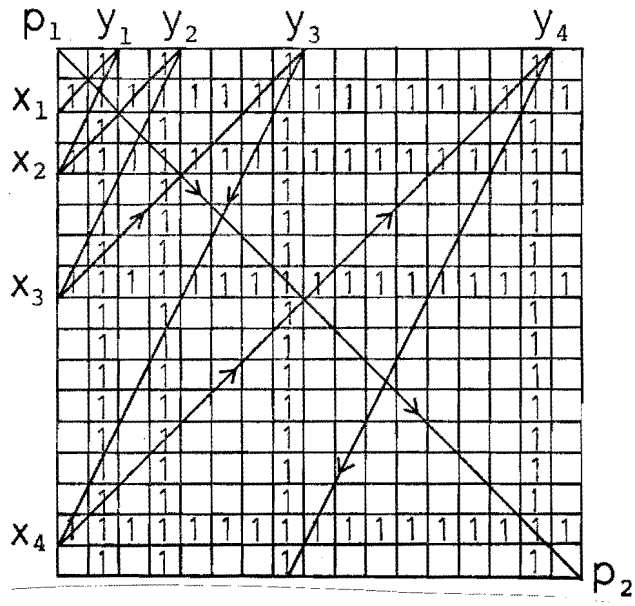

Fig. 2.4 Recognition of $w \in W_1$ by M_1.

(1) M_1 first scans w, row by row and column by column to check whether the squares in which the tape symbols 1's are written form cross stripes.

(2) Next, M_1 checks whether the numbers of vertical lines (i.e. vertical stretches of symbol "1") and horizontal lines are equal, and they are in the symmetrical positions (i.e. a vertical line lies in the i-th column $\Leftrightarrow$ a horizontal line lies in the i-th row). M_1 can do this by moving the input head from the north-west corner to the south-west corner (from p_1 to p_2 in Fig.2.4) to make sure that every

vertical line intersects some horizontal line on the diagonal line of the tape.

(3) Finally, M_1 checks whether a horizontal line lies only at every 2^k-th row ($k=1,2,\cdots$). Assume that M_1 has already ascertained that this property holds for the 1-st - 2^i-th row. M_1 now checks whether a horizontal line lies at the 2^{i+1}-th row but does not lie between the 2^i-th and the 2^{i+1}-th row, in the following way. Let x_i and y_i be the $(2^i,1)$-square and the $(1,2^i)$-square respectively. M_1 first brings the input head from x_i to y_i, and then moves it from y_i to x_{i+1} along a line of a slope 2. It is at the $(2^i,2^{i-1})$-square that M_1 first finds a vertical line on the way from y_i to x_{i+1}. M_1 then examines that a horizontal line lies at x_{i+1}, but does not lie between the $(2^i,2^{i-1})$-square and x_{i+1}. Starting from the point x_1, M_1 repeats this procedure until M_1 reads a border symbol of the bottom edge.

M_1 can accept W_1 by accepting the word w if and only if it satisfies the conditions examined in (1)-(3). (Q.E.D.)

Let $\log^{(k)}m$, $\exp^*m$ and $\log^*m$ be the functions from $(\mathbf{N}\cup\{0\})$ into $(\mathbf{N}\cup\{0\})$ which are defined as follows.

(1) $$\log^{(1)}m = \begin{cases} 0 & (m=0) \\ [\log m] & (m\geq 1) \end{cases}$$
$$\log^{(k+1)}m = \log^{(1)}(\log^{(k)}m)$$

(2) $$\exp^*0 = 1$$
$$\exp^*(m+1) = 2^{\exp^*m}$$

(3) $$\log^*m = \min\{x \mid \exp^*x \geq m\}$$

The tape functions $\log^{(k)}m$, $\log^{*}m$, and $\log^{(1)}\log^{*}m$ can be constructed by some $2TM^{S}$ with two input symbols.

Theorem 2.32 Let $\Sigma=\{0,1\}$. There exist $T_k \in 2TM^{S}(\log^{(k)}m)$, $T \in 2TM^{S}(\log^{*}m)$, and $T' \in 2TM^{S}(\log^{(1)}\log^{*}m)$ which construct the tape functions $\log^{(k)}m$, $\log^{*}m$, and $\log^{(1)}\log^{*}m$ respectively. $(k=1,2,\cdots)$

Proof. First, we construct $M_2 \in 2TM^{S}(0)$ which behaves as follows. Assume that $w \in W_1$ is given to M_2. If M_2 begins its movements setting the input head at the $(x,1)$-square of w, then it halts at the $(\log^{(1)}x,1)$-square. M_2 can be easily made up, if we notice that there are $\log^{(1)}x$ horizontal lines above the x-th row of w. Moving upwords square by square from the $(x,1)$-square, M_2 shifts the input head by one square to the right whenever M_2 meets a horizontal line. Thus M_2 can reach the $(1,\log^{(1)}x)$-square. Then M_2 moves the input head to the south-west direction along a line of a slope 1, and halts at the $(\log^{(1)}x,1)$-square.

$T_k \in 2TM^{S}(\log^{(k)}m)$ which constructs the tape function $\log^{(k)}m$ acts as follows. If an input $w \in \{0,1\}_{s}^{2+}$ of size $m\times n$ is given, T_k first simulates M_1 in Lemma 2.31 to examine whether $w \in W_1$. If $w \notin W_1$, T_k immediately halts. If otherwise, T_k brings the input head to the $(m,1)$-square, and simulates M_2, k times over again. By this, the input head of T_k comes to the $(\log^{(k)}m,1)$-square. Then T_k moves the input head to the $(1,1)$-square along the first column. By moving the

storage tape head synchronously with the input head, T_k can use $\log^{(k)} m$ squares of the storage tape.

We can also make up $T \in 2TM^S(\log^* m)$ and $T' \in 2TM^S(\log^{(1)} \log^* m)$ which construct the tape functions $\log^* m$ and $\log^{(1)} \log^* m$, in a similar way as in T_k. T first simulates M_1 to check whether $w \in W_1$. If $w \in W_1$, then T begins to simulate M_2 from the (m,1)-square. T simulates M_2 again and again until T comes to the (1,1)-square. T can mark $\log^* m$ squares of the storage tape by counting the number of times of the simulations of M_2 in an unary number. Similarly, T' can construct the tape function $\log^{(1)} \log^* m$ in the same way as T, except that T' counts this number in binary. (Q.E.D.)

From Corollary 2.23, the hierarchy of language acceptabilities among these tape complexity classes can be derived.

Theorem 2.33 $\mathcal{L}[2TM^S(0)] \subsetneq \mathcal{L}[2TM^S(\log^{(1)} \log^* m)] \subsetneq \mathcal{L}[2TM^S(\log^* m)] \subsetneq \mathcal{L}[2TM^S(\log^{(k+1)} m)] \subsetneq \mathcal{L}[2TM^S(\log^{(k)} m)]$. $(k=1,2,\cdots)$

Proof. We only consider the relation between $\mathcal{L}[2TM^S(\log^{k+1)} m)]$ and $\mathcal{L}[2TM^S(\log^{(k)} m)]$. The other relations can be proved likewise. Let $\{m_i\}$ be an increasing sequence of natural numbers such that $m_i = i + \exp^* k$. Then,

$$\lim_{i \to \infty} \frac{\log^{(k+1)} m_i}{\log^{(k)} m_i} = 0.$$

And $\log^{(k+1)}m \leq \log^{(k)}m$ holds, thus from Corollary 2.23, $\mathcal{L}[2TM^s(\log^{(k+1)}m)] \subsetneq \mathcal{L}[2TM^s(\log^{(k)}m)]$ is concluded.

(Q.E.D.)

2.8 Concluding Remarks

In this chapter, a tape-bounded two-dimensional Turing machine was introduced, and its basic properties were investigated. Especially, two important theorems were proved here. They are the hierarchy theorem (Theorem 2.19) and the theorem concerning the lower bounds on tape growth (Theorem 2.27). These theorems assert that there exists a dense hierarchy of acceptabilities among 2TM.

Here, we defined the tape function L(m,n) as a function of the horizontal and vertical sidelengths of the input, rather than a function of the area of it. Because, not only the former definition implies the latter one, but also we can characterize even a class of two-dimensional tape automata with non-isotropic computational powers such as parallel-sequential array automaton (Rosenfeld and Milgram [44]) by this definition (this is discussed in chapter 3). Furthermore, by considering the shape function $f : \mathbf{N} \to \mathbf{N}^2$, we can easily and uniformely investigate the acceptabilities of 2TM on the restricted input tapes.

The model of 2TM defined here has only one one-dimensional storage tape. However, when we consider the tape

complexity, the number or the dimension of the storage tape does not affect the acceptability of 2TM. Actually, the other models such as ones with multi storage tapes or a multi-dimensional storage tape can be simulated by the original model, because the number of storage tape symbols is arbitrary. So, this model seems to be reasonable. But, if we consider the time complexity of 2TM, the number or the dimension of the storage tape will become important.

In section 2.7, some examples of very slowly growing constructible tape functions were shown. The function $\log^{(1)}\log^{*}m$ was the slowest tape function in these examples. Of course, from Corollary 2.28, we can construct a tape function which grows more slowly than $\log^{(1)}\log^{*}m$. But, its concrete example (an example of a familiar function) is not known.

CHAPTER 3

SEVERAL TWO-DIMENSIONAL TAPE AUTOMATA

In the previous chapter, we introduced a two-dimensional Turing machine as a mathematical model of two-dimensional information processing. But, there are still many other models of two-dimensional tape automata. The notion of tape complexity is, in some sense, a special measure of computational complexity, so that we cannot always characterize any automaton by it. However, in the two-dimensional case, there exist many automata which are characterized by means of tape complexity. In this chapter, language acceptabilities of several such automata are investigated systematically from the standpoint of tape complexity.

3.1 Definitions

In this section we define several two-dimensional tape automata. They are a two-dimensional finite-state automaton, a two-dimensional multi-head automaton, a two-dimensional multi-marker automaton, a two-dimensional linear-bounded

automaton, a parallel-sequential array automaton, and some variants of a parallel-sequential array automaton. These automata also have rectangle-shaped inputs, and we regard them as acceptors of two-dimensional languages.

3.1.1 Two-Dimensional Finite-State Automaton (2FSA)

A two-dimensional finite-state automaton [5] consists of a rectangle-shaped input tape (which is the same as in 2TM), a four-way read-only input head, and a finite-state control.

Formally a *deterministic two-dimensional finite-state automaton* is defined as a 6-tuple

$$M = (K, \Sigma, \delta, q_0, B, F),$$

where K is a nonempty finite set of states, Σ is a nonempty finite set of input symbols, $q_0 \in K$ is an initial state, B is a border symbol of the input tape, and $F \subseteq K$ is a set of final states. δ is a mapping from a subset of $K \times (\Sigma \cup \{B\})$ into $K \times \{N,S,E,W,H\}$, where $\{N,S,E,W,H\}$ is a set of move directions of the input head. M halts for the element of $K \times (\Sigma \cup \{B\})$ on which δ is not defined.

The class of deterministic two-dimensional finite-state automata is denoted by 2FSA. The notion of the acceptance (of a word or a language) is similar to the case of 2TM, so its definition is omitted here.

3.1.2 Two-Dimensional Multi-Head Automaton (2MHA)

A two-dimensional multi-head automaton [23] consists of a rectangular input tape, a finite number of four-way read-only input heads, and a finite-state control.

Formally a *deterministic two-dimensional multi-head automaton* is a 7-tuple

$$M = (K, \Sigma, \delta, k, q_0, B, F),$$

where K, Σ, q_0, B, and F are the same as in 2FSA, k is the number of input heads, and δ is a mapping from a subset of $K\times(\Sigma \bigcup \{B\})^k$ into $K\times\{N,S,E,W,H\}^k$. M halts for the element of $K\times(\Sigma \bigcup \{B\})^k$ on which δ is not defined. Note that each input head cannot sense the other heads even if they are on the same square.

The class of deterministic k-head automata is denoted by 2MHA(k), and $\bigcup_{k=1}^{\infty}$ 2MHA(k) is denoted by 2MHA.

3.1.3 Two-Dimensional Multi-Marker Automaton (2MMA)

A two-dimensional multi-marker automaton [5,23] is a 2FSA with a finite number of markers which can be placed on the input tape.

Formally a *deterministic two-dimensional multi-marker automaton* is a 7-tuple

$$M = (K, \Sigma, \delta, k, q_0, B, F),$$

where K, Σ, q_0, B, and F are the same as in 2FSA, and k is the number of markers. But, K must be a direct product of some finite set K' and $\{0,1,\cdots,k\}$ (i.e. $K=K'\times\{0,1,\cdots,k\}$), and $q_0=(q_0', k)$ for some $q_0' \in K'$. The integer $i \in \{0,1,\cdots,k\}$ maens the number of markers possessed by the finite-state

control of M. δ is a mapping from a subset of $K\times(\Sigma\cup\{B\})\times\{0,1\}$ into $K\times\{N,S,E,W,H\}\times\{0,1\}$. 0 and 1 mean the absence and the existence of a marker on the tape square that M is scanning. Furthermore, δ must satisfy the following conditions. Suppose

$$\delta((q'_m,i),a,x) = ((q'_n,j),d,y),$$

for some $q'_m,q'_n \in K'$, $i,j \in \{0,1,\cdots,k\}$, $a \in \Sigma$, $x,y \in \{0,1\}$, $d \in \{N,S,E,W,H\}$.

(1) If (x=0 and y=0) or (x=1 and y=1), then i=j.

(2) If x=0 and y=1, then i-1=j.

(3) If x=1 and y=0, then i+1=j.

The class of deterministic two-dimensional k-marker automata is denoted by 2MMA(k), and $\bigcup_{k=1}^{\infty}$ 2MMA(k) is denoted by 2MMA.

3.1.4 Two-Dimensional Linear-Bounded Automaton (2LBA)

A two-dimensional linear-bounded automaton [2,61] consists of a rectangular input tape, a finite-state control, and a four-way read/write head which can freely read or write a tape symbol on the input tape (Fig.3.1).

Formally a *deterministic two-dimensional linear-bounded automaton* (2LBA) is defined as a 7-tuple

$$M = (K, \Sigma, \Gamma, \delta, q_0, B, F),$$

where K, Σ, q_0, B, and F are the same as in 2FSA, $\Gamma \supseteq \Sigma$ is a finite set of tape symbols, and δ is a mapping from a subset of $K\times(\Gamma\cup\{B\})$ into $K\times(\Gamma\cup\{B\})\times\{N,S,E,W,H\}$.

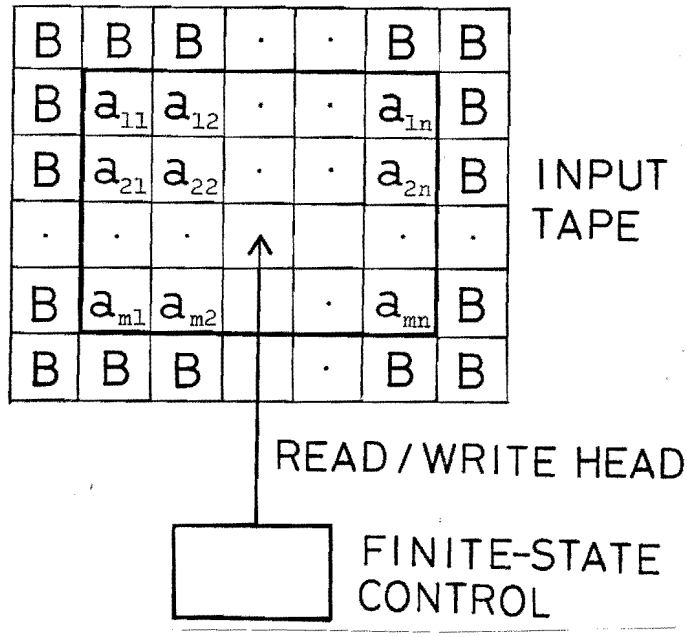

Fig. 3.1 A two-dimensional linear-bounded automaton.

3.1.5 Parallel-Sequential Array Automaton (PSA)

A parallel-sequential array automaton [44] consists of a rectangular input tape and a scanner which can move vertically on the tape in two ways (Fig.3.2). The input tape is divided into m×n squares, and border symbols are attached to the north and the south edges of it. The scanner is a horizontal array of n identical finite-state automata (each of which is called a *cell*) C^1, C^2,$\cdots$, C^n. (Hence the number of cells depends on the size of an input. The i-th cell C^i is positioned at the i-th column of the tape, thus C^i can read the tape squares of the i-th column. C^i

transits its state depending on the present states of C^{i-1}, C^i, C^{i+1} and the input symbol that C^i is reading. And C^1 determines the shift direction of the scanner depending on its state.

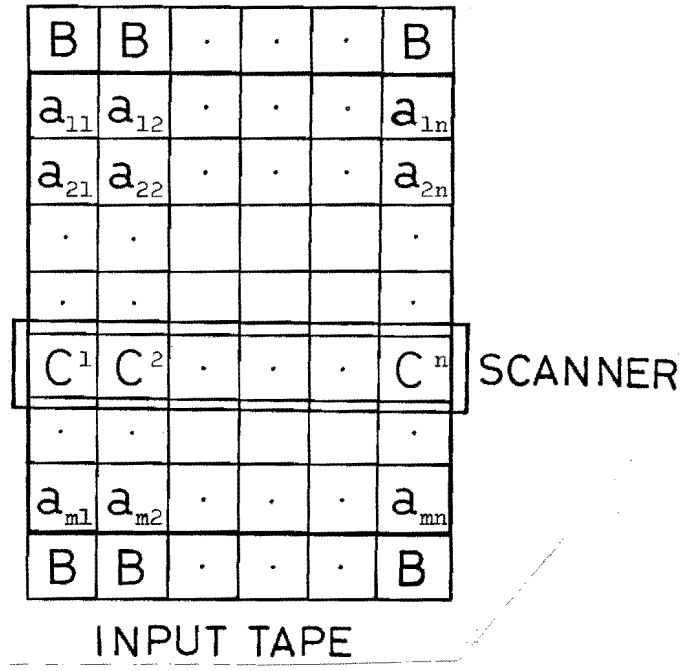

Fig. 3.2 A parallel-sequential array automaton.

Formally a *deterministic parallel-sequential array automaton* (PSA) is an 8-tuple

$$M = (Q, \Sigma, \delta, \lambda, q_0, B, e, F),$$

where Q is a nonempty finite set of states of each cell, Σ is a nonempty finite set of input symbols, $q_0 \in Q$ is an initial state of each cell, B is a border symbol of the input tape, e is an edge-state symbol for the edge cells C^1 and C^n $(Q \cap \{e\} = \emptyset)$, and $F \subseteq Q$ is a set of final states. δ is a mapping from $(Q \cup \{e\}) \times (Q-F) \times (Q \cup \{e\}) \times (\Sigma \cup \{B\})$ into Q.

The edge cell C^1 (or C^n) transits its state as if there is a cell of state e to the left (or right) of it. λ is a mapping from (Q-F) into {N,S,H}, where {N,S,H} is a set of shift directions of the scanner. One step of movements of M is advanced in the following manner : (1) apply δ to all the cells simultaneously, and then (2) apply λ to the cell C^1.

Suppose an input $w \in \Sigma^{2+}$ of size m×n is given to M, and M starts its movements setting the scanner at the first row with all the cells in q_0. We say that M accepts w, if the cell C^1 halts in a final state.

3.1.6 Multi-Scanner Parallel-Sequential Array Automaton (MPSA)

We now propose a multi-scanner parallel-sequential array automaton as a variant of PSA. It consists of a rectangular input tape, a finite-state control, and a finite number of scanners (S_1, S_2,$\cdots$,S_k) each of which can move vertically on the tape in two ways (Fig.3.3). If the size of the input is m×n, then each scanner S_i consists of n identical cells (C_i^1, C_i^2,$\cdots$, C_i^n). Each cell in S_i acts in the same way as in PSA. The finite-state control determines its next state and the shift directions of the scanners depending on its present state and the present states of the leftmost cells of the scanners.

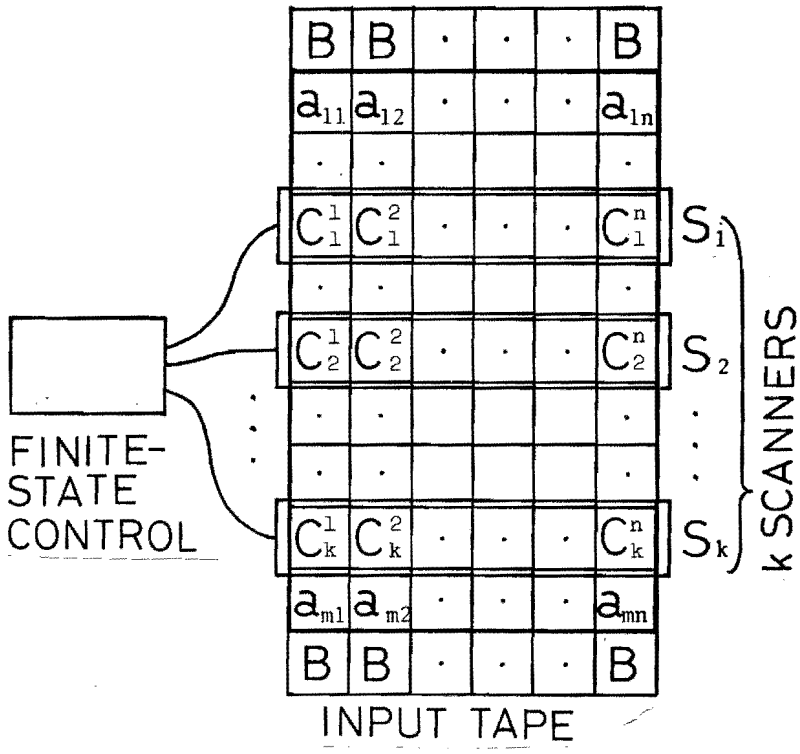

Fig. 3.3 A multi-scanner parallel-sequential array automaton.

Formally a *deterministic multi-scanner parallel-sequential array automaton* is a 12-tuple

$$M = (P, Q, \Sigma, \delta, \delta', \lambda, k, p_0, q_0, B, e, F),$$

where P is a nonempty finite set of states of the finite-state control, Q is a nonempty finite set of states of each cell, Σ is a nonempty finite set of input symbols, k is the number of scanners, $p_0 \in P$ is an initial state of the finite-state control, $q_0 \in Q$ is an initial state of each cell, B is a border symbol of the input tape, e is an edge-state symbol for the rightmost cell in each scanner ($Q \cap \{e\} = \emptyset$), $F \subseteq P$ is a set of final states (of the finite-state control). δ is

a mapping from $Q^2 \times (Q \cup \{e\}) \times (\Sigma \cup \{B\})$ into Q, which determines the next state of each cell except the leftmost cells. δ' is a mapping from $P \times Q \times (Q \cup \{e\}) \times (\Sigma \cup \{B\})$ into Q, which is applied to the leftmost cells. λ is a mapping from a subset of $P \times Q^k$ into $P \times \{N,S,H\}^k$. One step of movements of M is advanced by applying δ and δ' to all the cells simultaneously and then λ to the finite-state control.

The class of deterministic k-scanner parallel-sequential array automata is denoted by MPSA(k), and $\bigcup_{k=1}^{\infty} \text{MPSA}(k)$ is denoted by MPSA.

Suppose an input $w \in \Sigma^{2+}$ of size m×n is given to M, and M starts its movements from the initial state p_0, setting all the scanners at the first row and all the cells in q_0. If the finite-state control eventually halts in a final state, w is said to be accepted by M.

3.1.7 Orthogonal Multi-Scanner Parallel-Sequential Array Automaton (OMPSA)

Finally we define an orthogonal multi-scanner parallel-sequential array automaton as another variant of PSA. It consists of a rectangular input tape, a finite-state control, and a finite number of vertical scanners ($S_{V1}, \cdots, S_{Vk}$) and horizontal scanners ($S_{H1}, \cdots, S_{H\ell}$) (Fig.3.4). The vertical (or horizontal) scanner can move vertically (or horizontally) on the input tape in two ways.

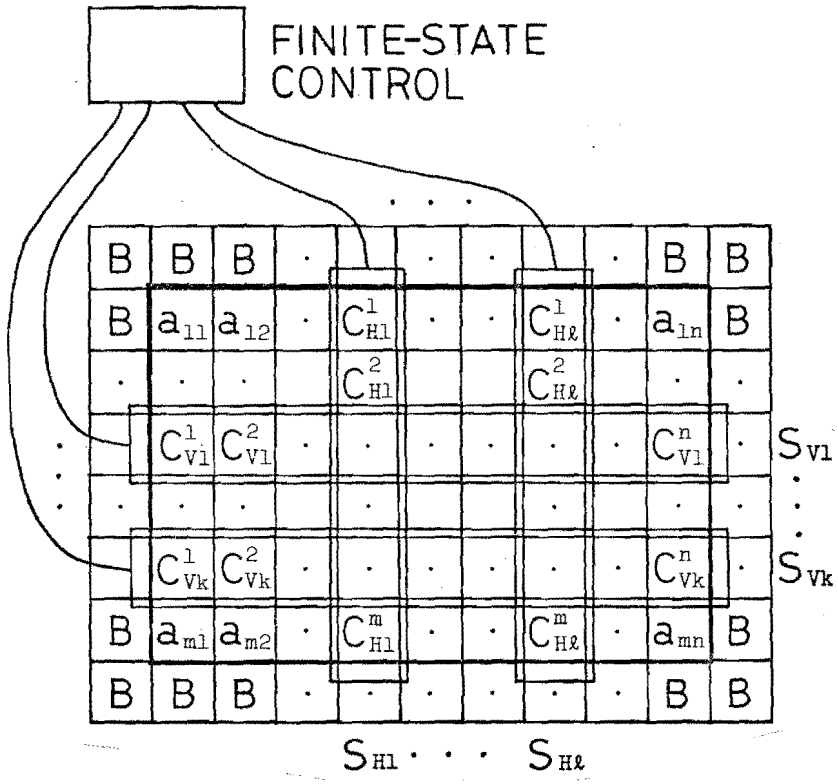

Fig. 3.4 An orthogonal multi-scanner parallel-sequential array automaton.

Formally a *deterministic orthogonal multi-scanner parallel-sequential array automaton* is a 13-tuple

$M = (P, Q, \Sigma, \delta, \delta', \lambda, k, \ell, p_0, q_0, B, e, F)$,

where, P, Q, Σ, p_0, q_0, B, e, and F are the same as in MPSA, and k and ℓ are the numbers of the vertical and horizontal scanners respectively. δ is a mapping from $Q^2\times(Q\cup\{e\})\times(\Sigma\cup\{B\})$ into Q, which determines the next state of each cell except the leftmost cells (in the vertical scanners) and the topmost cells (in the horizontal scanners). δ' is a mapping from $P\times Q\times(Q\cup\{e\})\times(\Sigma\cup\{B\})$ into Q, which determines the next states of the leftmost and the topmost cells. λ is a mapping

from a subset of $P\times Q^k\times Q^\ell$ into $P\times\{N,S,H\}^k\times\{E,W,H\}^\ell$, where {N,S,H} and {E,W,H} are the sets of move directions of vertical and horizontal scanners respectively.

The class of deterministic orthogonal multi-scanner parallel-sequential array automata with k vertical and ℓ horizontal scanners is denoted by OMPSA(k,ℓ), and $\bigcup_{k=1}^{\infty}\bigcup_{\ell=1}^{\infty}$ OMPSA(k,ℓ) is denoted by OMPSA.

Especially, OMPSA(1,1) is called a *deterministic orthogonal two-scanner parallel-sequential array automaton*, and denoted by O2PSA (Fig.3.5).

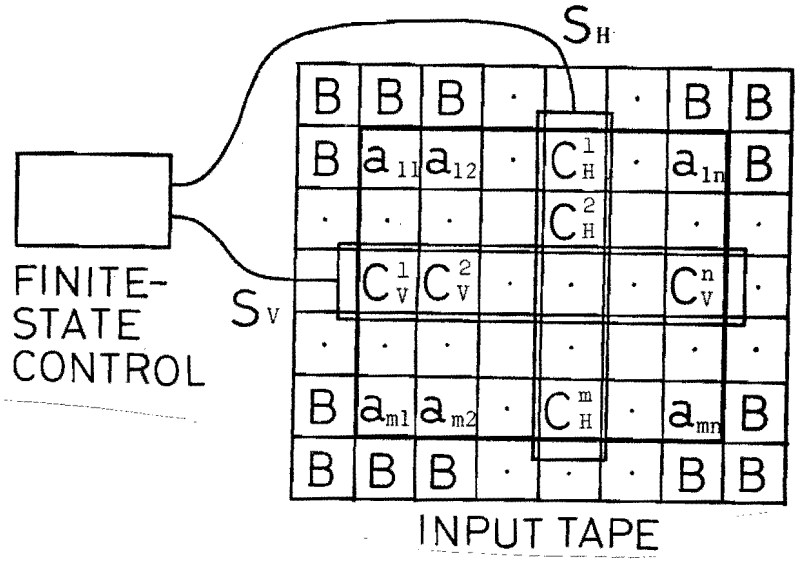

Fig. 3.5 An orthogonal two-scanner parallel-sequential array automaton.

3.2 Language Acceptabilities and Tape Complexities

Now we investigate the language acceptabilities of the automata defined in the previous section from the viewpoint of the tape complexity.

As for 2FSA, it is easily seen that the next theorem holds.

Theorem 3.1 $\mathcal{L}[2FSA] = \mathcal{L}[2TM(c)]$. (c : constant)

Next, we consider the language acceptabilities of 2MHA and 2MMA. The following theorem shows the relation between 2MHA(k) and 2MMA(k).

Theorem 3.2† $\mathcal{L}[2MHA(k)] \subseteq \mathcal{L}[2MMA(k)] \subseteq \mathcal{L}[2MHA(k+1)]$. (k=1,2,$\cdots$)

Proof. $\mathcal{L}[2MHA(k)] \subseteq \mathcal{L}[2MMA(k)]$ can be easily derived. Because, every $H \in 2MHA(k)$ can be simulated by some $M \in 2MMA(k)$ by placing k markers at the positions of k heads of H.

Conversely, for every $M' \in 2MMA(k)$, there exists some $H' \in 2MHA(k+1)$ which simulates M'. Let (u_i, v_i) and (u_{k+1}, v_{k+1}) represent the coordinates of the i-th marker and the input

---

† Ibarra and Melson [23] proved a similar theorem, where every head of 2MHA can sense each other. But in this case, it cannot sense each other.

head of M' respectively (i=1,2,$\cdots$,k). Similarly, let (x_i,y_i) represent the coordinate of the i-th head of H' (i=1,2,$\cdots$, k+1). The head of M' is simulated by the (k+1)-th head of H' (i.e. H' shifts the (k+1)-th head so as to become $(x_{k+1}, y_{k+1})=(u_{k+1},v_{k+1})$). The absolute value of the relative position of the i-th marker to the head of M' is kept by the i-th head of H' (i=1,2,$\cdots$,k) (i.e. H' shifts its heads so as to become $(x_i,y_i)=(|u_{k+1}-u_i|+1,|v_{k+1}-v_i|+1)$), and its direction (i.e. $\mathrm{sign}(u_{k+1}-u_i)$ and $\mathrm{sign}(v_{k+1}-v_i)$) is kept in the finite-state control. So, if the i-th head of H' is at the (1,1)-square, H' knows that the head of M' is reading the i-th marker or M' is holding it in the finite-state control (H' also remembers the markers that M' is holding, in the finite-state control). Thus, H' can simulate M' step by step, and accepts the input if and only if M' accepts it. (Q.E.D.)

Now, the relation among 2MHA, 2MMA, and 2TM(log mn) is shown.

Theorem 3.3 $\mathcal{L}$[2MHA] = $\mathcal{L}$[2MMA] = $\mathcal{L}$[2TM(log mn)]

Proof. $\mathcal{L}$[2MHA] = $\mathcal{L}$[2MMA] is easily seen from Theorem 3.2. So we prove $\mathcal{L}$[2MHA] = $\mathcal{L}$[2TM(log mn)].

For any M $\in$ 2MHA, we will construct T $\in$ 2TM(log mn) which accepts the same language as M. If an input w of size m×n is given to T, T first marks [log mn] squares of the storage tape by counting the total number of squares of w in binary. The

finite-state control of M is simulated by that of T. If M has k heads, T divides the storage tape into k tracks and keeps the coordinate of each head of M in each track with a binary number. By this, T can simulate M step by step, and it can accept the same language as M. Thus, $\mathcal{L}$[2MHA] $\subseteq$ $\mathcal{L}$[2TM(log mn)].

Next, for any T' $\in$ 2TM(log mn), M' $\in$ 2MHA which simulates T' is constructed in the following way. Suppose the storage tape of T' consists of k binary tracks (i.e. T' has at most 2^k storage tape symbols). Thus we can think that the i-th track (i=1,2,$\cdots$,k) contains a binary number x_i ($0 \leq x_i \leq mn-1$), where the leftmost bit is regarded as the least significant bit. M' remembers the number x_i by the position of the i-th head on the input. The (k+1)-th head and the (k+2)-th head of M' are used to keep the positions of the input head and the storage tape head of T'. If the (k+2)-th head shows that the storage tape head of T' is at the j-th square (j=1, 2,$\cdots$,[log mn]), M' divides each x_i by 2, j times, in order to know the tape symbol that the storage tape head is reading. This procedure can be done by some additional heads of M'. And M' can alter the j-th bit of x_i by adding (or subtracting) 2^{j-1} to (from) x_i. By this, M' can simulate T' and $\mathcal{L}$[2MHA] $\supseteq$ $\mathcal{L}$[2TM(log mn)] is concluded. (Q.E.D.)

As for 2LBA, the following theorem can be obtained.

Theorem 3.4 $\mathcal{L}$[2LBA] = $\mathcal{L}$[2TM(mn)].

Proof. For any $M \in 2LBA$, we can construct $T \in 2TM(mn)$ which simulates M. Suppose an input w of size m×n is given to T. Scanning w row by row, T converts it into an one-dimensional string of input symbols with punctuations (this punctuation means the end of a row), and writes it in the storage tape. Then T starts to simulate M on the storage tape. If M rewrites the tape symbol and shifts the input head one square to the east (or west), T also rewrites the storage tape symbol and shifts the storage tape head one square to the right (or left). If M shifts the input head to the south (or north), T must shift the storage tape head m squares to the right (or left). To do this, T counts the number m by moving the input head along the first column of the input. By this, T can simulate M step by step, and accepts w if and only if M accepts it.

Conversely, for any $T' \in 2TM(\log mn)$, there is some $M' \in$ 2LBA that simulates T'. In each square of the input tape of M', a combined symbol of the input symbol and the storage tape symbol of T' is written. Furthermore, the positions of the input head and the storage tape head are also marked on it. Thus, M' can easily simulate T' on the m×n squares of the input tape. (Q.E.D.)

Next, we investigate the language acceptabilities of PSA, MPSA, OMPSA, and O2PSA.

Theorem 3.5 $\mathcal{L}[PSA] = \mathcal{L}[2TM(n)]$.

Proof. First, for any $M \in PSA$, $T \in 2TM(n)$ which simulates M is constructed in the following way. T records the internal states of n cells of M using n squares of the storage tape. And T remembers the position of the scanner by placing the input head in the row where the scanner is positioned. In order to simulate one step of M, T reads the input symbols of the row that M is scanning one by one, and rewrites the internal states of the cells of M written in the storage tape according to the transition rule of M. T can accept the same language as M by accepting the input if and only if the leftmost cell of M becomes a final state.

Next, for any $T' \in 2TM(n)$, $M' \in PSA$ which accepts the same language as T' is constructed . The contents of n squares of the storage tape of T' are remembered by internal states of n cells of M'. The position of the storage tape head is also remembered by the cell at the corresponding position. The position of the input head of T' is simulated as follows. If the input head is at the (i,j)-square, then M' places the scanner in the i-th row and makes the j-th cell C^j be in a special state. The finite-state control of T' is simulated by the leftmost cell C^1. In order to advance one step of movements of T', the two cells, which are in the corresponding positions to the input head and the storage tape head, read the input symbol and the storage tape symbol, and send these informations to the cell C^1. By these informations, C^1 determines the next movement of T', and sends back the informations of the move directions of the heads

and the storage tape symbol to be rewritten to these two cells. Thus M' can simulate T' step by step, and can accept the same language as T'. (Q.E.D.)

Theorem 3.6 $\mathcal{L}$[MPSA] = $\mathcal{L}$[2TM(n + log m)].

Proof. First, we prove $\mathcal{L}$[MPSA] $\subseteq$ $\mathcal{L}$[2TM(n + log m)]. For any M $\in$ MPSA, T $\in$ 2TM(n + log m) which simulates M is constructed in the following way. Suppose that M has k scanners. Using n + [log m] squares of the storage tape, T simulates the k scanner of M. The storage tape of T is divided into k tracks. T records the internal states of n cells of the i-th scanner S_i using the first n squares of the i-th track of the stoarge tape. It also records the position of S_i with a binary number using the remaining [log m] squares. The finite-state control of M is simulated by that of T. For each S_i, T brings the input head to the row where S_i is scanning, and then, reading the input symbol of this row one by one, T rewrites the state of each cell of S_i on the storage tape. Then T simulates the finite-state control of M and rewrites the binary numbers which represent the positions of the scanners. By above, T can simulate M step by step, and it accepts the input if and only if M accepts it.

Next, $\mathcal{L}$[MPSA] $\supseteq$ $\mathcal{L}$[2TM(n + log m)] is shown. For any T' $\in$ 2TM(n + log m), M' $\in$ MPSA which simulates T' is constructed as follows. Suppose that the storage tape of T' consists of k binary tracks. M' uses (k+3) scanners ($S_1, \cdots, S_{k+3}$) to

simulate the input head, storage tape head and the storage tape of T', and also uses some additional scanners for working. The scanner S_1 is used to simulate the first n squares of the storage tape and the storage tape head. Each cell of S_1 memorizes the tape symbol of each square, and the cell that corresponds to the position of the storage tape head becomes a special state. The scanners $S_2,\cdots,S_{k+1}$ simulate the remaining [log m] squares of the storage tape, and the scanner S_{k+2} is used to keep the position of the storage tape head when it is reading one of these [log m] squares. It can be done in a similar way as in Theorem 3.3 using some additional working scanners. Namely, each of $S_2,\cdots,S_{k+1}$ remembers the binary number contained in each binary track of the [log m] squares, by its vertical position. The scanner S_{k+3} is used to simulate the input head of T'. If the input head is at the (i,j)-square, then S_{k+3} is placed at the i-th row, and the j-th cell of S_{k+3} becomes a special state. By above, M' can simulate T' step by step, and accepts the same language as T'. (Q.E.D.)

The following theorem shows that both OMPSA and O2PSA have tape complexity m+n. Thus the language acceptability of OMPSA(k,ℓ) is not enlarged even if the numbers k and ℓ are increased.

Theorem 3.7 $\mathcal{L}$[OMPSA] = $\mathcal{L}$[O2PSA] = $\mathcal{L}$[2TM(m+n)].

Proof. First we will show $\mathcal{L}[2TM(m+n)] \subseteq \mathcal{L}[O2PSA]$. Let T be an arbitrary 2TM(m+n). We construct $M \in O2PSA$ which simulates T. Let S_V and S_H be vertical and horizontal scanners of M. M must memorize the symbols written in the m+n squares of the storage tape and the position of the storage tape head of T. They can be archieved using the m+n cells of S_V and S_H. And the position of the input head of T is simulated by S_V. (Thus, M need not move S_H at all.) By above, M can simulate T step by step, and can accept the same language as T.

Next, we will show $\mathcal{L}[OMPSA] \subseteq \mathcal{L}[2TM(m+n)]$. In a similar manner as in Theorem 3.6, we can derive $\mathcal{L}[OMPSA] \subseteq \mathcal{L}[2TM(n + \log m + m + \log n)]$. Note that $(n + \log m + m + \log n) < 2(m+n)$ holds for every $m \geq 1$ and $n \geq 1$. Thus, from the tape reduction theorem (Theorem 2.12), $\mathcal{L}[OMPSA] \subseteq \mathcal{L}[2TM(m+n)]$ is obtained.

Now, $\mathcal{L}[O2PSA] \subseteq \mathcal{L}[OMPSA]$ is obvious, so that $\mathcal{L}[O2PSA] = \mathcal{L}[OMPSA] = \mathcal{L}[2TM(m+n)]$ is concluded. (Q.E.D.)

We have seen that several classes of two-dimensional tape automata are characterized by the notion of tape complexity. Thus, applying the hierarchy theorem of 2TM to them, we can systematically derive the relations of language acceptabilities among these classes of automata.

Theorem 3.8

$\mathcal{L}[2FSA] \subsetneq \mathcal{L}[PSA] \subsetneq \mathcal{L}[MPSA] \subsetneq \mathcal{L}[OMPSA] = \mathcal{L}[O2PSA] \subsetneq \mathcal{L}[2LBA]$,

$\mathcal{L}[2FSA] \subsetneq \mathcal{L}[2MHA] = \mathcal{L}[2MMA] \subsetneq \mathcal{L}[MPSA]$,

$\mathcal{L}[\mathrm{PSA}] \not\subseteq \mathcal{L}[\mathrm{2MHA}] = \mathcal{L}[\mathrm{2MMA}]$.

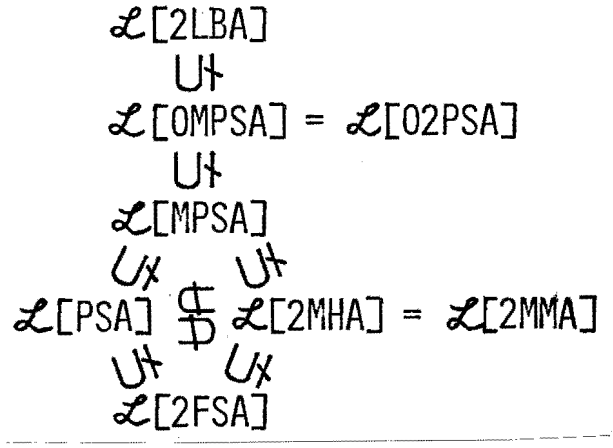

Fig. 3.6 The relations of language acceptabilities of several automata with rectangular input tapes.

Proof. We prove only the cases of (1) $\mathcal{L}[\mathrm{MPSA}] \subsetneq \mathcal{L}[\mathrm{O2PSA}]$ and (2) $\mathcal{L}[\mathrm{PSA}] \not\subseteq \mathcal{L}[\mathrm{2MHA}]$. The other relations can be proved likewise.

(1) From Theorem 3.6 and 3.7, $\mathcal{L}[\mathrm{MPSA}] = \mathcal{L}[\mathrm{2TM}(n + \log m)]$ and $\mathcal{L}[\mathrm{O2PSA}] = \mathcal{L}[\mathrm{2TM}(m+n)]$. It is easily seen that the tape functions $(n + \log m)$ and $(m+n)$ are both constructible. Let $\{(m_i, n_i)\}$ be a regular sequence such that $m_i = i$, $n_i = 1$. Then

$$\lim_{i\to\infty} \frac{(n_i + \log m_i)+1}{m_i + n_i} = 0.$$

Furthermore, $(n + \log m) < (m+n)$ holds for every $m \geq 1$, $n \geq 1$. Thus, from Theorem 2.22, $\mathcal{L}[\mathrm{2TM}(n + \log m)] \subsetneq \mathcal{L}[\mathrm{2TM}(m+n)]$

holds, and this leads $\mathcal{L}$[MPSA] $\subsetneq$ $\mathcal{L}$[O2PSA].

(2) From Theorem 3.3 and 3.5, $\mathcal{L}$[PSA] = $\mathcal{L}$[2TM(n)] and $\mathcal{L}$[2MHA] = $\mathcal{L}$[2TM(log mn)]. Apparently, the tape functions n and log mn are both constructible. Now, let $\{(m_i,n_i)\}$ be a regular sequence such that $m_i=i$, $n_i=1$. Then,

$$\lim_{i\to\infty} \frac{n_i+1}{\log m_i n_i} = 0.$$

On the other hand, let $\{(m'_i,n'_i)\}$ be another regular sequence such that $m'_i=1$, $n'_i=i$, then

$$\lim_{i\to\infty} \frac{\log m'_i n'_i + 1}{n'_i} = 0.$$

Thus, from Theorem 2.22, $\mathcal{L}$[PSA] $\not\supseteq$ $\mathcal{L}$[2MHA] is concluded.

(Q.E.D.)

3.3 Language Acceptabilities on Restricted Inputs

In the previous section, we investigated the language acceptabilities of several two-dimensional tape automata with rectangular inputs whose shapes are not restricted to special ones. In this section, we consider the case that the shapes of inputs are restricted. It will be seen that the automata with non-isotropic computational powers (such as PSA or MPSA) vary their language acceptabilities depending on the shapes of inputs. These properties can also be derived systematically by considering the tape complexities of such automata and the shape function.

First, we show the following lemma.

Lemma 3.9 Let L(m,n) be any tape function of 2TM, and let $f : \mathbf{N} \to \mathbf{N}^2$ and $g : \mathbf{N} \to \mathbf{N}^2$ be any shape functions such that $\Sigma_f^{2+} \subseteq \Sigma_g^{2+}$. Then

$$\mathcal{L}[2TM^f(L(f(j)))] = \mathcal{L}^f[2TM^g(L(g(j)))],$$

where $\mathcal{L}^f[2TM^g(L(g(j)))] = \{S \mid S=(S' \cap \Sigma_f^{2+}),\ S' \in \mathcal{L}[2TM^g(L(g(j)))]\}$.

Proof. For any given $T_f \in 2TM^f(L(f(j)))$, which accepts a language S, we consider $T_g \in 2TM^g(L(g(j)))$. T_g is precisely the same as T_f except that its inputs are extended to Σ_g^{2+}. Thus, if S' is the language accepted by T_g, then apparently $S=(S' \cap \Sigma_f^{2+})$ holds. So $\mathcal{L}[2TM^f(L(f(j)))] \subseteq \mathcal{L}^f[2TM^g(L(g(j)))]$ is concluded.

Conversely, for any given $T_g \in 2TM^g(L(g(j)))$, which accepts a language S', we consider $T_f \in 2TM^f(L(f(j)))$. T_f is also precisely the same as T_g except that its inputs are restricted to Σ_f^{2+}. Suppose that T_f accepts the languge S, then apparently $S=(S' \cap \Sigma_f^{2+})$. Thus $\mathcal{L}[2TM^f(L(f(j)))] \supseteq \mathcal{L}^f[2TM^g(L(g(j)))]$.

(Q.E.D.)

Now, let $\mathcal{A}$ be some class of two-dimensional tape automata, and let $f : \mathbf{N} \to \mathbf{N}^2$ be a shape function. The class of automata $\mathcal{A}$ whose inputs are restricted to Σ_f^{2+} is denoted by $\mathcal{A}^f$ (e.g. $2FSA^f$, PSA^f, etc.). Then the similar lemma for $\mathcal{A}$ can be derived in the same way as in Lemma 3.9.

Lemma 3.10 Let $\mathcal{A}$ be any class of two-dimensional tape automata, and let $f : \mathbf{N} \to \mathbf{N}^2$ and $g : \mathbf{N} \to \mathbf{N}^2$ be any shape functions such that $\Sigma_f^{2+} \subseteq \Sigma_g^{2+}$. Then,

$$\mathcal{L}[\mathcal{A}^f] = \mathcal{L}^f[\mathcal{A}^g],$$

where $\mathcal{L}^f[\mathcal{A}^g] = \{S \mid S=(S' \cap \Sigma_f^{2+}),\ S' \in \mathcal{L}[\mathcal{A}^g]\}$.

The next theorem means that if some class of automata $\mathcal{A}$ is equivalent (in languge acceptability) to 2TM(L(m,n)) on Σ^{2+}, then $\mathcal{A}$ is also equivalent to 2TM(L(m,n)) on the restricted inputs.

Theorem 3.11 Let $\mathcal{A}$ be a class of two-dimensional tape automata. If there exists some tape function L(m,n) of 2TM such that

$$\mathcal{L}[\mathcal{A}] = \mathcal{L}[2TM(L(m,n))],$$

then for any shape function $f : \mathbf{N} \to \mathbf{N}^2$,

$$\mathcal{L}[\mathcal{A}^f] = \mathcal{L}[2TM^f(L(f(j)))].$$

Proof. From Lemma 3.9 and 3.10, $\mathcal{L}[\mathcal{A}^f] = \mathcal{L}^f[\mathcal{A}] = \mathcal{L}^f[2TM(L(m,n))] = \mathcal{L}[2TM^f(L(f(j)))]$. (Here, the shape function g is regarded as some bijection.) (Q.E.D.)

Now, as a special case, we consider the shape function s(j)=(j,j), i.e. the case of square-shaped inputs.

Theorem 3.12

$$\mathcal{L}[2FSA^s] = \mathcal{L}[2TM^s(c)]$$

$$\mathcal{L}[2MHA^S] = \mathcal{L}[2TM^S(\log m)]$$

$$\mathcal{L}[2MMA^S] = \mathcal{L}[2TM^S(\log m)]$$

$$\mathcal{L}[2LBA^S] = \mathcal{L}[2TM^S(m^2)]$$

$$\mathcal{L}[PSA^S] = \mathcal{L}[2TM^S(m)]$$

$$\mathcal{L}[MPSA^S] = \mathcal{L}[2TM^S(m)]$$

$$\mathcal{L}[OMPSA^S] = \mathcal{L}[2TM^S(m)]$$

$$\mathcal{L}[O2PSA^S] = \mathcal{L}[2TM^S(m)]$$

Proof. We only prove the relation $\mathcal{L}[MPSA^S] = \mathcal{L}[2TM^S(m)]$. The other relations can also be derived in a similar way.

From Theorem 3.6 and 3.11, $\mathcal{L}[MPSA^S] = \mathcal{L}[2TM^S(m + \log m)] \subseteq \mathcal{L}[2TM^S(2m)]$. And from the tape reduction theorem (Corollary 2.13), $\mathcal{L}[MPSA^S] = \mathcal{L}[2TM^S(m)]$ is concluded. (Q.E.D.)

From Corollary 2.23 and Theorem 3.12, the relations of language acceptabilities among these automata on Σ_s^{2+} are easily derived.

Theorem 3.13 $\mathcal{L}[2FSA^S] \subsetneq \mathcal{L}[2MHA^S] = \mathcal{L}[2MMA^S] \subsetneq \mathcal{L}[PSA^S] = \mathcal{L}[MPSA^S] = \mathcal{L}[OMPSA^S] = \mathcal{L}[O2PSA^S] \subsetneq \mathcal{L}[2LBA^S]$.

Next, we consider the shape function $t(j)=(j,[\log j])$. The following theorem shows the relations of language acceptabilities among these automata on Σ_t^{2+}.

Theorem 3.13

$$\mathcal{L}[2FSA^t] = \mathcal{L}[2TM^t(c)]$$

$$\mathcal{L}[2MHA^t] = \mathcal{L}[2TM^t(\log m)]$$

$$\mathcal{L}[2MMA^t] = \mathcal{L}[2TM^t(\log m)]$$

$$\mathcal{L}[2LBA^t] = \mathcal{L}[2TM^t(m \log m)]$$

$$\mathcal{L}[PSA^t] = \mathcal{L}[2TM^t(\log m)]$$

$$\mathcal{L}[MPSA^t] = \mathcal{L}[2TM^t(\log m)]$$

$$\mathcal{L}[OMPSA^t] = \mathcal{L}[2TM^t(m)]$$

$$\mathcal{L}[O2PSA^t] = \mathcal{L}[2TM^t(m)]$$

Theorem 3.15 $\mathcal{L}[2FSA^t] \subsetneq \mathcal{L}[2MHA^t] = \mathcal{L}[2MMA^t] = \mathcal{L}[PSA^t] = \mathcal{L}[MPSA^t] \subsetneq \mathcal{L}[OMPSA^t] = \mathcal{L}[O2PSA^t] \subsetneq \mathcal{L}[2LBA^t]$.

$$\begin{array}{c} \mathcal{L}[2LBA^s] \\ \cup \\ \mathcal{L}[PSA^s] = \mathcal{L}[MPSA^s] = \mathcal{L}[OMPSA^s] = \mathcal{L}[O2PSA^s] \\ \cup \\ \mathcal{L}[2MHA^s] = \mathcal{L}[2MMA^s] \\ \cup \\ \mathcal{L}[2FSA^s] \end{array}$$

$$\begin{array}{c} \mathcal{L}[2LBA^t] \\ \cup \\ \mathcal{L}[OMPSA^t] = \mathcal{L}[O2PSA^t] \\ \cup \\ \mathcal{L}[2MHA^t] = \mathcal{L}[2MMA^t] = \mathcal{L}[PSA^t] = \mathcal{L}[MPSA^t] \\ \cup \\ \mathcal{L}[2FSA^t] \end{array}$$

Fig. 3.7 The relations of language acceptabilities of several automata on Σ_s^{2+} and Σ_t^{2+}.

For the other shape functions, the relation among these automata can also be derived in the same way as above.

Finally, we consider the relation of language acceptabilities for given two shape functions.

Theorem 3.16 Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be some classes of two-dimensional tape automata ($\mathcal{A}_1$ and $\mathcal{A}_2$ may be 2TM(L(m,n))), and let f and g be shape functions that satisfy $\Sigma_f^{2+} \subseteq \Sigma_g^{2+}$. Suppose that there exists a language S such that $S \in \mathcal{L}[\mathcal{A}_1^f]$ but $S \notin \mathcal{L}[\mathcal{A}_2^f]$. Then there exists some language S' such that $S' \in \mathcal{L}[\mathcal{A}_1^g]$ but $S' \notin \mathcal{L}[\mathcal{A}_2^g]$.

Proof. From Lemma 3.9 and 3.10, $\mathcal{L}[\mathcal{A}_1^f] = \mathcal{L}^f[\mathcal{A}_1^g]$ and $\mathcal{L}[\mathcal{A}_2^f] = \mathcal{L}^f[\mathcal{A}_2^g]$, so $S \in \mathcal{L}^f[\mathcal{A}_1^g]$ and $S \notin \mathcal{L}^f[\mathcal{A}_2^g]$. Thus there exists some $S' \in \mathcal{L}[\mathcal{A}_1^g]$ that satisfies $S=(S' \cap \Sigma_f^{2+})$. Such S' cannot be the element of $\mathcal{L}[\mathcal{A}_2^g]$, because $S \notin \mathcal{L}^f[\mathcal{A}_2^g]$.

(Q.E.D.)

Theorem 3.17 Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be some classes of two-dimensional tape automata, and let f and g be shape functions that satisfy $\Sigma_f^{2+} \subseteq \Sigma_g^{2+}$. Suppose that

$$\mathcal{L}[\mathcal{A}_1^g] \subseteq \mathcal{L}[\mathcal{A}_2^g],$$

then

$$\mathcal{L}[\mathcal{A}_1^f] \subseteq \mathcal{L}[\mathcal{A}_2^f].$$

Proof. From Lemma 3.9 and 3.10, $\mathcal{L}[\mathcal{A}_1^f] = \mathcal{L}^f[\mathcal{A}_1^g] \subseteq \mathcal{L}^f[\mathcal{A}_2^g] = \mathcal{L}[\mathcal{A}_2^f]$.

(Q.E.D.)

3.4 Concluding Remarks

In this chapter, several two-dimensional tape automata were introduced and their language acceptabilities were investigated. It was shown that the acceptabilities all these automata can be measured by means of the tape complexity. Thus their relations were derived from the hierarchy theorem of 2TM(L(m,n)).

Since the tape function L(m,n) is a function of the horizontal and the vertical sidelengths of the input, the notion of the tape complexity is also applicable to the automata with nonisotropic computational powers (such as PSA or MPSA). And thus, it is easily seen how the acceptabilities of such automata vary depending on the shape of the input. For example, on the set of the (unrestricted) rectangular input tapes, O2PSA is strictly more powerful than MPSA, and MPSA is strictly more powerful than PSA (Theorem 3.8). But it was shown that O2PSA, MPSA and PSA are all equivalent on the set of square-shaped input tapes (Theorem 3.12). It is interesting that these results were obtained from the consideration of the tape complexity.

CHAPTER 4

AUTOMATA OF TAPE COMPLEXITY log n

So far we have investigated several classes of automata which accept two-dimensional languages. Here, we consider some other classes of automata which accept one-dimensional languages and have tape complexity log n.

The class of automata of tape complexity log n has various interesting aspects. First, we can give several concrete (or easily imaginable) models of automata of tape complexity log n (e.g. a multi-head automaton, a multi-marker automaton, an n-bounded counter automaton, a multi-dimensional rebound automaton, etc.). They can be considered as finite-state automata with some simple auxiliary memories. Thus their computations are intuitive and easy to be understood. Conversely, it seems very hard for us to imagine a language that cannot be accepted by these automata, without using mathematical technics such as coding or diagonalization.

Automata of tape complexity log n are also closely related to multi-dimensional tape automata. For, the tape complexity log n is a necessary and sufficient amount of memory to keep coordinates on the multi-dimensional hyper-cubic tape of

sidelength n. A multi-dimensional rebound automaton is proposed in order to investigate this relation. It is a finite-state automaton with a hyper-cubic input tape, and an input string is written along one of the edges of this tape. Though the rebound automaton is a sort of multi-dimensional tape automaton, it accepts a one-dimensional language.

Language acceptabilities of multi-head automata and multi-marker automata have been studied by many researchers [5,14,19,22,25,30,42,43,48,49], and various results have been obtained. One of the important results is that there exists an infinite subhierarchy of language acceptabilities among them, where the number of heads or markers is considered as a complexity measure.

In this chapter, an n-bounded counter automaton and a multi-dimensional rebound automaton is newly defined, and their language acceptabilities are investigated in the comparison with other automata of tape complexity log n. All these automata are assumed to be deterministic and their input heads moves in two ways. (The input head of the multi-dimensional rebound automaton moves in many ways.)

4.1 Definitions

In this section, we give definitions of a multi-track log n tape-bounded Turing machine, a multi-head automaton, a multi-marker automaton, a multi-counter automaton, an

n-bounded multi-counter automaton (these automata have one-dimensional input tapes), and a multi-dimensional rebound automaton.

4.1.1 Multi-Track log n Tape-Bounded One-Dimensional Turing Machine (lTM(log n, k))

The log n tape-bounded one-dimensional Turing machine is the same one discussed in section 1.2. But, in order to investigate the detailed relations to the other automata of tape complexity log n, we adopt the number of binary tracks of the storage tape as a measure of computational complexity.

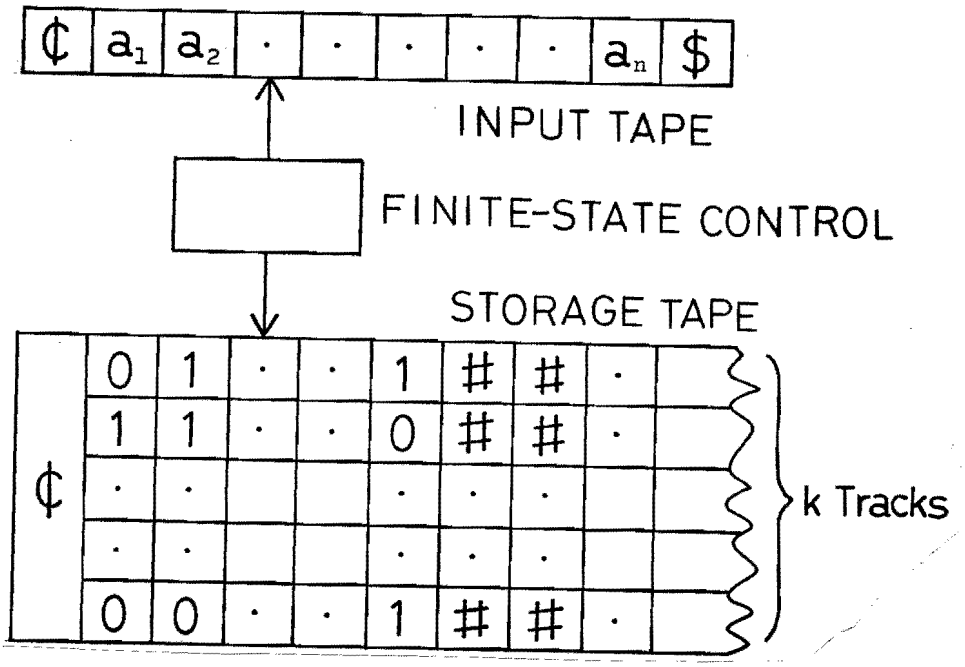

Fig. 4.1 A k-track log n tape-bounded one-dimensional Turing machine.

A *deterministic k-track log n tape-bounded one-dimensional Turing machine* (abbreviated to 1TM(log n, k)) consists of a finite-state control, a read-only input head, a storage tape head, a one-dimensional input tape, and a semi-infinite storage tape which is divided into k binary tracks (Fig.4.1). (Thus, it has 2^k storage tape symbols except the border symbol and the blank symbol.) It does not use more than [log n] squares of the storage tape for any input of length n. Note that $1TM(\log n) = \bigcup_{k=1}^{\infty} 1TM(\log n, k)$. The notion of the acceptance (of a word or a language) is the same as in 1TM(L(n)).

4.1.2 One-Dimensional Multi-Head Automaton (1MHA)

A *deterministic one-dimensional k-head automaton* (abbreviated to 1MHA(k)) consists of a one-dimensional input tape, k two-way read-only input heads, and a finite-state control. It is essentially the same as 2MHA(k) except that its input is one-dimensional. (Note that the input heads cannot sense each other.) $\bigcup_{k=1}^{\infty} 1MHA(k)$ is denoted by 1MHA.

4.1.3 One-Dimensional Multi-Marker Automaton (1MMA)

A *deterministic one-dimensional k-marker automaton* (abbreviated to 1MMA(k)) is also analogous to 2MMA(k), and it consists of a one-dimensional input tape, a two-way read-only input head, k markers, and a finite-state control. $\bigcup_{k=1}^{\infty} 1MMA(k)$ is denoted by 1MMA.

4.1.4 One-Dimensional Multi-Counter Automaton (1MCA)

A one-dimensional k-counter automaton consists of a one-dimensional input tape, a two-way read-only input head, k counters, and a finite-state control (Fig.4.2).

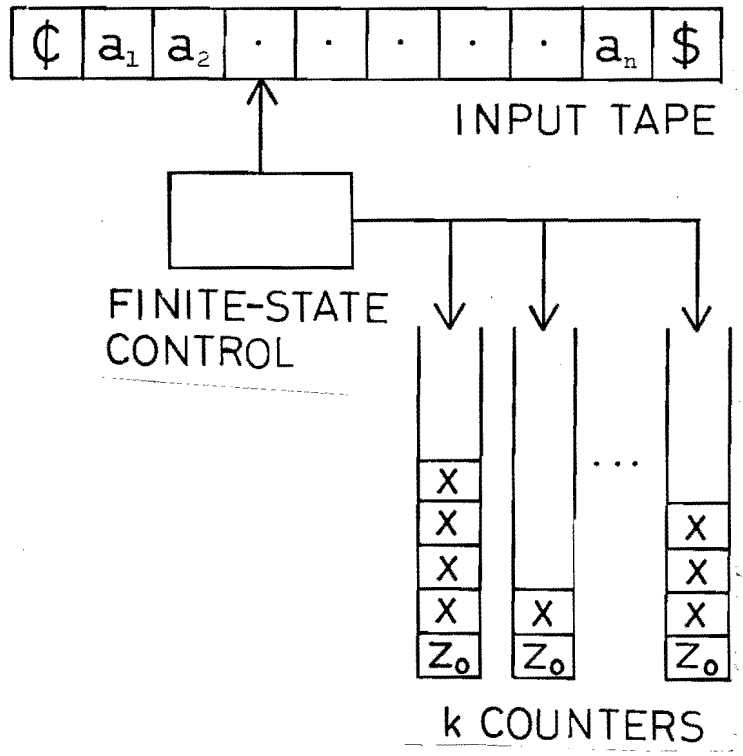

Fig. 4.2 A k-counter automaton.

Each of these counters can count any nonnegative integer The finite-state control can sense whether the content of each counter is zero or not. Thus the counter can be regarded as a pushdown memory with one pushdown symbol.

Formally a *deterministic one-dimensional k-counter automaton* is defined as a 9-tuple

$$M = (K, \Sigma, \Gamma, k, \delta, q_0, \{¢,\$\}, z_0, F),$$

where K is a nonempty finite set of internal states, Σ is a nonempty finite set of input symbols, Γ is a set of counter symbols ($\Gamma=\{x,z_0\}$), k is the number of counters, $q_0 \in K$ is an initial state, ¢ and $ are border symbols of the input tape ($\{¢,\$\} \cap \Sigma=\emptyset$), z_0 is a bottom symbol of the counter, and $F \subseteq K$ is a set of final states. δ is a mapping from a subset of $K\times(\Sigma \cup \{¢,\$\})\times\Gamma^k$ into $K\times\{L,R,H\}\times\{+1,0,-1\}^k$, where L, R, and H are shift directions of the input heads, and +1, 0, and -1 mean the alterations of the content of each counter. The class of deterministic k-counter automata is denoted by 1MCA(k), and $\bigcup_{k=1}^{\infty}$ 1MCA(k) is denoted by 1MCA.

Suppose that an input string $w \in \Sigma^+$ with border symbols (i.e., ¢w$) is given to M. M first sets the input head at the left side end, and makes all the counters empty. Then M begins its computation from the initial state q_0. We say that the word w is accepted by M, if M eventually halts in a final state.

4.1.5 One-Dimensional n-Bounded Multi-Counter Automaton (1BCA)

Next, an n-bounded multi-counter automaton is proposed. It resembles to 1MCA except that each counter can count a number up to n, where n is the length of the input. These counters are called n-bounded counters (abbreviated to nbc). The finite-state control can sense whether the content of each counter is 0 or n or not (Fig.4.3).

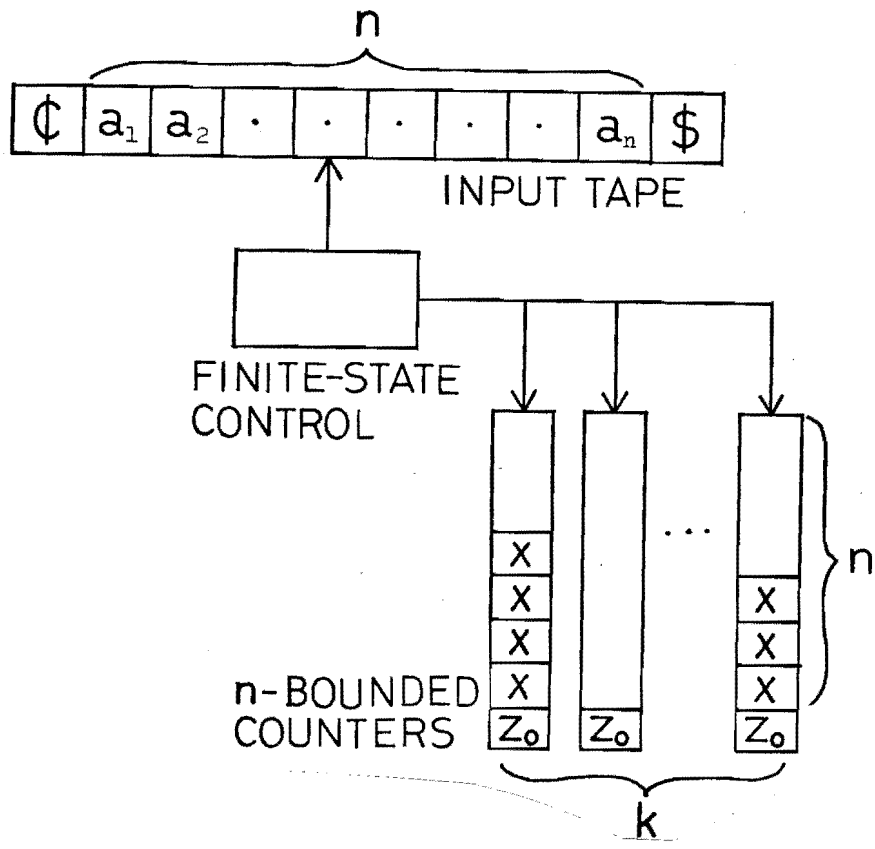

Fig. 4.3 An n-bounded k-counter automaton.

Formally a *deterministic one-dimensional n-bounded k-counter automaton* is a 9-tuple

$$M = (K, \Sigma, \Gamma, k, \delta, q_0, \{¢,\$\}, z_0, F),$$

where K, Σ, Γ, k, q_0, $\{¢,\$\}$, z_0, and F are the same as in 1MCA(k). δ is a mapping from a subset of $K\times(\Sigma \cup \{¢,\$\})\times(\Gamma \cup \{f\})^k$ into $K\times\{L,R,H\}\times\{+1,0,-1\}^k$, where f means that the counter is full (i.e. counting the number n). The class of deterministic n-bounded k-counter automata is denoted by 1BCA(k), and $1BCA = \bigcup_{k=1}^{\infty} 1BCA(k)$.

1BCA can be regarded as a restricted class of 1MHA. For, it is easily seen that 1BCA(k) is equivalent to 1MHA(k+1) such that its input heads, except only one head, cannot read the input symbols (i.e. they can sense only the border symbols).

4.1.6 Multi-Dimensional Rebound Automaton (RA)

Finally we give the definition of multi-dimensional rebound automaton. This is a kind of multi-dimensional tape automaton but accepts a one-dimensional language. A k-dimensional rebound automaton (abbreviated to RA(k)) consists of a k-dimensional hypercubic input tape, a read-only input head, and a finite-state control.

If a word $w = a_1 a_2 \cdots a_n$ on Σ ($a_i \in \Sigma$, $1 \leq i \leq n$, $n=1,2,\cdots$) is given, a k-dimensional input tape for RA(k) is defined as follows. Let us consider a k-dimensional hypercubic tape of sidelength n+2. It is divided into $(n+2)^k$ unit hypercubes. In each unit hypercube, an input symbol (Σ), a blank symbol (#), or a border symbol (B) is written. The coordinate of each unit hypercube is repesented by $(j_1, j_2, \cdots, j_k)$, where $j_\ell = 0,1,\cdots,n+1$ ($\ell = 1,2,\cdots,k$). Then the input tape is defined as a mapping $t : \{0,1,\cdots,n+1\}^k \to (\Sigma \cup \{\#, B\})$ which satisfy

$$t(1,1,\cdots,1,j) = a_j \quad (1 \leq j \leq n)$$

$$t(j_1, j_2, \cdots, j_k) = \# \quad (1 \leq j_k \leq n,\ 2 \leq j_\ell \leq n \text{ for some } \ell\ (1 \leq \ell \leq k-1))$$

$$t(j_1, j_2, \cdots, j_k) = B \quad (j_\ell = 0 \text{ or } n+1 \text{ for some } \ell\ (1 \leq \ell \leq k))$$

Fig.4.4 shows an input tape for RA(2) and its coordinates, and Fig.4.5 shows an input tape for RA(3) (border symbols are omitted in Fig.4.5).

The input head can move freely in this hypercubic tape, but never falls off. Note that RA(1) is identical to a (usual) two-way finite-state automaton with a one-dimensional input tape.

Formally a *deterministic k-dimensional rebound automaton* is defined as a 7-tuple

$$M = (K, \Sigma, k, \delta, q_0, \{\#,B\}, F),$$

where K is a nonempty finite set of internal states, Σ is a nonempty finite set of input symbols, k is the dimension of the input tape, $q_0 \in K$ is an initial state, # and B are a blank symbol and a border symbol, and $F \subseteq K$ is a set of final states. δ is a mapping from a subset of $K\times(\Sigma \cup \{\#,B\})$ into $K\times\{+1,0,-1\}^k$, where +1,0,-1 are shift directions of the input head along each axis of the input tape.

Now, let us give M a k-dimensional hypercubic input tape, in which a word $w \in \Sigma^+$ is written. Suppose M begins its computation from the initial state q_0 at the $(1,1,\cdots,1)$-square. If M halts in a final state, w is said to be accepted by M.

$\bigcup_{k=1}^{\infty}$ RA(k) is denoted by RA. RA can be also regarded as some restricted class of 1BCA. Because, we can easily see that RA(k) is equivalent to 1BCA(k-1) with the restriction that it can read the input symbol only when all the counters are empty.

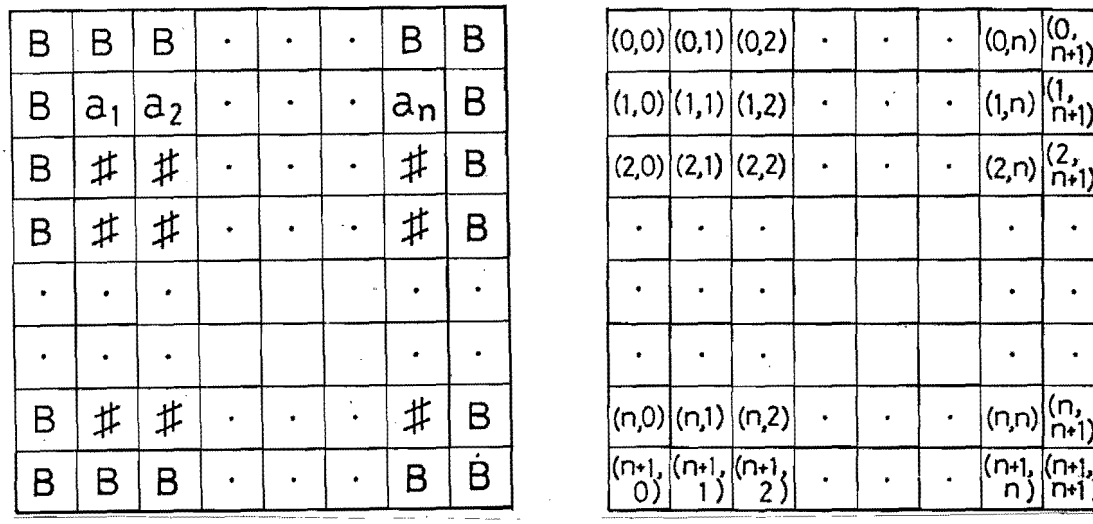

Fig. 4.4 An input tape for RA(2) and its coordinates.

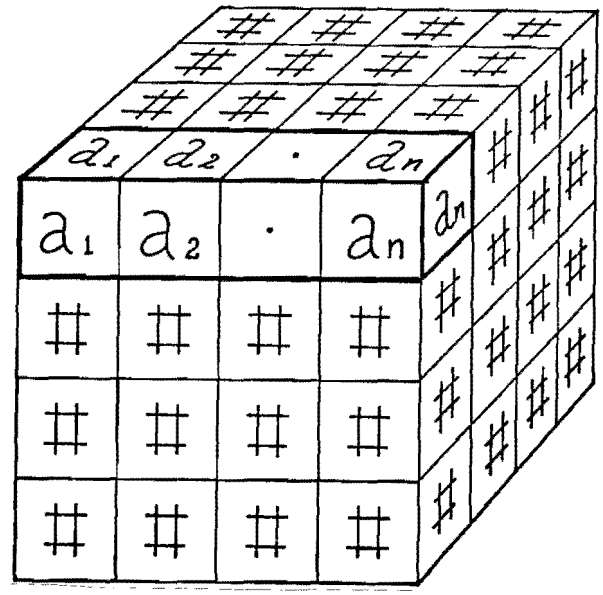

Fig. 4.5 An input tape for RA(3).

4.1.7 Some Notations

Several classes of automata of tape complexity log n have been defined above. Let $\mathcal{L}[\mathcal{A}]$ denote a class of one-dimensional languages accepted by $\mathcal{A}$, where $\mathcal{A}$ is some class of these automata (e.g. $\mathcal{L}$[1BCA], $\mathcal{L}$[RA(k)], etc.). (Note that all these classes of languages are one-dimensional ones, though RA has multi-dimensional input tapes.)

Let $\mathcal{A}^1$ denote a restricted class of $\mathcal{A}$ whose input symbol is only one (i.e. it has contentless inputs). And the class of languages accepted by $\mathcal{A}^1$ is denoted by $\mathcal{L}[\mathcal{A}^1]$.

Let $\mathcal{L}_0$, $\mathcal{L}_1$, $\mathcal{L}_2$, and $\mathcal{L}_3$ be the classes of the type 0 (recursively enumerable), the type 1 (context-sensitive), the type 2 (context-free), and the type 3 (regular) languages, respectively. And let $\mathcal{L}_0^1$, $\mathcal{L}_1^1$, $\mathcal{L}_2^1$, and $\mathcal{L}_3^1$ denote these classes of languages on one input symbol. (Note that $\mathcal{L}_2^1 = \mathcal{L}_3^1$. (Parikh [39]))

4.2 Relations of Language Acceptabilities

In this section, we investigate the relations of language acceptabilities among these automata.

First, the relations among 1MHA(k), 1MMA(k), and 1BCA(k) are considered.

Theorem 4.1

$\mathcal{L}$[1MHA(k)] $\subseteq$ $\mathcal{L}$[1MMA(k)] $\subseteq$ $\mathcal{L}$[1BCA(k)] $\subseteq$ $\mathcal{L}$[1MHA(k)].
(k=1,2,$\cdots$)

Proof. 1MMA(k) can easily simulate 1MHA(k) by placing k markers at the positions of k heads. Thus $\mathcal{L}$[1MHA(k)] $\subseteq$ $\mathcal{L}$[1MMA(k)].

$\mathcal{L}$[1MMA(k)] $\subseteq$ $\mathcal{L}$[1BCA(k)] can be derived in a similar way as in Theorem 3.2. Because, 1BCA(k) can simulate 1MMA(k) by remembering the relative position of each marker to the input head with each nbc.

$\mathcal{L}$[1BCA(k)] $\subseteq$ $\mathcal{L}$[1MHA(k+1)] is also obvious, because k nbc's of 1BCA(k) can be easily simulated by k heads of 1MHA(k+1). (Q.E.D.)

From this, we can see that 1MHA(k), 1MMA(k), and 1BCA(k) form a total order with respect to their language acceptabilities. Namely,

$\mathcal{L}$[1MHA(1)] $\subseteq$ $\mathcal{L}$[1MMA(1)] $\subseteq$ $\mathcal{L}$[1BCA(1)] $\subseteq$ $\mathcal{L}$[1MHA(2)] $\subseteq$ $\mathcal{L}$[1MMA(2)] $\subseteq$ $\mathcal{L}$[1BCA(2)] $\subseteq$ $\mathcal{L}$[1MHA(3)] $\subseteq$ $\cdots\cdots$.

In the case of one input symbol, the next theorem holds.

Theorem 4.2

$\mathcal{L}$[1MHA^1(k)] $\subseteq$ $\mathcal{L}$[1MMA^1(k)] $\subseteq$ $\mathcal{L}$[1BCA^1(k)] = $\mathcal{L}$[1MHA^1(k)].
(k=1,2,$\cdots$)

Proof. $\mathcal{L}$[1MHA^1(k)] $\subseteq$ $\mathcal{L}$[1MMA^1(k)] $\subseteq$ $\mathcal{L}$[1BCA^1(k)] $\subseteq$ $\mathcal{L}$[1MHA^1(k+1)] can be derived in the same way as in Theorem

4.1. And $\mathcal{L}[1BCA^1(k)] \supseteq \mathcal{L}[1MHA^1(k+1)]$ is also proved from the fact that each head of $1MHA^1(k+1)$ on the contentless input can be substituted by each nbc or the head of $1BCA^1(k)$.

(Q.E.D.)

Next, we consider the relation between 1BCA(k) and RA(k).

Theorem 4.3 $\mathcal{L}[RA(k+1)] \subseteq \mathcal{L}[1BCA(k)]$.

Proof. It is obvious from the fact that RA(k+1) is equivalent to some restricted class of 1BCA(k). (Q.E.D.)

Theorem 4.4 $\mathcal{L}[RA^1(k+1)] = \mathcal{L}[1BCA^1(k)]$.

Proof. $\mathcal{L}[RA^1(k+1)] \subseteq \mathcal{L}[1BCA^1(k)]$ is easily seen as in Theorem 4.3. Conversely, the input head and k nbc's of $1BCA^1(k)$ are simulated by the position of the input head on the (k+1)-dimensional contentless hypercubic tape of $RA^1(k)$, because the input is contentless. Thus $\mathcal{L}[RA^1(k+1)] \supseteq \mathcal{L}[1BCA^1(k)]$ also holds. (Q.E.D.)

Next, the relation between 1BCA(k) and 1TM(log n, k) is investigated. In the two-dimensional case, the equivalence of 2MHA, 2MMA, and 2TM(log m) has already been shown in Theorem 3.3. In the two-dimensional case, too, the equivalence of 1MHA, 1MMA, 1BCA, and 1TM(log n) will be shown. But we now derive the very precise relation between 1BCA and 1TM(log n),

in the following two theorems.

Theorem 4.5 $\mathcal{L}[1BCA(k)] \subseteq \mathcal{L}[1TM(\log n, k+2)]$. $(k=1,2,\cdots)$

Proof. For any given $M \in 1BCA(k)$, we can construct $T \in 1TM(\log n, k+2)$ which simulates M as follows.

If an input $w \in \Sigma^+$ of length n is given, T first writes the number n in the (k+1)-th track of the storage tape in binary, where the leftmost square contains the l.s.b. (least significant bit) of n. At the same time, T marks the [log n]-th square of the (k+2)-th track. These can be done by counting the length of w in binary, and simultaneously marking in the (k+2)-th track of the square where the most significant 1 in the (k+1)-th track is written.

Then T begins to simulate M step by step. The finite-state control and the input head of M are simulated by those of T. Each nbc of M is simulated by each track of the storage tape of T. Namely, the content of the i-th nbc $(i=1,2,\cdots,k)$ is recorded in the i-th track of the storage tape in binary so that the leftmost square contains the l.s.b. In order to check whether the content of the i-th counter is 0 or not, T examines whether all the [log n] squares of the i-th track contain 0's. Similarly, to check whether the content of the i-th counter is n, T examines whether the contents of the i-th track and the (k+1)-th track is the same. T examines only [log n] squares, in this case, too. Since T can know

the [log n]-th square by the contents of the (k+2)-th track, these procedures always terminate. If M increases (or decreases) the content of the i-th counter, T also increases (or decreases) the binary number contained in the i-th track.

By above, T can simulate the movements of M step by step, and can accept the same language as M. (Q.E.D.)

Theorem 4.6 $\mathcal{L}[1TM(\log n, k)] \subseteq \mathcal{L}[1BCA(k+3)]$. $(k=1,2,\cdots)$

Proof. For any given $T \in 1TM(\log n, k)$, we can construct $M \in 1BCA(k+3)$ which simulates T, in the following way.

Let c_i be the i-th nbc of M, and let $|c_i|$ denote the number kept by c_i $(i=1,2,\cdots,k+3)$. k nbc's c_1, c_2, $\cdots$, c_k are used to memorize the contents of k tracks of the storage tape of T, c_{k+1} remembers the position of the storage tape head of T (i.e. the distance from the symbol ¢), and c_{k+2} and c_{k+3} are used for working.

Now, let $a_{i,1}a_{i,2}\cdots a_{i,m}$ be the contents of the [log n] squares of the i-th track of T $(i=1,2,\cdots,k,\ m=[\log n],\ a_{i,j}=$ 0 or 1 $(j=1,2,\cdots,m))$, and suppose that the storage tape head of T is reading the h-th square. Then c_i memorizes the content of the i-th track as the following integer.

$$|c_i| = \sum_{j=0}^{h-2} a_{i,h-1-j} \cdot 2^j + \sum_{j=h-1}^{m-2} a_{i,m+h-1-j} \cdot 2^j + 2^{m-1}$$

Namely, c_i keeps the m-digit binary number

$$1\ a_{i,h+1}\ a_{i,h+2}\ \cdots\ a_{i,m}\ a_{i,1}\ a_{i,2}\ \cdots\ a_{i,h-1}\ ,$$

where the rightmost bit (i.e. $a_{i,h-1}$) is the l.s.b., and the leftmost 1 (i.e. 2^{m-1}) is added so that $|c_i|$ always becomes an m-digit number. The h-th digit $a_{i,h}$ is remembered in the finite-state control of M.

By the way, M can easily do the following operations with a help of c_{k+2}.

(i) Divide $|c_i|$ by 2 and get the remainder (l.s.b.).

(ii) Multiply $|c_i|$ by 2 and add a constant a'_h (=0 or 1).

Furthermore, M can do the following operation using c_{k+2} and c_{k+3}.

(iii) Transpose the binary number $|c_i|$ except the m.s.b. (most significant bit). (Namely, if $|c_i| = \sum_{j=0}^{m-2} b_j \cdot 2^j + 2^{m-1}$ for some $b_0, b_1, \cdots, b_{m-2}$, then this operation makes $|c_i| = \sum_{j=0}^{m-2} b_j \cdot 2^{m-2-j} + 2^{m-1}$.)

To perform the operation (iii), M first sets $|c_{k+2}|=0$ and $|c_{k+3}|=1$, and then repeats the following procedure until $|c_i|$ becomes 1.

(1) Apply the operation (i) to c_i and get the l.s.b. of $|c_i|$.

(2) Apply the operation (ii) to c_{k+3}, and add the l.s.b. gotten just now.

Consequently, c_{k+3} contains the transposition of initial $|c_i|$. Thus the operation (iii) is completed by transferring $|c_{k+3}|$ into c_i.

Now, M simulates T in the following way. M first sets $|c_i|=2^{m-1}$ $(i=1,2,\cdots,k)$ with a help of c_{k+2} and c_{k+3}. And then M begins to simulate T step by step. The finite-state control and the input head of T is simulated by those of M.

As for the storage tape of T, M simulate it track by track. If T rewrites $a_{i,h}$ into $a'_{i,h}$ and moves the storage tape head to the left, then M executes the following procedure.

(1) Get $a_{i,h-1}$ by applying (i) to c_i, and remember it in the finite-state control.

(2) Transpose $|c_i|$ by the operation (iii).

(3) Apply (ii) to c_i and add $a'_{i,h}$ to it.

(3) Transpose $|c_i|$ by the operation (iii).

If T rewrites $a_{i,h}$ into $a'_{i,h}$ and moves the storage tape head to the right, then M executes the following procedure.

(1) Transpose $|c_i|$ by the operation (iii).

(2) Get $a_{i,h+1}$ by applying (i) to c_i, and remember it in the finite-state control.

(3) Transpose $|c_i|$ by the operation (iii).

(4) Apply (ii) to c_i and add $a'_{i,h}$ to it.

M performs the above procedure for each track. Then M increments or decrements $|c_{k+1}|$ according to the shift direction of the storage tape head of T. If $|c_{k+1}|=0$, then M knows that the storage tape head of T is reading the border symbol ¢.

By above, M can accept the same language as T, and this completes the proof. (Q.E.D.)

Clearly, Theorem 4.5 and 4.6 hold even if the number of input symbols is restricted to one.

Corollary 4.7

$\mathcal{L}[1\mathrm{BCA}^1(k)] \subseteq \mathcal{L}[1\mathrm{TM}^1(\log n, k+2)]$,

$\mathcal{L}[1TM^1(\log n, k)] \subseteq \mathcal{L}[1BCA^1(k+3)]$.

The next corollary is easily obtained from Theorem 4.1-4.6 and Corollary 4.7.

Corollary 4.8

$\mathcal{L}[RA] \subseteq \mathcal{L}[1MHA] = \mathcal{L}[1MMA] = \mathcal{L}[1BCA] = \mathcal{L}[1TM(\log n)]$,

$\mathcal{L}[RA^1] = \mathcal{L}[1MHA^1] = \mathcal{L}[1MMA^1] = \mathcal{L}[1BCA^1] = \mathcal{L}[1TM^1(\log n)]$.

Next, we investigate the acceptability of 1MCA(k).

Theorem 4.9 If $k \geq 2$, then $\mathcal{L}[1MCA(k)] = \mathcal{L}_0$ and $\mathcal{L}[1MCA^1(k)] = \mathcal{L}_0^1$.

Proof. Minsky [29] showed that a two-counter automaton†, without an input tape, can simulate a one-dimensional one-tape Turing machine. Thus it is easily seen that 1MCA(2) can simulate any Turing machine with one input tape and one storage tape, where the storage tape is not bounded by any tape function. So 1MCA(k) ($k \geq 2$) is universal. (Q.E.D.)

Theorem 4.10 $\mathcal{L}[1MCA(1)] \subseteq \mathcal{L}[1BCA(1)]$.

Proof. Let A be an arbitrary 1MCA(1) with s internal states, and let n be the length of an input. We denote the

---

† See the footnote on page 49.

computational configuration of A by a triple (q, i, j), where q is an internal state, i is the position of the input head ($0 \leq i \leq n+1$), and j is the number kept by the counter ($j \geq 0$).

If A counts a number greater than s(n+2), then, for some q, i, j_1, and j_2 ($0 \leq i \leq n+1$, $0 < j_1 < j_2$), two computational configurations (q, i, j_1) and (q, i, j_2) must appear at some times t_1 and t_2 ($t_1 < t_2$) respectively, and the counter does not become empty between t_1 and t_2. So, at time $t_2 + k(t_2 - t_1)$ ($k = 1, 2, \cdots$), the computational configuration of A becomes $(q, i, j_2 + k(j_2 - j_1))$. Thus the number counted by A increases indefinitely. Accordingly, if A accepts the input, A never counts the number greater than s(n+2).

Using this fact, we can construct $B \in$ 1BCA(1) which accepts the same language as A. The nbc of B simulates the counter of A as follows. B has a finite counter which can count up to s in the finite-state control. If A increases (or decreases) the counter, then B increases (or decreases) this finite counter. B actually alters the nbc, only if the finite counter overflows or underflows. In other words, B memorizes the quotient of x/s in the nbc, and the remainder of it in the finite-state control, where x is the number counted by A. Thus B can count from 0 to s(n+1)+s (additional s is also counted by the finite-state control). By this B can simulate A step by step, and can accept the same language as A.

(Q.E.D.)

It is an interesting fact that $\mathcal{L}$[1MCA(1)] becomes some

restricted subclass of $\mathcal{L}$[1TM(log n)] despite of the universality of 1MCA(2).

4.3 Acceptability of RA(2)

Now, we especially investigate the acceptability of RA(2), which is one of the weakest subclass of automata of tape complexity log n (except the class of finite automata). Although RA(2) has no auxiliary memory, it can accept several non-regular languages. It depends on the fact that RA(2) can move freely on the input tape and can rebound at the edge of the input.

Theorem 4.11 There exists a language $L_1 \in \mathcal{L}$[RA(2)] such that $L_1 \in \mathcal{L}_2$ but $L_1 \notin \mathcal{L}_3$.

Proof. Let us consider the language $L_1=\{w\cdot w^R \mid w \in \{0,1\}^+\}$ (w^R denotes the transposed sequence of w). It is known that L_1 is context-free but not regular [18]. We can construct $M_1 \in$ RA(2) which accepts L_1. Suppose a word $w = a_1 a_2 \cdots a_{2n} \in \{0,1\}^+$ is given to M_1. In order to check if $a_i = a_{2n+1-i}$ for all i ($1 \leq i \leq 2n$), M_1 moves the input head as shown in Fig.4.6. From the square in which a_i is written, M_1 moves to the south-west direction at an angle of 45°. If M_1 reaches to the border, then it turns and moves to the east until it reads the border symbol. And then, moving to the north-west direction, M_1 can

reach the square where a_{2n+1-i} is written. M_1 does this for i=1,2,$\cdots$,2n, and accepts w if $a_i=a_{2n+1-i}$ for all i.

(Q.E.D.)

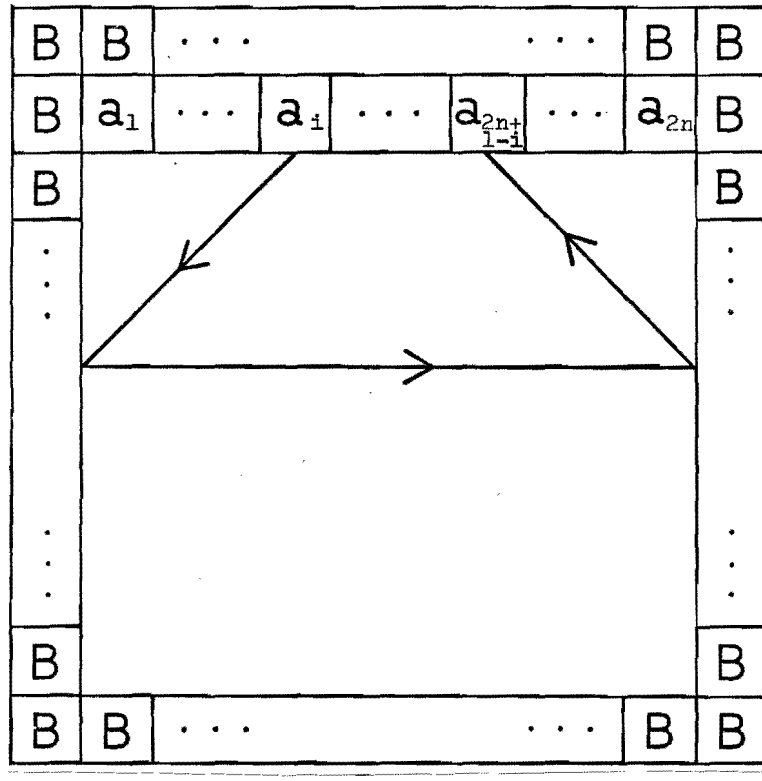

Fig. 4.6 Comparing the input symbols a_i and a_{2n+1-i}.

Theorem 4.12 There exists a language $L_2 \in \mathcal{L}[RA^1(2)]$ such that $L_2 \in \mathcal{L}_1^1$ but $L_2 \notin \mathcal{L}_2^1$.

Proof. We construct $M_2 \in RA^1(2)$ which accepts the language $L_2=\{a^{2^i} \mid i=1,2,\cdots\} \subset \{a\}^+$. L_2 is context-sensitive but not context-free [18]. Suppose an input tape of sidelength n is given to M_2 (it is a contentless input tape). M_2 can

find the $(2^i,1)$-square ($i=1,2,\cdots$) in a similar way as in Lemma 2.31. Suppose M_2 is at the $(2^k,1)$-square. Moving the input head to the north-east direction at an angle of 45°, M_2 can reach the $(1,2^k)$-square. M_2 then go to the south-west direction along the line of a slope 2. Thus M_2 can reach the $(2^{k+1},1)$-square (Fig.4.7)

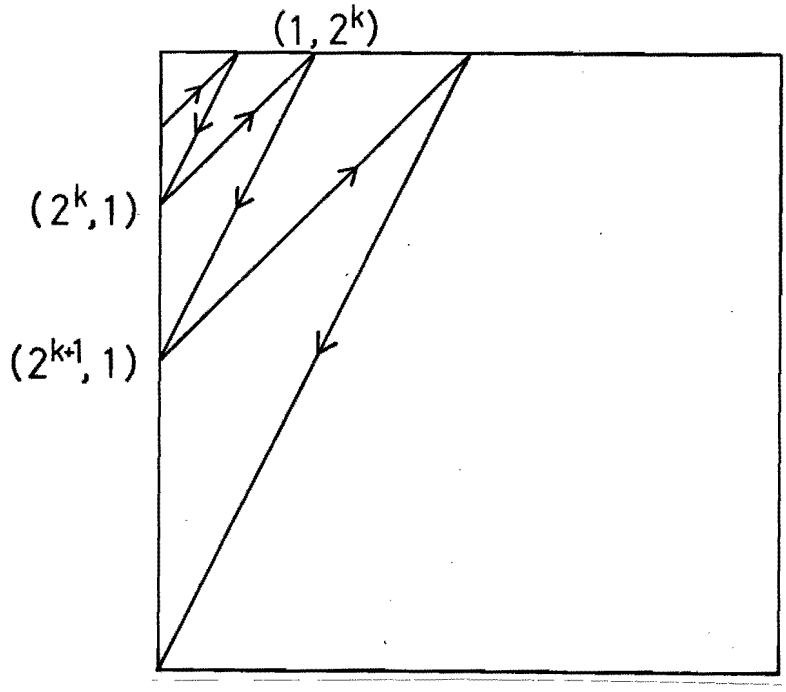

Fig. 4.7 The recognition process of $w \in L_2$ by M_2.

Starting from the (1,1)-square, M_2 repeats this procedure over again. By this, M_2 can check whether $m=2^j$ for some j. If it is so, then M_2 accepts the input. (Q.E.D.)

We can also show the other examples of languages accepted by RA(2), which are context-sensitive but not context-free.

$L_3 = \{w \cdot w | w \in \{0,1\}^+\}$

$L_4 = \{0^n 1^n 0^n | n=1,2,\cdots\}$

Fig.4.8 shows the method of comparing the input symbols a_i and a_{n+i} to accept the language L_3. And L_4 can also be accepted by finding the (1,[n/3])-square and the (1,[2n/3])-square.

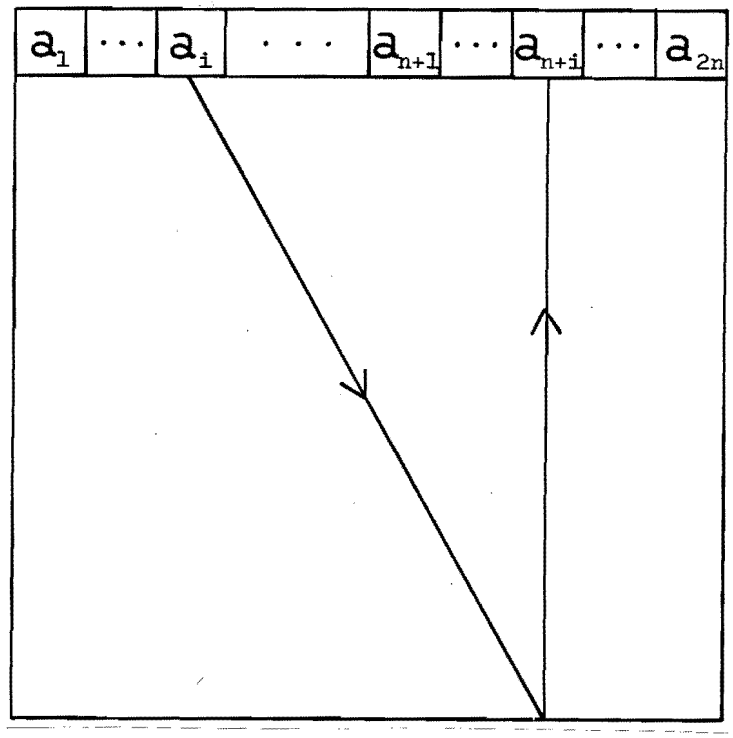

Fig. 4.8 Comparing the input symbols a_i and a_{n+i}. (From the symbol a_i, the automaton moves along the line of a slope -2.)

However, it seems that the following context-free language L_5 cannot be accepted by any RA(2). But its proof is remaining open.

$L_5 = \{w | w \in \{0,1\}^+$, and the number of the symbol 0's

contained in w is equal to the number of the symbol 1's.}
(But, it is easily seen that 1MCA(1) can accept L_5.)

Theorem 4.13

$\mathcal{L}[\mathrm{RA}(2)] \supsetneq \mathcal{L}_3$,

$\mathcal{L}[\mathrm{RA}^1(2)] \supsetneq \mathcal{L}_3^1$.

Proof. It is easy to see that $\mathcal{L}[\mathrm{RA}(2)] \supseteq \mathcal{L}[\mathrm{RA}(1)]$ and $\mathcal{L}[\mathrm{RA}^1(2)] \supseteq \mathcal{L}[\mathrm{RA}^1(1)]$. Thus, from Theorem 4.12 and the facts $\mathcal{L}[\mathrm{RA}(1)] = \mathcal{L}_3$ and $\mathcal{L}[\mathrm{RA}^1(1)] = \mathcal{L}_3^1$, $\mathcal{L}[\mathrm{RA}(2)] \supsetneq \mathcal{L}_3$ and $\mathcal{L}[\mathrm{RA}^1(2)] \supsetneq \mathcal{L}_3^1$ are concluded. (Q.E.D.)

Now, we consider a class of automata $\hat{\mathrm{RA}}(2)$ as a variant of RA(2). The shape of the input tape of $\hat{\mathrm{RA}}(2)$ is different

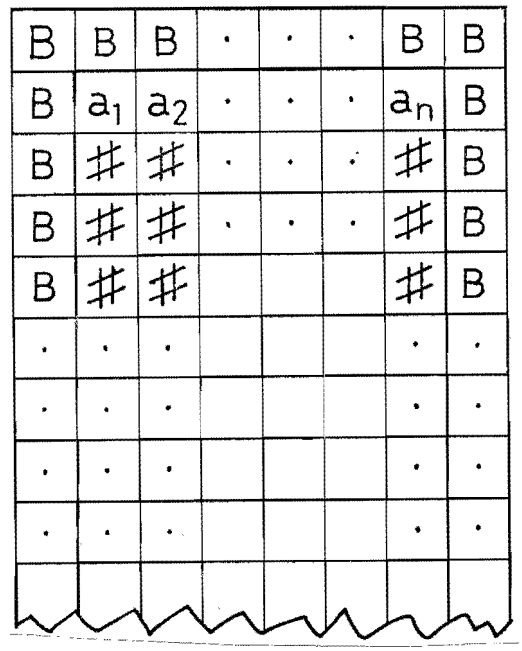

Fig. 4.9 The input tape for $\hat{\mathrm{RA}}(2)$.

from that of RA(2). The width of the input is finite, but it extends downward unlimitedly, and the border symbols are attached at the north, west, and east edges of it (Fig.4.8). A one-dimensional input word is written in its first row.

In order to show $\mathcal{L}[RA(2)] = \mathcal{L}[\hat{RA}(2)]$, we derive the following lemma.

Lemma 4.14 For any $M \in RA(2)$, there exists $M' \in RA(2)$ which accepts the same language as M and never reads the border symbol of the south edge of the input.

Proof. M' simulates M as follows.

Let s be the number of states of M. Scanning the first row of the input tape, M' first examines whether $n \leq s+1$ or not (n is the length of the input). If $n \leq s+1$, then M' memorizes n by the internal state, and simulates M moving the input head on the first row only. Namely, M' records the number of the row on which the input head of M is moving in the finite-state control of M'. Clearly, in this case, M' does not read the border symbol B at the south edge of the tape. Thus, in what follows, we only consider the case of $n>s+1$.

When M' simulates M, the internal state of M is simulated by that of M'. However, as for the position of the input head of M, M' remembers the position transformed by some coordinate transformation τ. Because M may read the border symbol at the south edge. Thus if the coordinate of the input head of M is (j,k), then M' places its head at $\tau(j,k)$.

But, τ is not a fixed transformation. Namely, τ is one of the following τ_i (i=0,1,2,3).

$\tau_0(j,k) = (j,k)$

$\tau_1(j,k) = (k,j)$

$\tau_2(j,k) = (n-k+1,n-j+1)$

$\tau_3(j,k) = (n-j+1,n-k+1)$

τ_0 is an identity transformation, τ_1 and τ_2 are symmetries (symmetrical transformations) with respect to the diagonals of an input tape, and τ_3 is a symmetry with respect to the center of an input tape (Fig.4.9). $\{\tau_i\}$ forms a group under the operation of composition (Table 4.1). M' chooses the transformation τ from $\{\tau_i\}$ depending on the conditions (stated later) at each time of the simulation, thus τ is

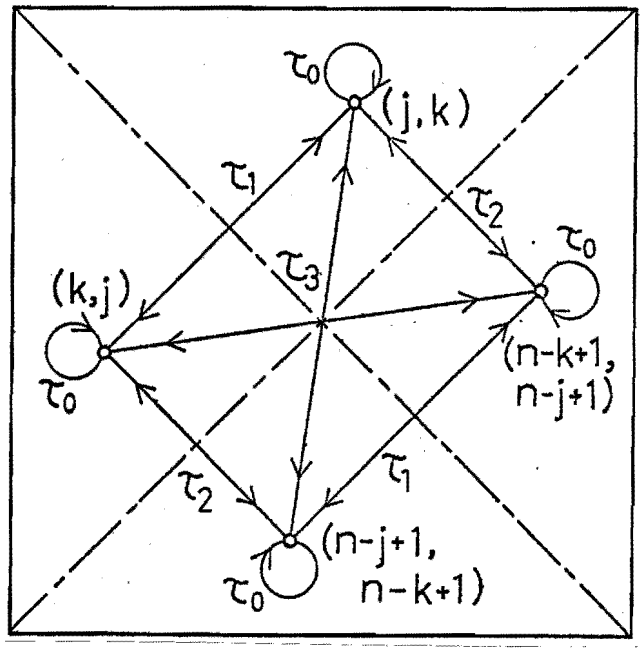

Fig. 4.10 Coordinate transformations by τ_i (i=0,1,2,3)

	τ_0	τ_1	τ_2	τ_3
τ_0	τ_0	τ_1	τ_2	τ_3
τ_1	τ_1	τ_0	τ_3	τ_2
τ_2	τ_2	τ_3	τ_0	τ_1
τ_3	τ_3	τ_2	τ_1	τ_0

Table 4.1 Composition of τ_i.

time variant. And M' remembers which transformation is being used now, in the finite-state control.

One series of M is a sequence of movements of M, since M has once read an element of $\Sigma \cup \{B\}$, until M next reads one of $\Sigma \cup \{B\}$ again. (M does not read $\Sigma \cup \{B\}$ during this period.) M' begins to simulate M from the (1,1)-square, setting $\tau=\tau_0$. One series of M is simulated by *one series of M'*, which is somewhat different from that of M. One series of M' is a sequence of movements since M' has once read the border symbol of the north edge or the element of Σ, until M' reads one of these in some time. More precisely, the movements in one series of M' are as follows. (Note that during one series of M', it may read the border symbol or the element of Σ several times.)

(1) First, M' simulates M up to (s+1) steps. In this process, M' records all the movements of M (state transitions, and shift directions of the input head) in the finite-state control. Since $s+1<n$, M' never reads the square of the n-th row, nor the border symbol of the south edge. When M' is simulating M step by step, the input head of M' moves so as to agree with the transformation τ. At every step of the simulation of M, M' checks the following conditions (a)-(d). If one of these conditions is satisfied, M' interrupts the simulation of M at that moment (even when M' has not finished the (s+1) steps yet), and performs the designated movements to each case.

(a) The case that M' is reading an element of Σ with $\tau=\tau_0$.

In this case, M is also reading this symbol. Thus the one series of M' is finished at this point.

(b) The case that M' is reading the (j,1)-square ($1 \leq j \leq n$) with $\tau=\tau_1$ (this can be examined by checking whether the left neighbouring square contains B, at each step).

In this case, M is reading the (1,j)-square. Thus, moving to the north-east direction at an angle of 45°, M' places the input head at the $\tau_1(j,1)$ = (1,j)-square. M' then simulates the movement of M, and finishes the one series of M' with $\tau=\tau_0$.

(c) The case that M' is reading the (j,n)-square with $\tau=\tau_2$.

In this case, M is reading the (1,n-j+1)-square. Thus, in a similar manner as in (b), M' brings the input head to the $\tau_2(j,n)$ = (1,n-j+1)-square. Then, simulating the movements of M, M' finishes the one series of M' with $\tau=\tau_0$.

(d) The case that M' is reading the border symbol B (without satisfying the conditions in (a)-(c), so far).

If M' is reading the border symbol at the north edge, the one series of M' is finished at this point, and the transformation τ is unchanged.

If M' is reading it at the (j,0)-square, then M' brings the head to the (0,j)-square (in the same way as in (b)), and finishes the one series of M' with $\tau = \tau \cdot \tau_1$ (righthanded τ is old τ).

If M' is reading it at the (j,n+1)-square, then

M' brings the head to the (0,n-j+1)-square, and finishes the one series of M' with $\tau=\tau\cdot\tau_2$, likewise.

(2) If M' has finished the simulation of M up to (s+1) steps without satisfying the conditions in (a)-(d), M' knows that M must have been reading only the blank symbol # during s steps of the simulation (except the first step). So M will repeat some subsequence of these s steps of movements until M next reads an element of $\Sigma \bigcup \{B\}$. Now, consider the sequence of (s+1) steps of movements memorized by M'. It is $(p_1,\alpha_1),\cdots,(p_i,\alpha_i),\cdots,(p_{s+1},\alpha_{s+1})$, where p_i is the internal state of M at the i-th step and $\alpha_i \in \{+1,0,-1\}$ is the horizontal shift direction(i.e. +1,0,-1 are right-shift, non-shift, and left-shift, respectively) of the input head of M' at the i-th step. Clearly, there exist $2\leq\ell<m\leq s+1$ such that $p_\ell=p_m$. Let $\alpha = \sum_{i=\ell}^{m-1} \alpha_i$. Suppose M' continues to simulate M after the (s+1)-th step. If $\alpha\leq 0$, M' never reads the border symbol at the east edge, nor the square in the n-th column with $\tau=\tau_2$. Conversely, If $\alpha\geq 0$, M' never reads the border symbol at the west edge, nor the square in the first column with $\tau=\tau_1$.

Thus M' traces (s+1) steps of the simulation of M in the backward direction until M' reaches the first computational configuration of the one series of M. This can easily be done, since M' has the record of all the movements of this period in the finite-state control. Then M' retries to simulate the one series of M from the position of the (0,j)- or (1,j)-square ($1\leq j\leq n$), in the following way.

If $\alpha \leq 0$, moving the input head from the (0,j)-square (or (1,j)-square) to the (j,0)-square (or (j,1)-square), M' begins to simulate M with $\tau = \tau \cdot \tau_1$. Similarly, if $\alpha > 0$, moving the input head from the (0,j)-square (or (1,j)-square) to the (n-j+1,n+1)-square (or (n-j+1,n)-square), M' begins to simulate M with $\tau = \tau \cdot \tau_2$.

If M does not loop in this series, M' will eventually satisfy one of the conditions in (a)-(d) during the simulation of M. But, apparently, M' never reads the border symbol of the south edge, nor the square of the n-th row with $\tau = \tau_3$. M' performs the operation corresponding to each case of (a)-(d), and finishes the one series of M'. If M loops, then M' also loops, but M' never reads the border symbol of the south edge in this case, too.

By above procedures (1) and (2), M' simulates M series by series, and accepts the input if and only if M accepts it. This M' never reads the border symbol of the south edge. (Q.E.D.)

From Lemma 4.14, the following theorem can be obtained.

Theorem 4.15 $\mathcal{L}[\mathrm{RA}(2)] = \mathcal{L}[\widehat{\mathrm{RA}}(2)]$.

Proof. $\mathcal{L}[\mathrm{RA}(2)] \subseteq \mathcal{L}[\widehat{\mathrm{RA}}(2)]$ is obvious from Lemma 4.14. Thus we show $\mathcal{L}[\mathrm{RA}(2)] \supseteq \mathcal{L}[\widehat{\mathrm{RA}}(2)]$.

Let $\hat{M}$ be an arbitrary $\widehat{\mathrm{RA}}(2)$, and let s and n be the

number of internal states of $\hat{M}$ and the length of a given input word. If $\hat{M}$ eventually halts for the input, $\hat{M}$ never reads the square below the s(n+2)-th row. Thus, in a similar method as in Theorem 4.10, we can construct $M \in RA(2)$ which simulates $\hat{M}$. The finite-state control of $\hat{M}$ is simulated by that of M. If the position of the input head of $\hat{M}$ is (i,j) ($0 \leq i \leq n+1$, $0 \leq j < s(n+2)$), then M places its input head at the (i,[j/s])-square and the remainder of j/s is remembered in the finite-state control. (Q.E.D.)

The relation between $\widehat{RA}(2)$ and 1MCA(1) is as follows.

Theorem 4.16 $\mathcal{L}[\widehat{RA}(2)] \subseteq \mathcal{L}[1MCA(1)]$.

Proof. It is easily derived from the fact that $\widehat{RA}(2)$ is equivalent to 1MCA(1) with the restriction that it can know what the input symbol is, only when the counter is empty. (Q.E.D.)

In the case of one input symbol, the following theorem can be derived.

Theorem 4.17 $\mathcal{L}[RA^1(2)] = \mathcal{L}[\widehat{RA}^1(2)] = \mathcal{L}[1MCA^1(1)]$.

Proof. Theorem 4.15 is independent of the number of input symbols, so $\mathcal{L}[RA^1(2)] = \mathcal{L}[\widehat{RA}^1(2)]$. The relation $\mathcal{L}[\widehat{RA}^1(2)] = \mathcal{L}[1MCA^1(1)]$ can be derived in a similar manner

as in Theorem 4.4 where $\mathcal{L}[RA^1(1)] = \mathcal{L}[1BCA^1(1)]$ was shown. (Q.E.D.)

From Theorem 4.2, 4.4, and 4.17, we can see that $RA^1(2)$, $\widehat{RA}^1(2)$, $1MCA^1(1)$, $1BCA^1(1)$, and $1MHA^1(2)$ are all equivalent (i.e. $\mathcal{L}[RA^1(2)] = \mathcal{L}[\widehat{RA}^1(2)] = \mathcal{L}[1MCA^1(1)] = \mathcal{L}[1BCA^1(1)] = \mathcal{L}[1MHA^1(2)]$).

The relation among RA(1), 1MHA(1), 1MMA(1), and RA(2) is as follows.

Theorem 4.18

$\mathcal{L}_3 = \mathcal{L}[RA(1)] = \mathcal{L}[1MHA(1)] = \mathcal{L}[1MMA(1)] \subsetneq \mathcal{L}[RA(2)]$.

Proof. $\mathcal{L}_3 = \mathcal{L}[RA(1)] = \mathcal{L}[1MHA(1)]$ is obvious. $\mathcal{L}_3 = \mathcal{L}[1MMA(1)]$ is shown by Blum and Hewitt [5]. $\mathcal{L}_3 \subsetneq \mathcal{L}[RA(2)]$ is shown in Theorem 4.13. (Q.E.D.)

Corollary 4.19

$\mathcal{L}_3^1 = \mathcal{L}[RA^1(1)] = \mathcal{L}[1MHA^1(1)] = \mathcal{L}[1MMA^1(1)] \subsetneq \mathcal{L}[RA^1(2)]$.

Fig. 4.12 and 4.13 summerizes the results obtained in section 4.2 and 4.3.

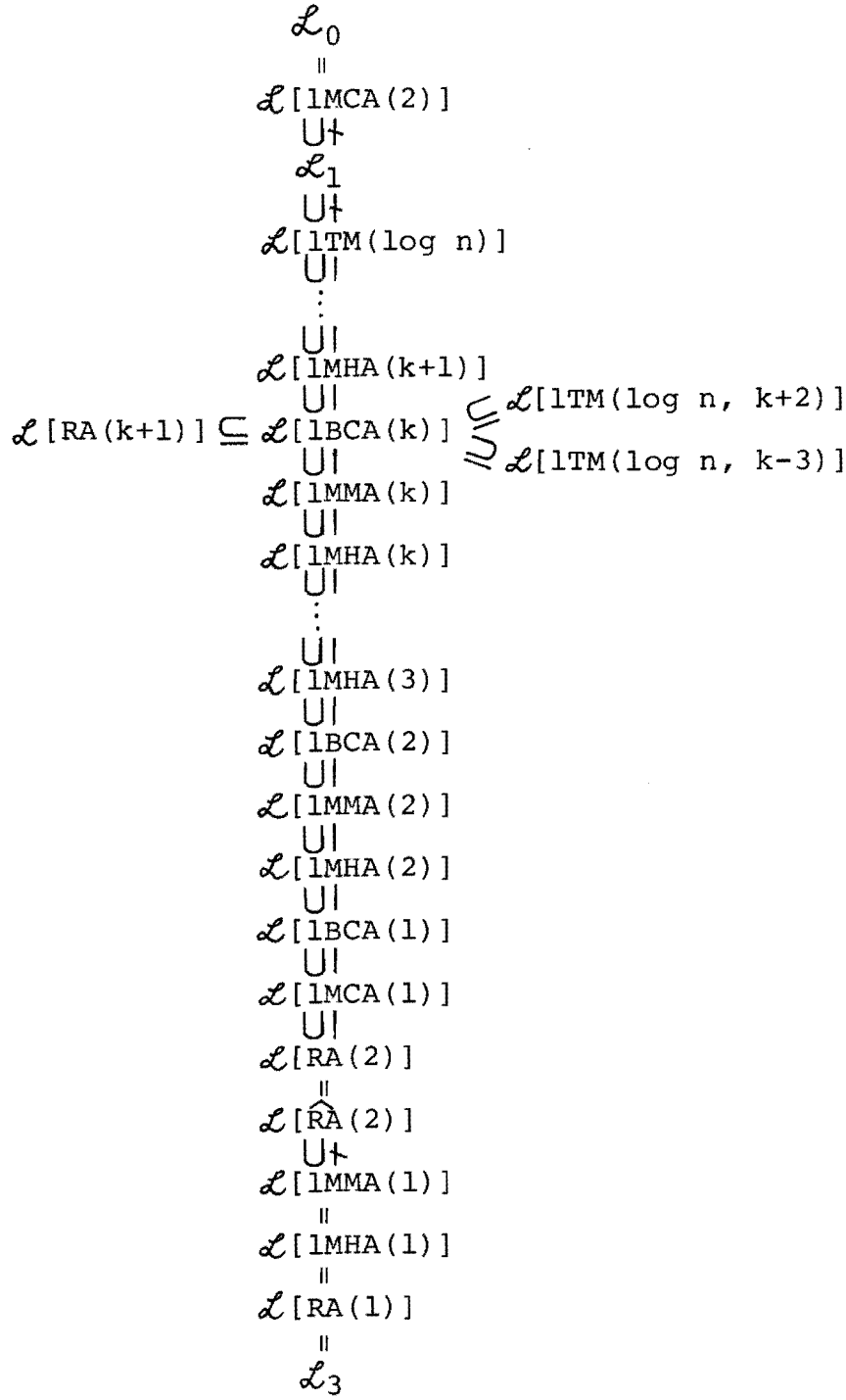

Fig. 4.12 The relation among various automata of tape complexity log n.

$$
\begin{array}{c}
\mathcal{L}_0^1 \\
\| \\
\mathcal{L}[1MCA^1(2)] \\
\cup\!\!\!\!\not\;\; \\
\mathcal{L}_1^1 \\
\cup\!\!\!\!\not\;\; \\
\mathcal{L}[1TM^1(\log n)] \\
\cup\!| \\
\vdots \\
\cup\!| \\
\mathcal{L}[1MHA^1(k+1)] \\
\| \\
\mathcal{L}[RA^1(k+1)] = \mathcal{L}[1BCA^1(k)] \quad \begin{array}{l} \subseteq \mathcal{L}[1TM^1(\log n,\ k+2)] \\ \supseteq \mathcal{L}[1TM^1(\log n,\ k-3)] \end{array} \\
\cup\!| \\
\mathcal{L}[1MMA^1(k)] \\
\cup\!| \\
\mathcal{L}[1MHA^1(k)] \\
\cup\!| \\
\vdots \\
\cup\!| \\
\mathcal{L}[1MHA^1(3)] \\
\| \\
\mathcal{L}[1BCA^1(2)] \\
\cup\!| \\
\mathcal{L}[1MMA^1(2)] \\
\cup\!| \\
\mathcal{L}[1MHA^1(2)] \\
\| \\
\mathcal{L}[1BCA^1(1)] \\
\| \\
\mathcal{L}[1MCA^1(1)] \\
\| \\
\mathcal{L}[RA^1(2)] \\
\| \\
\mathcal{L}[\widehat{RA}^1(2)] \\
\cup\!\!\!\!\not\;\; \\
\mathcal{L}[1MMA^1(1)] \\
\| \\
\mathcal{L}[1MHA^1(1)] \\
\| \\
\mathcal{L}[RA^1(1)] \\
\| \\
\mathcal{L}_3^1
\end{array}
$$

Fig. 4.13 The relation among various automata of tape complexity log n (with one input symbol).

4.4 Hierarchy

In this section, we consider the subhierarchy of language acceptabilities in the class of tape complexity log n.

Blum and Hewitt [5] first showed the existence of infinite hierarchy of 2MMA with respect to the number of markers. Using the diagonalization technic, they derived that 2MMA(2k+4) is strictly more powerful than 2MMA(k). Later, this result was improved by Ibarra [22] and Monien [30] (they showed the hierarchy of 1MHA).

Theorem 4.20 (Monien) The following relation holds for any k (k=1,2,$\cdots$), provided that the number of input symbols is more than one.

$$\mathcal{L}[\text{1MHA}(k)] \subsetneq \mathcal{L}[\text{1MHA}(k+1)]$$

From Theorem 4.1, 4.5 and 4.6, the hierarchies for 1MMA, 1BCA, and 1TM(log n) can also be obtained.

Theorem 4.21 The following relations hold for any k (k=1,2,$\cdots$), provided that the number of input symbols is more than one.

$$\mathcal{L}[\text{1MMA}(k)] \subsetneq \mathcal{L}[\text{1MMA}(k+2)]$$

$$\mathcal{L}[\text{1BCA}(k)] \subsetneq \mathcal{L}[\text{1BCA}(k+2)]$$

$$\mathcal{L}[\text{1TM}(\log n, k)] \subsetneq \mathcal{L}[\text{1TM}(\log n, k+7)]$$

Proof. From Theorem 4.1 and 4.19, $\mathcal{L}[1MMA(k)] \subseteq \mathcal{L}[1BCA(k)] \subseteq \mathcal{L}[1MHA(k+1)] \subsetneq \mathcal{L}[1MHA(k+2)] \subseteq \mathcal{L}[1MMA(k+2)] \subseteq \mathcal{L}[1BCA(k+2)]$. And, from Theorem 4.5 and 4.6, $\mathcal{L}[1TM(\log n, k)] \subseteq \mathcal{L}[1BCA(k+3)] \subsetneq \mathcal{L}[1BCA(k+5)] \subseteq \mathcal{L}[1TM(\log n, k+7)]$.

(Q.E.D.)

However, it is not known whetheher there exists a proof of a more refined result.

Theorem 4.19 and 4.20 hold only for the case of two or more input symbols. We now consider the case of one input symbol. First we derive the infinite hierarchy of $1BCA^1$.

Theorem 4.22 $\mathcal{L}[1BCA^1(k)] \subsetneq \mathcal{L}[1BCA^1(2k+8)]$.

Proof. Let A be an arbitrary $1BCA^1(k)$. We first consider a method to encode A into a natural number. Let $K_A = \{q_0, q_1, \cdots, q_{s-1}\}$ be the set of internal states of A (A has s states). We assume, without loss of generality, that q_0 is the initial state and $\{q_1\}$ is the set of final states. And let the set of input symbols of A be $\Sigma=\{a\}$. The transition function δ of A is a mapping from a subset of $S_1 = K_A \times \{a, ¢, \$\} \times \{z_0, x, f\}^k$ into $S_2 = K_A \times \{L, R, H\} \times \{+1, 0, -1\}^k$. Let us consider mappings h_1 and h_2 defined as follows. h_1 is a mapping from S_1 into $\{1, 2, \cdots, s \cdot 3^{k+1}\}$, and satisfies

$$h_1(q_\ell, x, y_1, \cdots, y_k) = \ell \cdot 3^{k+1} + t + 1$$

for any $q_\ell \in K_A$, $x \in \{a, ¢, \$\}$, $y_i \in \{z_0, x, f\}$. t is the value of a ternary number of (k+1) figures $x\ y_1\ y_2 \cdots y_k$, where

the elements of $\{a,¢,\$\}$ and $\{z_0,x,f\}$ are regarded as the number 0, 1, 2, respectively. The mapping $h_2 : S_2 \to \{1,2, \cdots, s\cdot 3^{k+1}\}$ is also defined likewise. Apparently h_1 and h_2 are bijections, so that there are inverse mapping for them. Now, we define a mapping $\delta' : \{1,2,\cdots, s\cdot 3^{k+1}\} \to \{0,1,\cdots, s\cdot 3^{k+1}\}$ as follows.

$$\delta'(v) = h_2(\delta(h_1^{-1}(v)))$$
$$(v \in \{1,2,\cdots, s\cdot 3^{k+1}\})$$

Note that, for $h_1^{-1}(v) \in S_1$ on which δ is not defined, we assume $\delta'(v)=0$. Here we encode A into a natural number (Gödel number). The Gödel number of A is

$$n_A = \prod_{i=1}^{s\cdot 3^{k+1}} p_i^{\delta'(i)},$$

where p_i is the i-th prime (i.e. $p_1=2$, $p_2=3$, $p_3=5,\cdots$).

Next, we construct $B \in \mathrm{lBCA}^1(2k+8)$ which diagonalize all $\mathrm{lBCA}^1(k)$. Suppose an input of length n is given to B. Then B simulates the movements of $A_n \in \mathrm{lBCA}^1(k)$ with the input of length n, where the Gödel number of A_n is n. If A_n accepts the input, B does not accept it. And if A_n rejects or loops, then B accepts.

In order to construct such B, we first explain that B can execute the following primitive operations. Let u_1, $u_2,\cdots,u_{2k+8}$ be the 2k+8 nbc's of B, and let $|u_j|$ denote the number counted by u_j. (c is a constant.)

(1) $|u_j| \to |u_j,|$.

(2) $|u_j| + c \to |u_j|$.

(3) $|u_j| - c \to |u_j|$.

(4) $|u_j| \times c \to |u_j|$.

(5) $|u_j| \div c \to |u_j|$ $\cdots$ quotient.

(6) $|u_j| \div |u_{j'}| \to |u_{j''}|$ $\cdots$ quotient, $|u_j|$ $\cdots$ remainder.
(If only the remainder is needed, $u_{j''}$ is not used.)

(7) Deciding whether $|u_j|=|u_{j'}|$.

Note that u_{2k+8} is used as a working nbc to perform these operations (thus $1 \leq j,j',j'' \leq 2k+7$). For example, the operation (5), the division by a constant c, is executed as follows. B first sets $|u_{2k+8}|=0$. Then B repeats to add 1 to $|u_{2k+8}|$, every time B subtracts c from $|u_j|$ until $|u_j|$ becomes 0. By this, B can gain the quotient in u_{2k+8}, so B transfers it into u_j. The remainder can be obtained in the finite-state control of B. The other operations (1)-(7) can also be executed by using only u_{2k+8} as a working nbc. The contents of nbc's other than u_{2k+8} and the nbc in which the answer is to be held are unchanged.

Using these operations, B simulates A_n in the following manner. The input head of A_n is simulated by that of B. The k nbc's of A_n are simulated by $u_1,\cdots,u_k$, and the internal state of A_n is kept in u_{k+1}. The k+2 nbc's $u_{k+2},\cdots, u_{2k+3}$ are used to check whether A_n is looping or not, and the remaining 5 nbc's, $u_{2k+4},\cdots, u_{2k+8}$ are used for working to transit the computational configuration of A_n. B begins to simulate A_n, setting all the 2k+8 nbc's empty.

Now, we show how to obtain $\delta(\mathbf{x}) = h_2^{-1}(\delta'(h_1(\mathbf{x})))$ from $\mathbf{x} \in K_{A_n} \times \{a,¢,\$\} \times \{z_0,x,f\}^n$, a present situation of A_n (K_{A_n} is a set of internal states of A_n), in order to simulate each

step of A_n. As $\mathbf{x}$ is known from $u_1,\cdots, u_{k+1}$ and the symbol being read by the head of B, B calculates $h_1(\mathbf{x})$ from these informations and stores it in u_{2k+4}. This can be done as follows. First B makes $|u_{k+1}|\times 3^{k+1} + 1 \to |u_{2k+4}|$, and then depending on the counter symbol (z_0 or x or f) of u_j (j=1,2, $\cdots$, k), B adds the constant $0\times 3^{k-j}$ or $1\times 3^{k-j}$ or $2\times 3^{k-j}$, respectively. Finary B adds the constant 0×3^k or 1×3^k or 2×3^k depending on the tape symbol. If u_{2k+4} overflows in these calculations, B concludes that $\delta(\mathbf{x})$ is not defined.

Next, B calculates $p_{h_1(\mathbf{x})}$ from $|u_{2k+4}| = h_1(\mathbf{x})$. B can easily decide whether $|u_{2k+5}|$ is a prime number, with a help of $u_{2k+6},\cdots,u_{2k+8}$. Thus, B first sets $|u_{2k+5}|=2$ as an initial value, and then, repeats to check whether $|u_{2k+5}|$ is a prime number every time $|u_{2k+5}|$ is increased by 1. If $|u_{2k+5}|$ is a prime number, B subtracts 1 from $|u_{2k+4}|$. Repeating this procedure until $|u_{2k+4}|$ becomes 0, B can get $p_{h_1(\mathbf{x})}$ in u_{2k+5}.

$\delta'(h_1(\mathbf{x}))$, i.e. the number of times that $p_{h_1(\mathbf{x})}$ divides n without remainder, can also be obtained in u_{2k+4} with a help of $u_{2k+5},\cdots, u_{2k+8}$.

Finally, B computes $\delta(\mathbf{x}) = h_2^{-1}(\delta'(h_1(\mathbf{x})))$ from $|u_{2k+4}| = \delta'(h_1(\mathbf{x}))$, and transits the computational configuration of A_n. The next internal state of A_n is the quotient of $(|u_{2k+4}| -1)\div 3^{k+1}$, so B stores it in u_{k+1}. The alteration of each nbc and the shift direction of the input head of A_n can be known by replacing the remainder of $(|u_{2k+4}|-1)\div 3^{k+1}$ into a ternary number. Thus B increases or decreases $u_1,\cdots,u_k$ and shifts the input head according to them. (If $|u_{2k+4}|=0$,

then $\delta(\mathbf{x})$ is undefined.)

By the method described above, B simulates A_n step by step. Every time B advances one step of the simulation, B examines if A_n has become a final state (i.e. $|u_{k+1}|=1$). If so, B halts in a rejecting state. Otherwise B proceeds the simulation of A_n. In order to check if A_n loops, B counts the number of steps of A_n up to n^{k+2} steps using $u_{k+2}, \cdots, u_{2k+3}$. The total number of computational configurations of A_n is $s \cdot n^{k+1}$. Furthermore $s \cdot n^{k+1} < n^{k+2}$ holds, because $n > s$. Thus B can know whether A_n loops or not. B halts in an accepting state, if A_n halts in a rejecting state or loops.

Suppose some $A' \in 1BCA(k)$ accepts the same language as B. Then a contradiction occurs, if we give A' an input tape whose length is the Gödel number of A'. Thus $\mathcal{L}[1BCA^1(k)] \subsetneq \mathcal{L}[1BCA^1(2k+8)]$ is concluded. (Q.E.D.)

By this theorem, it is seen that there exists an infinite hierarchy among $1BCA^1$, too. And from Theorem 4.2, 4.4, and 4.7 we can also derive the hierarchies of $1MHA^1$, $1MMA^1$, RA^1, and $1TM^1(\log n)$.

Theorem 4.23

$\mathcal{L}[1MMA^1(k)] \subsetneq \mathcal{L}[1MMA^1(2k+9)]$,

$\mathcal{L}[1MHA^1(k)] \subsetneq \mathcal{L}[1MHA^1(2k+7)]$,

$\mathcal{L}[RA^1(k)] \subsetneq \mathcal{L}[RA^1(2k+7)]$,

$\mathcal{L}[1TM^1(\log n, k)] \subsetneq \mathcal{L}[1TM^1(\log n, 2k+16)]$.

Proof. These are derived from the following relations.
$\mathcal{L}[1MMA^1(k)] \subseteq \mathcal{L}[1BCA^1(k)] \subsetneq \mathcal{L}[1BCA^1(2k+8)] \subseteq \mathcal{L}[1MMA^1(2k+9)]$.
$\mathcal{L}[1MHA^1(k)] = \mathcal{L}[RA^1(k)] = \mathcal{L}[1BCA^1(k-1)] \subsetneq \mathcal{L}[1BCA^1(2k+6)]$
$= \mathcal{L}[1MHA^1(2k+7)] = \mathcal{L}[RA^1(2k+7)]$. $\mathcal{L}[1TM^1(\log n, k)] \subseteq$
$\mathcal{L}[1BCA^1(k+3)] \subsetneq \mathcal{L}[1BCA^1(2k+14)] \subseteq \mathcal{L}[1TM^1(\log n, 2k+16)]$.

(Q.E.D.)

We can also show the hierarchy of RA (where the number of input symbols is not restricted) from this.

Theorem 4.24 $\mathcal{L}[RA(k)] \subsetneq \mathcal{L}[RA(2k+7)]$.

Proof. Let $\Sigma_1=\{a\}$, and let $\mathcal{L}^1[RA(k)] = \{L \mid L=L' \cap (\Sigma_1)^+, L' \in \mathcal{L}[RA(k)]\}$. As it is easily seen that $\mathcal{L}[RA^1(k)] = \mathcal{L}^1[RA(k)]$, $\mathcal{L}^1[RA(k)] = \mathcal{L}[RA^1(k)] \subsetneq \mathcal{L}[RA^1(2k+7)] = \mathcal{L}^1[RA(2k+7)]$ holds. Thus $\mathcal{L}[RA(2k+7)] \cap \overline{\mathcal{L}[RA(k)]} \supset \mathcal{L}^1[RA(2k+7)] \cap \overline{\mathcal{L}^1[RA(k)]} \neq \emptyset$, and this completes the proof. (Q.E.D.)

4.5 Concluding Remarks

In this chapter, various automata of tape complexity log n have been studied. 1BCA and RA were newly defined here, and their relations to 1MHA and 1MMA were precisely considered.

In section 4.3, the acceptability of RA(2) was particularly investigated. RA(2) is a very interesting automaton, because

it is the weakest class of automata we can imagine, in the class of tape complexity log n. However, RA(2) can accept some subclass of context-sensitive languages which is properly including the class of regular languages. Furthermore, some results concerning RA(2) were derived from the two-dimensional properties of the input tape. It is interesting that the relation $\mathcal{L}[1MCA^1(1)] = \mathcal{L}[1BCA^1(1)]$ was derived from such two-dimensional properties.

The hierarchy of 1MHA(k) has already been shown by Monien [30] and some others. But it was the case of two or more input symbols. In section 4.4, we studied the case of one input symbol, and derived $\mathcal{L}[1BCA^1(k)] \subsetneq \mathcal{L}[1BCA^1(2k+8)]$ using the diagonalization technic. The chief point of this proof is to encode each $1BCA^1(k)$ into an integer (Gödel number). But it is not known whether this result can be refined.

In this chapter, many problems are remaining open. The notation $A \subseteq B$ in Fig.4.12 or 4.13 means that it is not yet known whether A is a proper subset of B or A=B. The relation between the class of context-free languages ($\mathcal{L}_2$) and $\mathcal{L}[1TM(\log n)]$ is also an important unsolved problem.

CHAPTER 5

TURING TRANSDUCERS

All the automata introduced in chapter 2-4 were acceptors of one- or two-dimensional languages. However, if output tapes are attached to these automata, they can be regarded as transducers which translate input words into output words. We may think that they are computers of recursive mappings.

In this chapter, an L(n) tape-bounded Turing transducer is proposed. Here, we only consider the case that both the numbers of input symbols and output symbols are restricted to one, in order to investigate its computing ability of recursive functions. Some results concerning the L(n) tape-bounded Turing machine (acceptor) also hold for the Turing transducer, because an acceptor can be thought to compute a 0-1 valued recursive function. But there also exist many properties which are characteristic of the transducer, such as the increasing degree of a function computed by it.

Furthermore, we propose a finite-state transducer, a multi-head transducer and a multi-counter transducer, and investigate the precise relations among their computing abilities of functions.

5.1 Definitions

Let us consider a Turing transducer T illustrated in Fig.5.1.

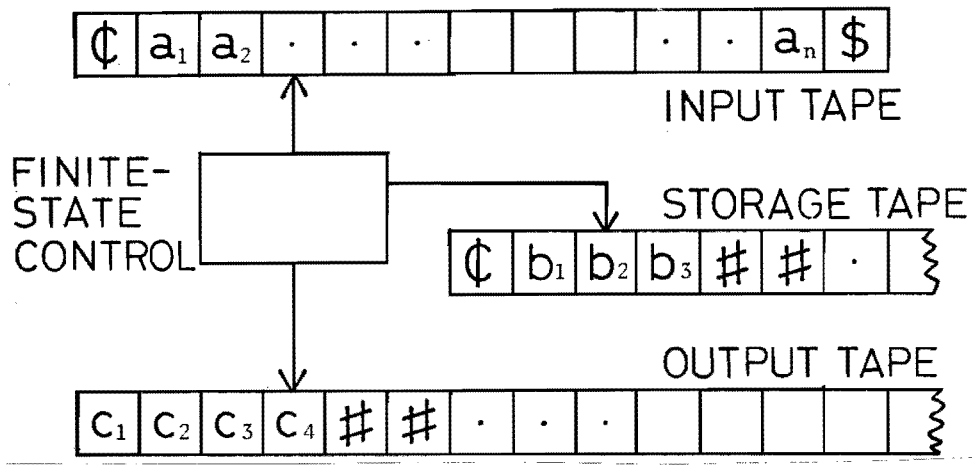

Fig. 5.1 A Turing transducer.

T consists of an input tape, a two-way read-only input head, a storage tape, a two-way storage tape head, an output tape, a one-way write-only output head, and a finite-state control. The input tape, the input head, the storage tape, and the storage tape head are the same as in 1TM. The output tape is a one-dimensional semi-infinite tape, and divided into squares. The output head can move only in one way on the output tape, and can write the output symbols. Depending on the present internal state, and the symbols read by the input head and the storage tape head, the finite-state control determines the storage tape symbol to be written, the output symbol to be written (or T may not write any output symbol),

the move directions of the input head and the storage tape head, and the next internal state.

Generally, the numbers of the input symbols and the output symbols are arbitrary. But here, in order to consider the computing ability of number-theoretic functions, we restrict both these numbers to one.

Formally, a *deterministic Turing transducer* (abbreviated to TT) is a 10-tuple

$$T = (K, \{a\}, \Gamma, \{c\}, \delta, q_0, ¢, \$, \#, F),$$

where K is a nonempty finite set of internal states, $\{a\}$ and $\{c\}$ are the sets of input symbols and output symbols (each of them consists of single element), Γ is a nonempty finite set of storage tape symbols, $q_0 \in K$ is an initial state, ¢ and \$ are border symbols of the input tape and the storage tape, # is a blank symbol of the storage tape and the output tape, $F \subseteq K$ is a set of final states. δ is a mapping from a subset of $K\times\{a,¢,\$\}\times(\Gamma \cup \{¢,\#\})$ into $K\times(\Gamma \cup \{¢\})\times\{c,\varepsilon\}\times\{L,R,H\}^2$, where ε means that T does not write the output symbol, and $\{L,R,H\}$ is the set of move directions of the input head or the storage tape head. T halts for the element of $K\times\{a,¢,\$\}\times(\Gamma \cup \{¢,\#\})$ on which δ is not defined. If T writes the output symbol c, the output head automatically shifts to the right by one square.

Now, let us give an input word $w \in \{a\}^*$ of length n (i.e. the word a^n $(n=0,1,\cdots)$) to T. Assume that T begins its movements from the initial state q_0 setting all the heads at the left side ends. If T halts in a final state with

an output word of length m (i.e. c^m), then we say that the nonnegative integer n is mapped to m by T. If T halts in a state other than the final states or loops for the input a^n, then n is not mapped to any integer. Thus T defines a function f which maps a subset of $\mathbf{N} \cup \{0\}$ into $\mathbf{N} \cup \{0\}$. We say that the function f is *computed* by T.

Let L(n) be a mapping from $\mathbf{N}$ into $\mathbf{R}_+$. If T does not scan more than [L(n)] squares of the storage tape for the input a^n, T is said to be a *deterministic L(n) tape-bounded Turing transducer* (abbreviated to TT(L(n))). The class of recursive functions computed by TT(L(n)) is denoted by $\mathcal{F}$[TT(L(n))].

L(n) is called a tape function, and the notion of a constructible tape function is the same as in 1TM(L(n)).

Furthermore, the notions of the computational configuration and the storage state can be defined in a similar way as in 2TM(L(m,n)).

5.2 Computing Ability of TT(L(n))

In this section, we investigate the properties of TT(L(n)) and the functions computed by it.

5.2.1 Relation to 1TM(L(n))

As we have seen in section 1.2, many theorems concerning

1TM(L(n)) have already been shown. Some technics used in the proofs of these theorems are also applicable to the case of TT(L(n)). So, we now show some properties of TT(L(n)) which can be derived in the same way as in 1TM(L(n)) (or $1TM^1(L(n))$).

First, the tape reduction theorem of TT(L(n)) is as follows.

Theorem 5.1 Let L(n) be a tape function of TT. Then for any constant $c>0$,

$$\mathcal{F}[TT(L(n))] = \mathcal{F}[TT(c\cdot L(n))].$$

The hierarchy theorems of 1TM(L(n)) (i.e. Theorem 1.2 and 1.3) hold only for the case of two or more input symbols. In the case of one input symbol, however, the following result has been shown.

Proposition 5.2 (Hartmanis and Berman [15]) Let $L_1(n)$ and $L_2(n)$ be constructible tape functions of $1TM^1$. Suppose that

$$\lim_{i\to\infty} \frac{L_1(n_i)}{L_2(n_i)} = 0$$

$$\lim_{i\to\infty} L_2(n_i) = \infty$$

for some recursively enumerable sequence of natural numbers $\{n_i\}$. Then there exists a language $L\subset\{a\}^*$ such that $L\in \mathcal{L}[1TM^1(L_2(n))]$ but $L\notin \mathcal{L}[1TM^1(L_1(n))]$.

The next theorem can be directly derived from this.

Theorem 5.3 Let $L_1(n)$ and $L_2(n)$ be constructible tape functions of TT. Suppose that

$$\lim_{i\to\infty} \frac{L_1(n_i)}{L_2(n_i)} = 0$$

$$\lim_{i\to\infty} L_2(n_i) = \infty$$

for some recursively enumerable sequence of natural numbers $\{n_i\}$. Then there exists a 0-1 valued function f such that $f \in \mathcal{F}[TT(L_2(n))]$ but $f \notin \mathcal{F}[TT(L_1(n))]$.

The property concerning the lower bounds on tape complexity of 1TM(L(n)) (Theorem 1.4) also holds for TT(L(n)).

Theorem 5.4 Let L(n) be a constructible tape function of TT. If $\limsup_{n\to\infty} L(n) = \infty$, then

$$\limsup_{n\to\infty} \frac{L(n)}{\log \log n} > 0.$$

Now, as for $1TM^1$, the following theorem is known.

Proposition 5.5 (Hartmanis and Berman [15]) Let L(n) be a tape function of $1TM^1$, and let $L \subset \{a\}^*$ be some language. If $L \in \mathcal{L}[1TM^1(L(n))]$, then $\bar{L} \in \mathcal{L}[1TM^1(L(n))]$, where $\bar{L} = \{a\}^* - L$.

In order to prove this proposition, Hartmanis and Berman

showed that for any $M \in 1TM^1(L(n))$ there exists some $M' \in 1TM^1(L(n))$ which accepts the same language as M and halts for any input. The following theorem can be derived using the same argument as this (a similar argument was used to prove Lemma 2.6).

Theorem 5.6 Let L(n) be a tape function of TT. For any $M \in TT(L(n))$, there exists $M' \in TT(L(n))$ which computes the same function as M and halts for any input.

5.2.2 Increasing Degree of Functions

Now, we consider how the increasing degree of functions computed by TT(L(n)) is affected by the tape function L(n).

The hierarchy theorem stated in Theorem 5.3 is independent of the increasing degree of functions. But the following theorem shows that the rapidity of increase also forms an infinite hierarchy. Indeed, the rapidity of increase is bounded by the tape function.

Theorem 5.7 Let $L_1(n)$ and $L_2(n)$ be constructible tape functions of TT. Suppose that

$$\lim_{i\to\infty} \frac{L_1(n_i)}{L_2(n_i)} = 0$$

$$\lim_{i\to\infty} L_2(n_i) = \infty$$

for some increasing sequence of natural numbers $\{n_i\}$. Then there exists some function $f \in \mathcal{F}[TT(L_2(n))]$ which satisfies

$$\lim_{i\to\infty} \frac{g(n_i)}{f(n_i)} = 0$$

for any $g \in \mathcal{F}[TT(L_1(n))]$ such that $g(n_i)$ is defined for all n_i.

Proof. Let $f(n) = n \cdot 2^{[L_2(n)]}$. We can construct $M \in TT(L_2(n))$ which computes f, in the following way. Let M' be one of $TT(L_2(n))$ which constructs the tape function $L_2(n)$. M first simulates M', and marks $[L_2(n)]$ squares of the storage tape. Then M begins to count a number from 0 to $2^{[L_2(n)]}-1$ consecutively. Every time M counts up the number, M generates the output of length n by scanning the input tape. By this, M can generate the number $n \cdot 2^{[L_2(n)]}$.

Now, let g be any function which is defined on the sequence $\{n_i\}$ and is computed by some $T \in TT(L_1(n))$. Let s and t be the numbers of internal states and storage tape symbols of T. Then, the total number of computational configurations of T for the input of length n is at most $s \cdot n \cdot [L_1(n)] \cdot t^{[L_1(n)]}$, and g(n) must be less than or equal to this number. Thus, $g(n) \leq s \cdot n \cdot [L_1(n)] \cdot t^{[L_1(n)]}$, and from

$$\lim_{i \to \infty} \frac{L_1(n_i)}{L_2(n_i)} = 0,$$

we can conclude

$$\lim_{i \to \infty} \frac{g(n_i)}{f(n_i)} = 0.$$

(Q.E.D.)

Next, we consider the relation between the slowness of increase and the tape complexity. The following theorem shows that the slowness of increase does not form a hierarchy,

i.e., for any unbounded tape function L(n), $\mathcal{F}[TT(L(n))]$ contains an "arbitrarily slow" function. The proof of this theorem is essentially the same as in Theorem 2.27.

Theorem 5.8 Let L(n) be a constructible tape function of TT that satisfies $\limsup_{n\to\infty} L(n) = \infty$. Let $f : \mathbf{N} \to \mathbf{N}$ be any monotone nondecreasing total recursive function such that $\lim_{n\to\infty} f(n) = \infty$. Then there exists a function $g \in \mathcal{F}[TT(L(n))]$ which satisfies $g(n)<f(n)$ and $\limsup_{n\to\infty} g(n) = \infty$.

Proof. We construct $T \in TT(L(n))$ which computes the desired function g, as follows.

Let $\psi : \mathbf{N} \to \mathbf{N}$ be a function such that $\psi(x) = \min\{n \mid f(n+1)>x\}$. And let M' be a two-counter automaton which computes $[L(\psi(1))], [L(\psi(2))], \cdots$, and never halts (as M' in Theorem 2.27).

T first marks [L(n)] squares of the storage tape, and uses it so as to simulate two counters, each of which can count up to [L(n)]. Then, using these counters, T begins to simulate M' until they overflow. Every time T finishes the calculation of $[L(\psi(i))]$ $(i=1,2,\cdots)$, T writes one output symbol. By a similar argument as in Theorem 2.27, the function g computed by T satisfies $g(n)<f(n)$ and $\limsup_{n\to\infty} g(n) = \infty$, and this completes the proof. (Q.E.D.)

The following theorem shows that, when the tape function

L(n) grows more slowly than the order of log n, the function computed by TT(L(n)) must grow linearly on some infinite set of n. Its proof is based on the method of Hartmanis and Berman [15].

Theorem 5.9 Let L(n) be a tape function of TT which satisfies

$$\lim_{n\to\infty} \frac{L(n)}{\log n} = 0 .$$

Then for any $f \in \mathcal{F}[TT(L(n))]$ and for every sufficiently large n such that f(n) is defined, there exist an integer $i \geq 0$, integers $k_j > 0$ $(1 \leq j \leq i)$, and integers $m_j \geq 0$ $(1 \leq j \leq i)$. And the following equality holds for every integer $t \geq 0$.

$$f(n+t\cdot n!) = f(n) + t\cdot\left(\frac{m_1}{k_1} + \frac{m_2}{k_2} + \cdots + \frac{m_i}{k_i} \right)\cdot n!$$

Proof. Let M be TT(L(n)) which computes f, and let s and t be the numbers of internal states and storage tape symbols of M. From Theorem 5.5, we may assume, without loss of generality, that M always halts for any input. The total number of storage states of M for the input of length n is $s\cdot[L(n)]\cdot t^{[L(n)]}$. Let i be the number of times that M scans the input tape from one side end to another. Since M never loops, $0 \leq i \leq s\cdot[L(n)]\cdot t^{[L(n)]}$.

Now we consider the j-th scan of M $(1 \leq j \leq i)$. For sufficiently large n, the inequality $s\cdot[L(n)]\cdot t^{[L(n)]} < n$ holds, because $\lim_{n\to\infty}(L(n)/\log n) = 0$. Thus M must repeat some sequence

(cycle) of the storage states during the j-th scan (except the first transient sequence of storage states whose length is less than $s\cdot[L(n)]\cdot t^{[L(n)]}$). Let k_j be the shifting distance of the input head and m_j be the number of symbols written in the output tape, in one cycle. If we give M the input tape of length $n+t\cdot n!$ rather than n, then M will writes the output of length $(y_j + \frac{m_j}{k_j}\cdot t\cdot n!)$ in the j-th scan, where y_j is the length of the output for the input n in the j-th scan. Furthermore, when M reaches another side end of the input, M becomes the same storage state as in the case of the input n. Thus for the input $n+t\cdot n!$, M also scans i times and halts in the same storage state. In the case that M once leaves one side end of the input and returns to the same side end, or M is staying at the border symbol, M writes the same number of output symbols. So, let y_0 be the total number of output symbols in these cases.

Accordingly, for the input $n+t\cdot n!$, M writes the output

$$y_0 + \sum_{j=1}^{i} (y_j + \frac{m_j}{k_j}\cdot t\cdot n!).$$

Since $f(n) = \sum_{j=0}^{i} y_j$,

$$f(n+t\cdot n!) = f(n) + t\cdot(\frac{m_1}{k_1} + \cdots + \frac{m_i}{k_i})\cdot n!$$

is concluded. (Q.E.D.)

From this theorem, the following corollaries can be derived.

Corollary 5.10 Let L(n) be a tape function of TT such that $\lim_{n\to\infty}(L(n)/\log n) = 0$. If $f \in \mathcal{F}[TT(L(n))]$ is a total function, then at least one of the following inequalities holds.

(1) $\limsup_{n\to\infty}(f(n)/n) > 0$

(2) $\liminf_{n\to\infty} f(n) < \infty$

Proof. If there exists some $m_j \neq 0$ $(1 \leq j \leq i)$ for some sufficiently large n, then $\limsup_{n\to\infty}(f(n)/n) > 0$. For some n, if $m_j = 0$ for all $1 \leq j \leq i$, then $\liminf_{n\to\infty} f(n) < \infty$. (Q.E.D.)

Corollary 5.11 Let L(n) be a tape function of TT such that $\lim_{n\to\infty}(L(n)/\log n) = 0$, and let $f \in \mathcal{F}[TT(L(n))]$ be a total function that satisfies $\limsup_{n\to\infty} f(n) = \infty$. If one of the following conditions holds, then f is not a monotone non-decreasing function.

(1) $\limsup_{n\to\infty}(f(n)/n) = \infty$

(2) $\limsup_{n\to\infty}(f(n)/n) = 0$

Proof. From Theorem 5.9, $f(n_0+t\cdot n_0!) = f(n_0) + t\cdot K$ holds for sufficiently large n_0, where $K = \left(\frac{m_1}{k_1} + \cdots + \frac{m_i}{k_i}\right)n_0!$. Suppose f is monotone nondecreasing. Then the following inequality holds for all $n \geq n_0$.

$$f(n_0) + [(n-n_0)/n_0!]\cdot K \leq f(n) \leq f(n_0) + [(n-n_0)/n_0! + 1]\cdot K$$

Thus, $\limsup_{n\to\infty}(f(n)/n) = K/n_0!$. But since $\limsup_{n\to\infty} f(n) = \infty$, K must be positive $(K>0)$. So, this contradicts (1) and (2).

(Q.E.D.)

5.3 Multi-Head, Multi-Counter and Finite-State Transducers

In this section, we investigate the computabilities of multi-head, multi-counter and finite-state transducers. They can be considered as special cases of tape-bounded Turing transducers.

5.3.1 Multi-Head Transducer

A multi-head transducer consists of an input tape, a finite number of read-only input heads, an output tape, a one-way write-only output head, and a finite-state control (Fig.5.2).

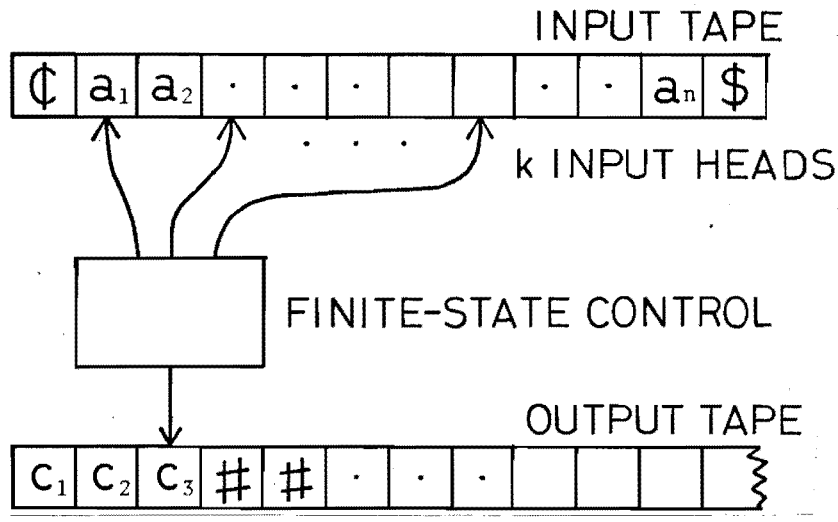

Fig. 5.2 A multi-head transducer.

Here, we also restrict both the numbers of input symbols and output symbols to one.

Formally a *deterministic two-way multi-head transducer* (abbreviated to 2WMHT) is a 10-tuple

$$M = (K, \{a\}, \{c\}, \delta, k, q_0, ¢, \$, \#, F),$$

where K, $\{a\}$, $\{c\}$, q_0, ¢, \$, # and F are the same as in TT, k is the number of input heads, and δ is a mapping from a subset of $K\times\{a,¢,\$\}^k$ into $K\times\{c,\varepsilon\}\times\{L,R,H\}^k$.

The class of deterministic two-way k-head transducers is denoted by 2WMHT(k). Thus, $2WMHT = \bigcup_{k=1}^{\infty} 2WMHT(k)$.

The function computed by $M \in 2WMHT(k)$ is defined in a similar way as in TT(L(n)). The class of functions computed by 2WMHT(k) (or 2WMHT) is denoted by $\mathcal{F}$[2WMHT(k)] (or $\mathcal{F}$[2WMHT]).

A *deterministic one-way multi-head transducer* (1WMHT) is the same one as 2WMHT except that each head can move only in one way. Thus the transition function δ of 1WMHT is a mapping from a subset of $K\times\{a,\$\}^k$ into $K\times\{c,\varepsilon\}\times\{R,H\}^k$. The class of deterministic one-way k-head transducers is denoted by 1WMHT(k).

Especially, 2WMHT(1) (or 1WMHT(1)) is called a *deterministic two-way* (or *one-way*) *finite-state transducer* and denoted by 2WFST (or 1WFST).

5.3.2 Multi-Counter Transducer

A multi-counter transducer consists of an input tape, a read-only input head, an output tape, a one-way write-only output head, a finite number of counters, and a finite-state

control (Fig.5.3). This model can be regarded as 1MCA with an output tape.

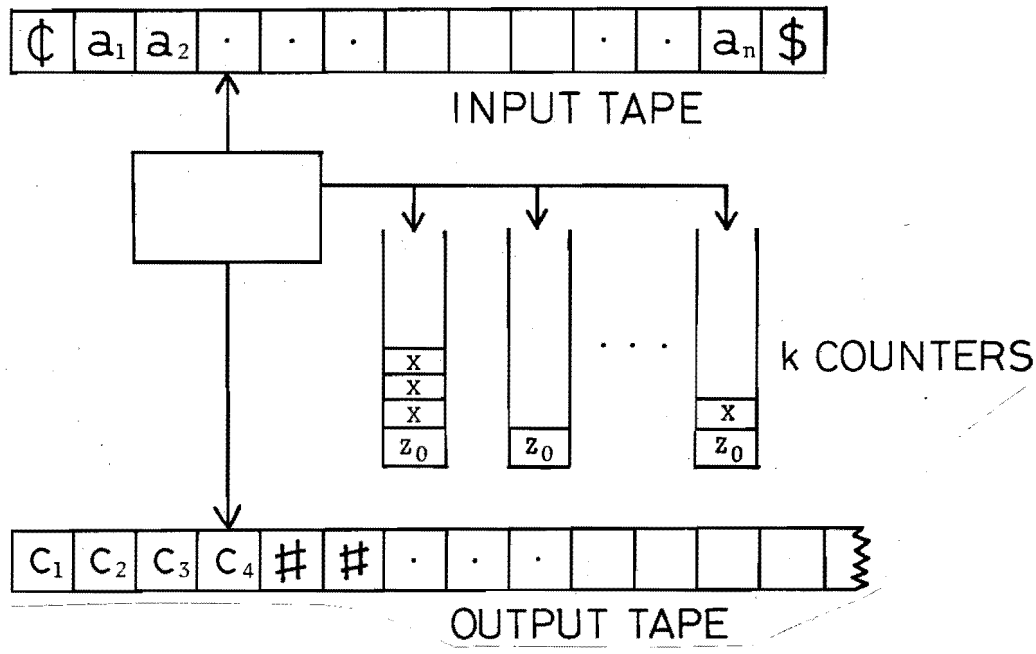

Fig. 5.3 A multi-counter transducer.

Formally a *deterministic two-way multi-counter transducer* (2WMCT) is an 11-tuple

$M = (K, \{a\}, \Gamma, \{c\}, \delta, k, q_0, ¢, \$, \#, F)$,

where K, {a}, {c}, q_0, ¢, $, #, and F are the same as in TT, Γ is a set of counter symbols ($\Gamma=\{x,z_0\}$, z_0 is a bottom symbol), k is the number of counters, and δ is a mapping from a subset of $K\times\{a,¢,\$\}\times\Gamma^k$ into $K\times\{c,\varepsilon\}\times\{L,R,H\}\times\{-1,0,+1\}^k$.

The class of deterministic two-way k-counter transducers is denoted by 2WMCT(k).

Similarly, a *deterministic one-way multi-counter transducer* can be also defined. The class of one-way k-counter transducers is denoted by 1WMCT(k).

5.3.3 Hierarchy

Here, we study the relations of computing abilities among 2WMHT(k), 1WMHT(k), 2WMCT(k), 1WMCT(k), etc.

First, it is shown that 2WMHT and 2WFST have tape complexity log n and c, respectively. This can be proved in a similar way as in the case of 2MHA or 1MHA (in chapter 3 or 4).

Theorem 5.12

$\mathcal{F}$[2WMHT] = $\mathcal{F}$[TT(log n)]

$\mathcal{F}$[2WFST] = $\mathcal{F}$[TT(c)] (c : constant)

Now, we investigate the relations of computing abilities among these transducers in more detail. The following theorem shows that computing ability of 2WMHT forms an infinite hierarchy.

Theorem 5.13

$\mathcal{F}$[2WMHT(k)] $\subsetneq$ $\mathcal{F}$[2WMHT(k+1)] (k=1,2,$\cdots$)

Proof. Consider the function $f(n) = n^{k+1}$. It is easily seen that $f \in \mathcal{F}$[2WMHT(k+1)]. Because we can construct $A \in$ 2WMHT(k+1) which computes f by using the k+1 heads to count up the number n^{k+1}.

Suppose some $B \in$ 2WMHT(k) computes f. And let s be the number of internal states of B. The total number of computational configurations of B for the input of length n is

$s \cdot n^k$. But, if B eventually halts, then the number of the output symbols is not greater than this number. Thus $s \cdot n^k > f(n) = n^{k+1}$ must hold for every n, but this is a contradiction. So, $f \notin \mathcal{F}[2WMHT(k)]$.

It is obvious that $\mathcal{F}[2WMHT(k)] \subseteq \mathcal{F}[2WMHT(k+1)]$. Thus $\mathcal{F}[2WMHT(k)] \subsetneq \mathcal{F}[2WMHT(k+1)]$ is concluded. (Q.E.D.)

Next, the relation between 1WFST and 2WFST is investigated. It is well known that one-way and two-way finite-state automata are equivalent in their accepting abilities (Rabin and Scott [40]). In the case of transducers, however, it is shown that 2WFST is strictly more powerful than 1WFST.

Theorem 5.14 $\mathcal{F}[1WFST] \subsetneq \mathcal{F}[2WFST]$

Proof. Consider the following function $f : \mathbf{N} \cup \{0\} \to \mathbf{N} \cup \{0\}$.

$$f(n) = \begin{cases} 0 & (n : \text{even}) \\ n & (n : \text{odd}) \end{cases}$$

It is easily seen that $f \in \mathcal{F}[2WFST]$. So, we will show $f \notin \mathcal{F}[1WFST]$.

Suppose there exists some $M \in 1WFST$ which computes f. Now, we consider the case that the input 2m-1 is given to M. Until M reads the right border symbol, the movements of M must be the same as the ones when the input 2m is given. Thus, until this moment, M does not write the output symbol. At the next step, however, M reads the right border symbol,

and knows that the length of the input is odd. So, M must write the output of length 2m-1, staying at this position. But 2m-1 may be greater than the number of internal states of M, so M cannot do this. Thus, $f \notin \mathcal{F}$[1WFST].

Since $\mathcal{F}$[1WFST] $\subseteq$ $\mathcal{F}$[2WFST] is clear, $\mathcal{F}$[1WFST] $\subsetneq$ $\mathcal{F}$[2WFST] is concluded. (Q.E.D.)

Now, we define a deterministic two-scan finite-state transducer as a variant of 2WFST.

Definition. A *deterministic two-scan finite-state transducer* (2SFST) is a finite-state transducer which scans the input tape only twice. In the first scan, the input head moves from the left border symbol to the right border symbol in one way. And in the second scan, it also moves from right to left in one way.

The following theorem states that 2WFST can be simulated by 2SFST, i.e. 2WFST need not scan the input more than twice.

Theorem 5.15 $\mathcal{F}$[2WFST] = $\mathcal{F}$[2SFST]

Proof. Since $\mathcal{F}$[2WFST] $\supseteq$ $\mathcal{F}$[2SFST] is obvious, we only prove $\mathcal{F}$[2WFST] $\subseteq$ $\mathcal{F}$[2SFST].

Let M be an arbitrary 2WFST. As a preliminary, we first construct M' $\in$ 2WFST which computes the same function as M and acts as follows. If the input head of M' once leaves one side

end of the input, then it moves only in one way until it reaches another side end.

Let s be the number of internal states of M. From Theorem 5.6, we may assume that M always halts for any input. First M' scans the input and examines if $n \leq s$, where n is the length of the input. If $n \leq s$, then M' remembers the number n, and simulates all the movements of M in the finite-state control of M', and writes the same number of output symbols as M. In this case, M' scans the input tape only once. So, in what follows, we assume $n > s$.

Now, M' begins to simulate M step by step. If the input head of M leaves one side end of the input, M' first simulates M until the input head of M goes s+1 squares away from the border symbol or it returns to the same side end or M halts. M' does this without moving the input head. (Of course, M' writes the same number of output symbols as M does.) This can be done by counting the position of the input head in the finite-state control. If the input head of M goes more than s squares away, it will reach another side end without returning the same side end. Thus M' now makes the input head actually go ahead by s+1 squares, and then proceeds the further simulation of M. In this process, the input head of M may temporarily go backward, but it is at most s squares. So, remembering the relative position in the finite-state control, M' can simulate it without moving the input head until the input head of M returns to this position.

By above, M' can simulate the movements of M since M leaves one side end until M next reaches the same or another side end. M' repeats this procedure until M halts, and M' can compute the same function as M. Note that M' always halts at the border symbol.

Next, we construct M"$\in$2SFST which simulates M'. Let $\{q_0,q_1,\cdots,q_r\}$ be the set of internal states of M', where q_0 is the initial state. Let α_n be the mapping from $\{q_0,\cdots,q_r\}\times\{L,R\}$ into $\{q_0,\cdots,q_r\}\times\{L,R\}\cup\{-\}$, which is defined by M' and n (n is the length of the input). $\alpha_n(q_i,L) = (q_j,L)$ $(0\leq i,j\leq r)$ means that if M' starts from the left border symbol in the state q_i, M' immediately becomes the state q_j without moving the input head. $\alpha_n(q_i,L) = (q_j,R)$ means that if M' starts from the same situation, then M' reaches the right border symbol in the state q_j. Note that, in this case, M' must move to the right in its first step. And $\alpha_n(q_i,L) = -$ means that M' immediately halts. $\alpha_n(q_i,R)$ is similar to these except that M' starts from the right border symbol.

If the input of length n is given, M" can determine the mapping α_n by scanning the input only once. Since M' does not go backward in the middle of the input, M" can simulate all the states of M' simultaneously during the first scan (M" does not write the output symbol in the first scan). And M" remembers the mapping α_n in the finite-state control (α_n can be described as a finite table). From the mapping α_n, M" can uniquely determine the sequence

(p_1,X_1), (p_2,X_2), $\cdots$, (p_m,X_m)

$(p_k \in \{q_0,\cdots,q_r\}$, $X_k \in \{L,R\}$, $(1 \leq k \leq m))$

which satisfies the following conditions.

$(p_1,X_1) = (q_0,L)$

$\alpha_n(p_k,X_k) = (p_{k+1},X_{k+1})$ $\quad (1 \leq k \leq m-1)$

$\alpha_n(p_m,X_m) = -$

This sequence is the whole history of M' at the side ends of the input. And thus, the whole movements of M' can be partitioned into m-1 segments. Since M' never loops, $m \leq 2(r+1)$ holds. So, M" can remember this history in the finite-state control.

In the second scan, M" simulates all the movements of M'. M" can do this by simulating m-1 segments of M' simultaneously, each of which is the sequence of movements from the situation (p_k,X_k) to the situation (p_{k+1},X_{k+1}) $(1 \leq k \leq m-1)$. And at each step, M" writes the total number of output symbols. By this, M" can compute the same function as M'. (Q.E.D.)

Next, we investigate the computing ability of 1WMHT. It was shown, in Theorem 5.13, that the computing ability of 2WMHT forms an infinite hierarchy. However, it will be seen that 1WMHT(k) $(k \geq 2)$ is equivalent to 2SFST. To prove this, we now introduce a quasi-deterministic one-way multi-head transducer.

Definition. A nondeterministic one-way multi-head transducer M is called a *quasi-deterministic one-way multi-*

head transducer (QlWMHT), if M satisfies the following conditions.

(1) M nondeterministically chooses its behaviour only at the first step (i.e. at the initial state).

(2) There exists at most one choice which leads M to a final state, for any input.

Owing to the condition (2), QlWMHT also defines a function which maps a subset of $\mathbf{N} \cup \{0\}$ into $\mathbf{N} \cup \{0\}$.

The class of quasi-deterministic one-way k-head transducers is denoted by QlWMHT(k).

The following theorem shows that QlWMHT(k+1) is simulated by QlWMHT(k). Thus, there exists no hierarchy among QlWMHT.

Theorem 5.16 $\mathcal{F}[\text{QlWMHT}(k)] = \mathcal{F}[\text{QlWMHT}(k+1)]$

Proof. $\mathcal{F}[\text{QlWMHT}(k)] \subseteq \mathcal{F}[\text{QlWMHT}(k+1)]$ is obvious, so we only show $\mathcal{F}[\text{QlWMHT}(k)] \supseteq \mathcal{F}[\text{QlWMHT}(k+1)]$.

Let M be an arbitrary QlWMHT(k+1) with s internal states, and let $h_1, \cdots, h_{k+1}$ be the k+1 input heads of M. Suppose an input of length n is given to M. We may assume, without loss of generality, that M always halts after shifting all the input heads to the right border symbol, for any n. Now, we consider the period between the moment M starts to move and the moment some input head of M first reaches the right border symbol. This period is called the *first stage* of M. If n>s, the first stage is further partitioned into three

phases, i.e. the transient phase, the cyclic phase, and the closing phase. This is because the input is contentless, and thus M moves autonomously. M first goes through the transient phase, then, (in the cyclic phase) M repeats some sequence of movements (called the basic cycle), and finally, (in the closing phase) some initial segment of the basic cycle closes the first stage of M. However, since M may behave nondeterministically at the first step, the movements of M in the first stage cannot be determined uniquely. But, now we suppose that we have been able to know the choice M had made. Let m be the total number of possible choices M can make at the first step. And we assume that M had made the ℓ-th choice ($1 \leq \ell \leq m$). Then we can determine the movements of M in the first stage from the transition diagram of M.

The transient phase is a sequence

$$T^{\ell} = \{(p^{\ell}(1), \alpha_1^{\ell}(1), \cdots, \alpha_{k+1}^{\ell}(1)), \cdots \\ \cdots, (p^{\ell}(t), \alpha_1^{\ell}(t), \cdots, \alpha_{k+1}^{\ell}(t))\},$$

where $p^{\ell}(i)$ is an internal state of M at the i-th step, $\alpha_j^{\ell}(i)$ (= 0 or 1) is the shift direction of the j-th head at the i-th step ($\alpha_j^{\ell}(i)=1$ means the right-shift and $\alpha_j^{\ell}(i)=0$ means the non-shift), and $p^{\ell}(i) \neq p^{\ell}(i')$ holds if $i \neq i'$. The cyclic phase is a repetition of the following basic cycle,

$$B^{\ell} = \{(q^{\ell}(1), \beta_1^{\ell}(1), \cdots, \beta_{k+1}^{\ell}(1)), \\ \cdots, (q^{\ell}(u), \beta_1^{\ell}(u), \cdots, \beta_{k+1}^{\ell}(u))\},$$

where $q^{\ell}(i)$ is an internal state of M, $\beta_j^{\ell}(i)$ is the shift direction, $q^{\ell}(i) \neq q^{\ell}(i')$ if $i \neq i'$, and $q^{\ell}(i)$ differs every $p^{\ell}(i'')$ in T^{ℓ}. Assume M repeats the basic cycle B^{ℓ}, r times.

The closing phase is an initial segment of B^{ℓ} (it may be an empty sequence),

$$C^{\ell} = \{(q^{\ell}(1),\beta_1^{\ell}(1),\cdots,\beta_{k+1}^{\ell}(1)),$$
$$\cdots, (q^{\ell}(u'),\beta_1^{\ell}(u'),\cdots,\beta_{k+1}^{\ell}(u'))\},$$

where $0 \leq u' < u$. Note that T^{ℓ} and B^{ℓ} are uniquely determined from the transition diagram of M if ℓ is given. But C^{ℓ} (namely u') cannot be determined until M completes the first stage for the given input n.

Now, let x_j^{ℓ}, y_j^{ℓ}, z_j^{ℓ}, and v_j^{ℓ} be the numbers defined as follows ($1 \leq j \leq k+1$).

$$x_j^{\ell} = \sum_{i=1}^{t} \alpha_j^{\ell}(i)$$
$$y_j^{\ell} = \sum_{i=1}^{u} \beta_j^{\ell}(i)$$
$$z_j^{\ell} = \sum_{i=1}^{u'} \beta_j^{\ell}(i)$$
$$v_j^{\ell} = x_j^{\ell} + r \cdot y_j^{\ell} + z_j^{\ell}$$

When M becomes the final configuration of the first stage, the j-th input head is at the $(v_j^{\ell}+1)$-th square (Note that $(v_j^{\ell}+1)=n+1$ means that h_j is reading the right border symbol). If the j_0-th head reaches the right border symbol first of all, then $v_{j_0}^{\ell}=n$ and $v_j^{\ell} \leq n$ for every $j \neq j_0$.

Next, we construct $M' \in$ Q1WMHT(k) which computes the same function as M. Let $h_1', \cdots, h_k'$ be the k heads of M'. Since the number of heads of M is k+1, M' begins to simulate M from the final configuration of the first stage of M. To do this, M' nondeterministically guesses the answer of the

following questions, at its first step.

(1) Is n greater than $n_0 = s \times \max_{\ell,j} x_j^{\ell}$ $(1 \leq \ell \leq m,\ 1 \leq j \leq k+1)$?
(Here, if M' has conjectured $n \leq n_0$, M' then guesses (2). If M' has conjectured $n > n_0$, M' then guesses (3) and (4).)

(2) How long is the input tape? (Thus, M' guesses the length n from $\{0,\cdots,n_0\}$.)

(3) Which behaviour does M choose nondeterministically, at the first step? (Thus M' guesses the integer ℓ from $\{1,\cdots,m\}$.)

(4) What is the closing phase of the first stage of M? (Thus M' guesses the integer u' from $\{0,\cdots,u-1\}$. Note that u is uniquely determined from the integer ℓ that M' has conjectured.)

It is easily seen that M' has only finite choices for each of these questions.

First, we consider the case that M' has conjectured $n \leq n_0$. Let f be the function computed by M. M' remembers the finite table of f(n) for $0 \leq n \leq n_0$ in the finite-state control beforehand. M' first scans the input tape with the head h_1', and checks whether the conjectured number n is correct or not. If it is correct and f is defined for n, then M' writes the output f(n) by this table and halts in a final state. If otherwise, M' halts in a state other than the final states.

Next, in the case that M' has conjectured $n > n_0$, M' simulates M in the following manner. M' begins to simulate

M step by step from the moment when the first stage of M has finished. Let $J^{\ell}=\{j \mid y_j^{\ell} \geq y_i^{\ell}$, for all $i\}$. If the conjectures in (1), (3) and (4) are true, $H^{\ell}=\{h_j \mid j \in J^{\ell}$, and $x_j^{\ell}+z_j^{\ell} \geq x_i^{\ell}+z_i^{\ell}$ for all $i \in J^{\ell}\}$ is the set of heads which first reach the right border symbol, because $n>n_0$. Let h_{j_0} be the element of H^{ℓ}, which has the smallest suffix. (h_{j_0} can be immediately determined by M' from these conjectures.) The j-th head h_j ($1 \leq j \leq k+1$) of M (except h_{j_0}) is simulated by h_j' (if $j<j_0$) or h_{j-1}' (if $j>j_0$). At the final computational configuration of the first stage of M, h_j is at the position $v_j^{\ell}+1$. Thus, in the whole simulation, h_j' (or h_{j-1}') goes the distance n while h_j goes the distance $d_j^{\ell}=n-v_j^{\ell}$. Since h_{j_0} is the head which first reaches the right border symbol,

$$x_{j_0}^{\ell} + r \cdot y_{j_0}^{\ell} + z_{j_0}^{\ell} = n.$$

From this equation and

$$d_j^{\ell} = n - (x_j^{\ell} + r \cdot y_j^{\ell} + z_j^{\ell}),$$

we can obtain

$$n = a_j \cdot d_j^{\ell} + b_j,$$

where

$$a_j = \frac{y_{j_0}^{\ell}}{y_{j_0}^{\ell} - y_j^{\ell}}$$

$$b_j = \frac{y_{j_0}^{\ell}}{y_{j_0}^{\ell} - y_j^{\ell}} \cdot (x_j^{\ell} + z_j^{\ell}) - \frac{y_j^{\ell}}{y_{j_0}^{\ell} - y_j^{\ell}} \cdot (x_{j_0}^{\ell} + z_{j_0}^{\ell})$$

Note that x_j^{ℓ}, y_j^{ℓ}, z_j^{ℓ}, $x_{j_0}^{\ell}$, $y_{j_0}^{\ell}$, and $z_{j_0}^{\ell}$ can be uniquely and immediately determined from the conjectures in (3) and (4). Thus, when M shifts h_j one square to the right, then M'

simulates it by shifting h'_j (or h'_{j-1}) a_j squares to the right. (The constant b_j is previously adjusted by moving the head b_j squares to the right (if $b_j \geq 0$), or counting it in the finite-state control (if $b_j < 0$).) Although a_j and b_j may not be integers, M' can remember the fractional parts in the finite-state control, because a_j and b_j are rational numbers. By above, M' simulates the movements of M step by step, and writes the same number of output symbols as M. (If M halts in a state other than the final states, M' also does so.) At the same time, M' simulates the first stage of M. Since M moves autonomously in the first stage, this can be done by simulating the movements of h_{j_0} by h'_1. And M' also writes the same number of output symbols as M.

Now, M' must examine whether the conjectures in (1) and (4) are true. (The conjecture in (3) need not be examined, because M is originally quasi-deterministic.) The conjecture (1) can be examined by counting the length of the input with h'_1 (up to n_0). The conjecture in (4) can also be examined by simulating the movements of h_{j_0} (of the first stage) by h'_1. These examinations are also done simultaneously with the simulation of M. M' halts in a final state, if M halts in a final state and these conjectures are both true.

It is easily seen that there exists only one choice of movement which makes M' simulate M correctly. And if M' has made a wrong choice, M' does not halt in a final state. Thus M' computes the same function as M. (Q.E.D.)

The following lemmas show the relations among 2SFST, Q1WMHT(1), and 1WMHT(2).

Lemma 5.17 $\mathcal{F}$[Q1WMHT(1)] $\subseteq$ $\mathcal{F}$[2SFST]

Proof. Let M be an arbitrary Q1WMHT(1). In what follows, we construct M' $\in$ 2SFST which computes the same function as M. Let m be the total number of possible choices M can make at the first step. In the first scan, M' simulates all these m cases of movements of M, simultaneously. But, here, M' does not write the output symbols. At the end of the first scan, M' can know which choice is correct (i.e. which choice leads M to a final state). Thus, in the second scan, M' simulates only the correct choice, and writes the same number of output symbols as M. Clearly this M' computes the same function as M. (Q.E.D.)

Lemma 5.18 $\mathcal{F}$[2SFST] $\subseteq$ $\mathcal{F}$[1WMHT(2)]

Proof. It is easily proved from the facts that the first scan of 2SFST is simulated by the first head of 1WMHT(2), and the second scan is simulated by the second head.

(Q.E.D.)

From Theorem 5.16, Lemma 5.17 and 5.18, we can derive the following theorem.

Theorem 5.19

$\mathcal{F}$[2SFST] = $\mathcal{F}$[Q1WMHT(k)] (k=1,2,$\cdots$)

$\mathcal{F}$[2SFST] = $\mathcal{F}$[1WMHT(k')] (k'=2,3,$\cdots$)

Proof. $\mathcal{F}$[1WMHT(k)] $\subseteq$ $\mathcal{F}$[1WMHT(k+1)] and $\mathcal{F}$[1WMHT(k)] $\subseteq$ $\mathcal{F}$[Q1WMHT(k)] (k=1,2,$\cdots$) are obvious. So, from Theorem 5.16, Lemma 5.17 and 5.18, $\mathcal{F}$[2SFST] $\subseteq$ $\mathcal{F}$[1WMHT(2)] $\subseteq$ $\mathcal{F}$[1WMHT(3)] $\subseteq \cdots \subseteq$ $\mathcal{F}$[1WMHT(k)] $\subseteq$ $\mathcal{F}$[Q1WMHT(k)] = $\mathcal{F}$[Q1WMHT(k-1)] = $\cdots$ = $\mathcal{F}$[Q1WMHT(1)] $\subseteq$ $\mathcal{F}$[2SFST]. Thus, $\mathcal{F}$[2SFST] = $\mathcal{F}$[Q1WMHT(k)] (k=1,2,$\cdots$) and $\mathcal{F}$[2SFST] = $\mathcal{F}$[1WMHT(k')] (k'=2,3,$\cdots$) are concluded. (Q.E.D.)

Finally, the computing abilities of 2WMCT(k) and 1WMCT(k) are investigated.

As in the case of 1MCA (Theorem 4.9), it is easily seen that 2WMCT(2) and 1WMCT(2) are both universal.

Theorem 5.20 $\mathcal{F}$[2WMCT(k)] ($k \geq 2$) is the set of all the partial recursive functions†, and so is $\mathcal{F}$[1WMCT(k)].

The next theorem states that 1WMCT(1) and 2SFST are equivalent.

Theorem 5.21 $\mathcal{F}$[1WMCT(1)] = $\mathcal{F}$[2SFST]

---

† Of course, all the total recursive functions are included in this set.

Proof. First, $\mathcal{F}$[1WMCT(1)] $\subseteq$ $\mathcal{F}$[2SFST] is shown.

Let A_1 be an arbitrary 1WMCT(1) with s states, and let f be the function computed by A_1. We may assume, without loss of generality, that A_1 makes the counter empty when it halts. If A_1 halts or loops in the middle of the input tape of some length n_0, then so does it for any input of length $n>n_0$. Thus, $f(n)=f(n_0)$ holds for all $n>n_0$ (or, if $f(n_0)$ is undefined, f(n) is also undefined). Such a function can easily be computed by 2SFST, because it can be described by a finite table. So, we may assume that A_1 always reaches to the right border symbol for any n.

Now, let us give A_1 an input of length n. The computational configuration of A_1 is described by a triple $(p,x,y) \in K\times(\mathbf{N}\cup\{0\})\times\{1,2,\cdots,n,n+1\}$, where K is the set of internal states of A_1, x is the content of the counter, and y is the position of the input head. Let (p_t,x_t,y_t) be the computational configuration of A_1 at time t ($t=0,1,2,\cdots$). If $n>s^2$, then there exist integers i and j ($0\leq i<j\leq s^2$) which satisfy one of the following conditions.

(1) $p_i=p_j$, $0<x_i<x_j$, $x_t>0$ $(i<t<j)$, and $1\leq y_i<y_j<n+1$.

(2) $p_i=p_j$, $x_i=x_j$, and $1\leq y_i<y_j<n+1$.

Because, if there exists some integer ℓ ($0\leq\ell\leq s^2$) such that $x_\ell>s$, then there exist i and j that satisfy (1). Conversely, if $x_\ell\leq s$ for all ℓ ($0\leq\ell\leq s^2$), then there exist i and j that satisfy (2).

If the transition diagram of A_1 is given, we can determine beforehand whether (1) or (2) occurs, by simulating

the movements of A_1 up to s^2 steps.

Now, we construct $B_1 \in$ 2SFST which simulates A_1. Let m be the integer such that $y_{m-1}=n$ and $y_m=n+1$. In the first scan, B_1 simulates the movements of A_1 from the configuration $(p_1,x_1,y_1)=(q_0,0,1)$ to (p_m,x_m,y_m), and writes the output symbols as A_1 does. Note that, in both cases of (1) and (2), the sequence $(p_1,x_1,y_1),\cdots,(p_j,x_j,y_j)$ can be previously determined from the transition diagram of A_1. B_1 remembers this finite sequence and the numbers i and j, in the finite-state control as a table, and simulates A_1 according to this table. In the case of (2), the sequence of storage states $(q_i,x_i),\cdots,(q_j,x_j)$ repeatedly appears for sufficiently large n. Thus, at the end of the first scan, B_1 can uniquely determine the configuration (p_m,x_m,y_m). In the case of (1), the content of the counter grows linearly to n, until A_1 reaches to the right border symbol. In this case, the sequence of states $p_i,\cdots,p_j$ repeatedly appears, so B_1 can determine the state p_m.

B_1 next simulates the movements of A_1 from the configuration (p_m,x_m,y_m) to the final configuration. (The input head of A_1 stays at the right border symbol during this period.) In the case of (2), if A_1 eventually halts, then the content of the counter does not grow more than 2s, because $x_m \leq s$. So, B_1 can simulate the whole movements of A_1 in the finite-state control, and writes the same number of output symbols as A_1. (Thus, in this case, B_1 scans the input head only once.) In the case of (1), B_1 simulates the movements of

A_1 of this period by scanning the input head once more. Let r and u be the quotient and the remainder of (m-i)/(j-i). Then the following equations hold.

$$x_m = r\cdot(x_j-x_i) + x_{i+u}$$
$$n = r\cdot(y_j-y_i) + y_{i+u}$$

From these equations, we can obtain

$$n = \frac{y_j-y_i}{x_j-x_i}\cdot x_m + y_{i+u} - \frac{y_j-y_i}{x_j-x_i}\cdot x_{i+u} .$$

Note that u can be determined in the first scan of B_1, so the constants x_i, x_j, x_{i+u}, y_i, y_j, y_{i+u} are known by B_1. If A_1 eventually halts, then the content of the counter gradually decreases. Thus, the counter can be simulated by the second scan of the input head of B_1. (Since the input head of A_1 does not move, B_1 need not simulate it.) This can be done in a similar way as in Theorem 5.16. Namely, if A_1 decreases the content of the counter by 1, then B_1 shifs the input head by $(y_j-y_i)/(x_j-x_i)$ squares. (Its fractional part is remembered in the finite-state control, and the constant $y_{i+u} - x_{i+u}\cdot(y_j-y_i)/(x_j-x_i)$ is adjusted first.) In the decreasing process, the content of the counter may temporarily increase. But it is at most s, so that it can be simulated in the finite-state control of B_1. B_1 writes the same number of output symbols as A_1, and halts in a final state if and only if A_1 halts in a final state.

Clearly, this B_1 computes the function f. So, $\mathcal{F}$[1WMCT(1)] $\subseteq$ $\mathcal{F}$[2SFST] is concluded.

Next, we show $\mathcal{F}[1WMCT(1)] \supseteq \mathcal{F}[2SFST]$. For any $B_2 \in$ 2SFST, we now construct $A_2 \in 1WMCT(1)$ which simulates B_2. If the input of length n is given, A_2 begins to simulate the first scan of B_2. A_2 does this by scanning the input head of A_2, and writes the output symbols as B_2 does. Simultaneously, A_2 counts the number of squares of the input tape using the counter. Thus when A_2 reaches to the right border symbol, the counter keeps the number n. Next, A_2 simulates the second scan of B_2 using the counter. Namely, if B_2 shifts the input head by one square, then A_2 decreases the content of the counter by 1. And A_2 also writes the output symbols as B_2 does.

Clearly, A_2 computes the same function as B_2. And $\mathcal{F}[1WMCT(1)] \supseteq \mathcal{F}[2SFST]$ is concluded. (Q.E.D.)

The following theorem shows that 2WMCT(1) and 2WMHT(2) are equivalent. In section 4.3, we proved $\mathcal{L}[RA^1(2)] = \mathcal{L}[\widehat{RA}^1(2)] = \mathcal{L}[1MCA^1(1)] = \mathcal{L}[1BCA^1(1)] = \mathcal{L}[1MHA^1(2)]$. Since 2WMCT(1) and 2WMHT(2) are the same as $1MCA^1(1)$ and $1MHA^1(2)$ except that they have output tapes, we can prove the relation $\mathcal{F}[2WMCT(1)] = \mathcal{F}[2WMHT(2)]$ in a very similar way as in the case of $1MCA^1(1)$ and $1MHA^1(1)$.

Theorem 5.22 $\mathcal{F}[2WMCT(1)] = \mathcal{F}[2WMHT(2)]$

Proof. $\mathcal{F}[2WMCT(1)] \subseteq \mathcal{F}[2WMHT(2)]$ can be proved in a similar manner as in Theorem 4.9. Let A_1 be an arbitrary

2WMCT(1) with s states. Then, $B_1 \in$ 2WMHT(2) which simulates A_1 is constructed as follows. If A_1 halts for a given input of length n, then the counter of A_1 does not counts the number greater than $s \cdot (n+2)$. Thus the counter can be simulated by one of the heads of B_1. And at each step of the simulation, B_1 writes the same number of output symbols as A_1 does. By this, B_1 can compute the same function as A_1.

$\mathcal{F}$[2WMCT(1)] $\supseteq$ $\mathcal{F}$[2WMHT(2)] is based on Lemma 4.14. Let B_2 be an arbitrary 2WMHT(2). In the same way as in this lemma, we can construct $B_2' \in$ 2WMHT(2) such that one of the input heads of B_2' never reads the right border symbol for any input, and computes the same function as B_2. Since we can easily construct $A_2 \in$ 2WMCT(1) which simulates B_2', $\mathcal{F}$[2WMCT(1)] $\supseteq$ $\mathcal{F}$[2WMHT(2)] is concluded. (Q.E.D.)

Fig.5.4 summerizes the hierarchy of computing abilities of multi-head, multi-counter, and finite-state transducers.

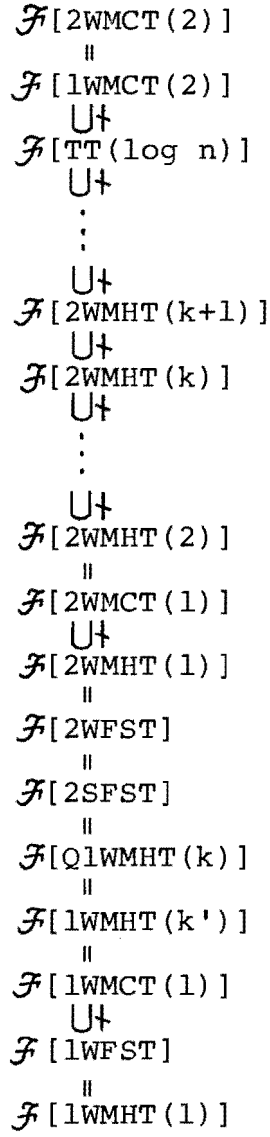

Fig. 5.4 The relations of computing abilities of multi-head, multi-counter, and finite-state transducers. (k=1,2,··· , k'=2,3,···)

5.4 Concluding Remarks

In this chapter, a Turing transducer and some other transducers were defined, and their computing abilities were investigated. Since the numbers of input symbols and output symbols are restricted to one, these models may seem to be particular ones. Indeed, many results in this chapter were derived from the fact that the input is contentless. But, they are very convenient models to investigate the computing abilities of number-theoretic functions.

As for TT(L(n)), the relation between the increasing degree of functions and the tape complexity was considered. Theorem 5.8 states that there exists no lower bound on slowness of increase in any class of tape complexity. It is easily seen that a similar result can be proved for 2WMHT(2), too.

In section 5.3, several relations among various kinds of transducers were derived. Since an acceptor (with a contentless input) is a special case of a transducer, the equivalent relations shown in Fig.5.4 also hold for the acceptors. But the relation " $\subsetneq$ " does not always hold for the acceptors. For example, the relation $\mathcal{L}$[1WFST] $\subsetneq$ $\mathcal{L}$[2WFST] forms a contrast with the fact that one-way and two-way finite-state acceptors are equivalent.

BIBLIOGRAPHY

[1] Aho, A.V.(ed.), "Currents in the Theory of Computing," Prentice-Hall, Englewood Cliffs, N.J. (1973)

[2] Beyer, W.T., "Recognition of topological invariants by iterative arrays," Ph.D. dissertation, MIT (1969)

[3] Blum, M., "A machine-independent theory of the complexity of recursive functions," J. Assoc. Comput. Mach., 14, 322-336 (1967)

[4] Blum, M., "On the size of machines," Inf. and Control, 11, 257-265 (1967)

[5] Blum, M. and Hewitt, C., "Automata on a two-dimensional tape," IEEE Symp. on Switching and Automata Theory, 155-160 (1967)

[6] Book, R.V., Greibach, S.A. and Wegbreit, B., "Time- and tape-bounded Turing acceptors and AFLs," J. Comput. and Syst. Sci., 4, 606-621 (1970)

[7] Cook, S.A., "A hierarchy for nondeterministic time complexity," J. Comput. and Syst. Sci., 7, 343-353 (1973)

[8] Ebi, H., Morita, K. and Sugata, K., "Computability of tape-bounded Turing transducer," Technical Report of IECE* Japan, AL77-59 (1977)

[9] Fischer, P.C., Meyer, A.R. and Rosenberg, A.L., " Counter machines and counter languages," Math. Syst. Theory, 2, 265-283 (1968)

[10] Freedman, A.R. and Ladner, R.E., "Space bounds for processing contentless inputs," J. Comput. and Syst. Sci.,

11 , 118-128 (1975)

[11] Glinert, E.P., "On restricted Turing computability," Math. Syst. Theory, 5, 331-343 (1971)

[12] Hartmanis, J. and Stearns, R.E., "On the computational complexity of algorithms," Trans. Am. Math. Soc. 117, 285-306 (1965)

[13] Hartmanis, J., "Computational complexity of one-tape Turing machine computations," J. Assoc. Comput. Mach., 15, 325-339 (1968)

[14] Hartmanis, J., "On nondeterminancy in simple computing devices," Acta Informatica, 1, 336-344 (1972)

[15] Hartmanis, J. and Berman, L., "A note on tape bounds for SLA language processing," Proc. of 16th Ann. Symp. on Foundations of Computer Science, 65-70 (1975)

[16] Hopcroft, J.E. and Ullman, J.D., "Relations between time and tape complexities," J. Assoc. Comput. Mach., 15 414-427 (1968)

[17] Hopcroft, J.E. and Ullman, J.D., "Some results on tape-bounded Turing machines," J. Assoc. Comput. Mach., 16, 168-177 (1969)

[18] Hopcroft, J.E. and Ullman, J.D., "Formal Languages and Their Relation to Automata," Addison-Wesley, Reading, Mass. (1969)

[19] Hsia, P. and Yeh, R.T., "Marker automata," Information Sciences, 8, 71-88 (1975)

[20] Ibarra, O.H., "Characterizations of some tape and time complexity classes of Turing machines in terms of multi-

head and auxiliary stack automata," J. Comput and Syst. Sci., 5, 88-117 (1971)

[21] Ibarra, O.H., "A note concerning nondeterministic tape complexities," J. Assoc. Comput. Mach., 19, 608-612 (1972)

[22] Ibarra, O.H., "On two-way multihead automata," J. Comput. and Syst. Sci., 7, 28-36 (1973)

[23] Ibarra, O.H. and Melson, R.T., "Some results concerning automata on two-dimensional tapes," Int. J. Comput. Math., 4-A, 269-279 (1974)

[24] Ibarra, O.H. and Sahni, S.K., "Hierarchies of Turing machines with restricted tape alphabet size," J. Comput. and Syst. Sci., 11, 56-67 (1975)

[25] Ibarra, O.H. and Kim, C.E., "On 3-head versus 2-head finite automata," Acta Informatica, 4, 193-200 (1975)

[26] Inoue, K. and Nakamura, A., "Some properties on two-dimensional nondeterministic finite automata and parallel sequential array acceptors," Trans.IECE* Japan, 60-D, 990-997 (1977)

[27] Inoue, K. and Takanami, I., "A note on some closure properties of tape-bounded two-dimensional Turing machines," Technical Report of IECE* Japan, AL77-48 (1977)

[28] Lewis II, P.M., Stearns, R.E. and Hartmanis, J., "Memory bounds for recognition of context-free and context-sensitive languages," IEEE Conf. Rec. on Switching Circuit Theory and Logical Design, 191-202 (1965)

[29] Minsky, M.L., "Computation : Finite and Infinite Machines," Prentice-Hall, Englewood Cliffs, N.J. (1967)

[30] Monien, B., "Transformational methods and their application to complexity problems," Acta Informatica, 6, 95-108 (1976)

[31] Morita, K., Umeo, H., Sugata, K. and Hirao, M., "The computer simulation system for 2-dimensional tape automata and some pattern recognition problems," Technical Report of IECE* Japan, AL75-25 (1975)

[32] Morita, K., Umeo, H. and Sugata, K., "Some properties of n-bounded counter automaton and rebound automaton," Technical Report of IECE* Japan, AL76-23 (1976)

[33] Morita, K., Umeo, H. and Sugata, K., "Computational complexity of n-bounded counter automaton and multi-dimensional rebound automaton," Trans. IECE* Japan, 60-D, 283-290 (1977)

[34] Morita, K., Umeo, H. and Sugata, K., "Language recognition abilities of tape-bounded 2-dimensional Turing machine and parallel-sequential array automata," Technical Report of IECE* Japan, AL76-72 (1977)

[35] Morita, K., Umeo, H. and Sugata, K., "Computational complexity of L(m,n) tape-bounded two-dimensional tape Turing machines," Technical Report of IECE* Japan, AL77-8, Trans.IECE* Japan, 60-D, 982-989 (1977)

[36] Morita, K., Umeo, H. and Sugata, K., "Language recognition abilities of several two-dimensional tape automata and their relation to tape complexities," Trans. IECE* Japan, 60-D, 1077-1084 (1977)

[37] Morita, K., Umeo, H., Ebi, H. and Sugata, K., "Lower

bounds on tape complexity of two-dimensional tape Turing machines," Trans. IECE* Japan, 61-D, 381-386 (1978)

[38] Morita, K., Umeo, H. and Sugata, K., "Tape complexities of two-dimensional tape Turing machines," (in contributing)

[39] Parikh, R.J., "On context free languages," J. Assoc. Comput. Mach., 13, 570-581 (1966)

[40] Rabin, M.O. and Scott, D., "Finite automata and their decision problems," IBM J. Res. and Dev., 3, 114-125 (1959)

[41] Rado, T., "On non-computable functions," Bell Syst. Tech. J., 41, 877-884 (1962)

[42] Ritchie, R.W. and Springsteel, F.N., "Language recognition by marking automata," Inf. and Control, 20, 313-330 (1972)

[43] Rosenberg, A.L., "On multi-head finite automata," IBM J. Res. and Dev., 10, 388-394 (1966)

[44] Rosenfeld, A. and Milgram, D.L., "Parallel/sequential array automata," Inf. Proc. Letters, 2, 43-46 (1973)

[45] Savitch, W.J., "Relationship between nondeterministic and deterministic tape complexities," J. Comput. and Syst. Sci., 4, 177-192 (1970)

[46] Smith III, A.R., "Two-dimensional formal languages and pattern recognition by cellular automata," IEEE Symp. on Switching and Automata Theory, 144-152 (1971)

[47] Stearns, R.E., Hartmanis, J. and Lewis II, P.M., "Hierarchies of memory limited computations," IEEE

Conf. Rec. on Switching Circuit Theory and Logical Design, 179-190 (1965)

[48] Sudborough, I.H., "Bounded-reversal multihead finite automata languages," Inf. and Control, 25, 317-328 (1974)

[49] Sudborough, I.H., "On tape-bounded complexity classes and multihead finite automata," J. Comput. and Syst. Sci., 10, 62-76 (1975)

[50] Sugata, K., Umeo, H. and Morita, K., "On the computing abilities of rebound automaton," Technical Report of IECE* Japan, AL75-43 (1975)

[51] Sugata, K., Umeo, H. and Morita, K., "Infinite hierarchy of computing ability of rebound automata, n-bounded counter automata and marker automata," Technical Report of IECE* Japan, AL75-67 (1976)

[52] Sugata, K., Morita, K. and Umeo, H., "Relationship of computing ability between n-bounded multi-counter automaton and multi-track log n tape-bounded Turing machine," Technical Report of IECE* Japan, AL76-15 (1976)

[53] Sugata, K., Umeo, H. and Morita, K., "The language accepted by rebound automaton and its computing ability," Trans.IECE* Japan, 60-A, 367-374 (1977)

[54] Sugata, K., Morita, K. and Umeo, H., "The language accepted by n-bounded multi-counter automaton and its conputing ability," Trans. IECE* Japan, 60-D, 275-282 (1977)

[55] Sugata, K., Umeo, H. and Morita, K., "Computing ability of marker automata on two-dimensional connected tape,"

Technical Report of IECE* Japan, AL77-19 (1977)

[56] Taniguchi, K. and Kasami, T., "Some decision problems for two-dimensional nonwriting automata," Trans. IECE* Japan, 54-C, 578-585 (1971)

[57] Turing, A.M., "On computable numbers, with an application to the Entscheidungsproblem," Proc. London Math. Soc., ser.2, 42, 230-265 (1936)

[58] Umeo, H., Morita, K. and Sugata, K., "On the computing abilities of marker automata on a one-dimensional tape," Technical Report of IECE* Japan, AL75-7 (1975)

[59] Umeo, H., Morita, K. and Sugata, K., "Pattern recognition capabilities of automata on the 2-dimensional tape," Technical Report of IECE* Japan, AL75-10 (1975)

[60] Umeo, H., Morita, K. and Sugata, K., "A new concept of dimensionality of automata on the multi-dimensional tapes," Technical Report of IECE* Japan, AL76-8 (1976)

[61] Umeo, H., Morita, K. and Sugata, K., "Pattern recognition by automata on a two-dimensional tape," Trans. IECE* Japan, 59-D, 817-824 (1976)

[62] Umeo, H., Morita, K. and Sugata, K., "Some closure properties of the class of sets accepted by two-dimensional L(m,n) tape-bounded Turing machines," Technical Report of IECE* Japan, AL77-18 (1977)

[63] Umeo, H., Morita, K. and Sugata, K., "Some notes concerning the shape of two-dimensional input tapes," Trans. IECE* Japan, 60-D, 570-577 (1977)

*IECE : The Institute of Electronics and Communication Engineers of Japan