コオペレーティング・オートマタ・システム

(Cooperating Systems of Automata)

平成６年３月

王　　躍

山口大学大学院工学研究科

COOPERATING SYSTEMS OF AUTOMATA

by

YUE WANG

DEPARTMANT OF COMPUTER SCIENCE

AND SYSTEMS ENGINEERING

FACULTY OF ENGINEERING

YAMAGUCHI UNIVERSITY

To my Father,

this little thesis is respectfully dedicated.

ABSTRACT

This thesis is a study of cooperating systems of automata as (one-dimensional or two-dimensional) language acceptors. A cooperating system of automata consists of a finite number of automata (such as finite-state automata, counter machines, and so on) which work independently (in parallel) on the same input tape. Those automata whose input heads are on the same cell of the input tape can communicate with each other. Some investigations about the maze and labyrinth search problems for cooperating systems of finite-state automata have already been made. But there is no investigation of such models as language acceptors at all. It is worthwhile to study the properties of cooperating systems of automata as language acceptors, because we can consider these machines as one of simple models of parallel computation. We introduce in this thesis two types of cooperating systems of automata: the cooperating system of finite-state automata, and the cooperating system of counter machines, where the later may be considered as a natural extension of the former in computing power. The thesis consists of five chapters.

Chapter 1 gives a general introduction to some important (nonwriting) models of computation in the theory of automata and formal languages, as well as a historical overview. Among these models, some are closely related to our concerned models.

Chapter 2 studies the properties of cooperating systems of finite-state automata as string acceptors. It is shown that there exists an infinite hierarchy, based on the number of finite-state automata, of the classes of languages accepted by cooperating systems of finite-state automata. The differences between the computing powers of the cooperating systems of one-way deterministic finite-state automata and the cooperating systems of one-way nondeterministic finite-state automata, and between the computing powers of the cooperating systems of one-way finite-state automata and the cooperating systems of two-way finite-state automata are explored. We also prove some results concerning how the reversal number restriction to the cooperating system of two-way finite-state automata affects its computing power. Finally the

closure properties of the classes of languages accepted by cooperating systems of one-way finite-state automata under basic language operations are investigated.

Chapter 3 establishes a relationship of the cooperating systems of finite-state automata to other kinds of automata. One of main results is that one-way (one-) counter machines and cooperating systems of two one-way finite-state automata are equivalent in computing power. From this fact, several results on the decision problems associated with the cooperating systems of finite-state automata are obtained.

Chapter 4 turns to study the properties of cooperating systems of two-dimensional finite-state automata as two-dimensional language acceptors. Many results similar to those for the one-dimensional case are proved. For example, it is shown that hierarchies of the classes of languages accepted by cooperating systems of two-dimensional finite-state automata can be obtained by varying the number of finite-state automata in both the four-way case and the three-way case (corresponding to the two-way and one-way cases in one-dimension, respectively). The differences between the computing powers of the cooperating systems of three-way two-dimensional finite-state automata and the cooperating systems of four-way finite-state automata, and between the computing powers of the cooperating systems of three-way two-dimensional deterministic finite-state automata and the cooperating systems of three-way two-dimensional nondeterministic finite-state automata can be also shown. On the other hand, some results on cooperating systems of three-way two-dimensional finite-state automata are sharply contrast with the corresponding results in the one-dimensional case, when the input tapes are restricted to square ones. For example, in the one-dimensional case, the cooperating systems of one-way finite-state automata are less powerful than one-way sensing multihead finite automata, whereas in the two-dimensional case, the cooperating systems of three-way two-dimensional finite-state automata and three-way two-dimensional sensing multihead finite automata are equivalent in computing power when the input tapes are restricted to square ones. We also examine the closure properties of the classes of languages accepted by cooperating systems of three-way two-dimensional finite-state automata under some basic two-dimensional language operations.

Chapter 5 introduces the cooperating systems of counter machines, and investigates their properties as (one-dimensional) language acceptors. Our attention is directed to cooperating systems of polynomial time- (or space-) bounded counter machines. A relationship between the accepting powers of the cooperating systems of counter machines and multicounter machines with polynomial time or space bound is investigated. For the classes of languages accepted by cooperating systems of counter machines with polynomial time or space bound, a full hierarchy result based on the number of counter machines is proved for both the one-way case and the two-way case, and some hierarchy results based on the time or space allowed are obtained for the one-way case. We also show the differences between the computing powers of the cooperating systems of one-way counter machines and the cooperating systems of two-way counter machines, and between the computing powers of the cooperating systems of one-way deterministic counter machines and the cooperating systems of one-way nondeterministic counter machines, where all the automata are polynomial time bounded. Finally the closure properties of the classes of languages accepted by cooperating systems of one-way counter machines with polynomial time bound under basic language operations are investigated.

ACKNOWLEDGEMENTS

The work on this thesis has been made under the guidance of my thesis advisor, professor Katsushi Inoue. I wish to express my sincere gratitude to him for his contant encouragement and generous support.

I am greatly indebted to Professor Itsuo Takanami for his constant encouragement and valuable comments. I am also greatly indebted to Professors Toyoshi Torioka, Taiho Kanaoka, Takahiro Watanabe and Akira Ito for their helpful suggestions and hospitality.

I would like to express my deep thanks to Professor Kenichi Morita who has provided me with useful insights during the initial stage of this work.

Finally, the support recevied by the author from the staff and students in Inoue Laboratory is gratefully acknowledge.

GLOSSARY

Symbol	Interpretation
$\emptyset$	empty set
∞	infinity
$\in$	is an element of
$\notin$	is not an element of
$\subseteq$	is a subset of
$\subsetneq$	is a proper subset of
$\cup$	union
$\cap$	intersection
$\times$	Cartesian product
$\lfloor x \rfloor$ ($[x]$)	greatest integer less than or equal to x
$\lceil x \rceil$	least integer greater than or equal to x
$f(n) = O(g(n))$	if there exist positive constants c and n_0 such that $f(n) \leq c \cdot g(n)$ for all $n > n_0$
$f(n) = \Omega(g(n))$	if there exist positive constants c and n_0 such that $c \cdot f(n) \geq g(n)$ for all $n > n_0$
2^Q	power set of a set Q
$\|Q\|$	cardinality of a set Q
$\overline{X}$	complement of a set X
XY	concatenation of sets X and Y
Σ^*	set of strings over Σ
$\|w\|$	length of a string w
w^R	reversal of a string w
$\Sigma^{(2)}$	set of two-dimensional tapes over Σ

Symbol	Interpretation
T^R	rotation of a set of two-dimensional tapes T
T^{RR}	row reflection of a set of two-dimensional tapes T
T^{RC}	row cyclic closure of a set of two-dimensional tapes T
T^{CC}	column cyclic closure of a set of two-dimensional tapes T
$\mathcal{L}$[A]	language accpted by the automata A's
CS-FA	a cooperating system of finite-state automata
CS-FA(k)	a cooperating system of k (two-way nondeterministic) finite-state automata
CS-DFA(k)	a cooperating system of k (two-way) deterministic finite-state automata
CS-1FA(k)	a cooperating system of k one-way (nondeterministic) finite-state automata
CS-1DFA(k)	a cooperating system of k one-way deterministic finite-state automata
HA(k)	a two-way k-head (nondeterministic) finite automaton
DHA(k)	a two-way k-head deterministic finite automaton
SeHA(k)	a two-way k-head sensing (nondeterministic) finite automaton
SeDHA(k)	a two-way k-head sensing deterministic finite automaton
SHA(k)	a two-way k-head simple (nondeterministic) finite automaton
SDHA(k)	a two-way k-head simple deterministic finite automaton
SeSHA(k)	a two-way k-head sensing simple (nondeterministic) finite automaton
SeSDHA(k)	a two-way k-head sensing simple deterministic finite automaton
1HA(k)	a one-way k-head (nondeterministic) finite automaton
1DHA(k)	a one-way k-head deterministic finite automaton

Symbol	Interpretation
1SeHA(k)	a one-way k-head sensing (nondeterministic) finite automaton
1SeDHA(k)	a one-way k-head sensing deterministic finite automaton
1SHA(k)	a one-way k-head simple (nondeterministic) finite automaton
1SDHA(k)	a one-way k-head simple deterministic finite automaton
1SeSHA(k)	a one-way k-head sensing simple (nondeterministic) finite automaton
1SeSDHA(k)	a one-way k-head sensing simple deterministic finite automaton
MA(k)	a k-marker (nondeterministic) finite automaton
DMA(k)	a k-marker deterministic finite automaton
PA(k)	a two-way (nondeterministic) finite automaton with k processors
DPA(k)	a two-way deterministic finite automaton with k processors
1PA(k)	a one-way (nondeterministic) finite automaton with k processors
1DPA(k)	a one-way deterministic finite automaton with k processors
CM(k)	a two-way (nondeterministic) k-counter machine
DCM(k)	a two-way deterministic k-counter machine
1CM(k)	a one-way (nondeterministic) k-counter machine
1DCM(k)	a one-way deterministic k-counter machine
CM(k)-Space($S(n)$)	a two-way (nondeterministic) k-counter machine which accepts in space $S(n)$
DCM(k)-Space($S(n)$)	a two-way deterministic k-counter machine which accepts in space $S(n)$
1CM(k)-Space($S(n)$)	a one-way (nondeterministic) k-counter machine which accepts in space $S(n)$
1DCM(k)-Space($S(n)$)	a one-way deterministic k-counter machine which accepts in space $S(n)$

Symbol	Interpretation
CM(k)-Time($T(n)$)	a two-way (nondeterministic) k-counter machine which accepts in time $T(n)$
DCM(k)-Time($T(n)$)	a two-way deterministic k-counter machine which accepts in time $T(n)$
1CM(k)-Time($T(n)$)	a one-way (nondeterministic) k-counter machine which accepts in time $T(n)$
1DCM(k)-Time($T(n)$)	a one-way deterministic k-counter machine which accepts in time $T(n)$
CS-2-FA(k)	a cooperating system of k (four-way) two-dimensional (nondeterministic) finite-state automata
CS-2-DFA(k)	a cooperating system of k (four-way) two-dimensional deterministic finite-state automata
CS-TR2-FA(k)	a cooperating system of k three-way two-dimensional (nondeterministic) finite-state automata
CS-TR2-DFA(k)	a cooperating system of k three-way two-dimensional deterministic finite-state automata
TR2-HA(k)	a three-way two-dimensional k-head (nondeterministic) finite automaton
TR2-DHA(k)	a three-way two-dimensional k-head deterministic finite automaton
TR2-SeHA(k)	a three-way two-dimensional k-head sensing (nondeterministic) finite automaton
TR2-SeDHA(k)	a three-way two-dimensional k-head sensing deterministic finite automaton
TR2-SHA(k)	a three-way two-dimensional k-head simple (nondeterministic) finite automaton

Symbol	Interpretation
TR2-SDHA(k)	a three-way two-dimensional k-head simple deterministic finite automaton
TR2-SeSHA(k)	a three-way two-dimensional k-head sensing simple (nondeterministic) finite automaton
TR2-SeSDHA(k)	a three-way two-dimensional k-head sensing simple deterministic finite automaton
TR2-CM(k)^S-Time($L(m)$)	a three-way two-dimensional (nondeterministic) k-counter machine with $L(m)$ time bound whose input tapes are restricted to square ones
TR2-DCM(k)^S-Time($L(m)$)	a three-way two-dimensional deterministic k-counter machine with $L(m)$ time bound whose input tapes are restricted to square ones
TR2-CM(k)^S-Space($L(m)$)	a three-way two-dimensional (nondeterministic) k-counter machine with $L(m)$ space bound whose input tapes are restricted to square ones
TR2-DCM(k)^S-Space($L(m)$)	a three-way two-dimensional deterministic k-counter machine with $L(m)$ space bound whose input tapes are restricted to square ones
CS-CM	a cooperating system of counter machines
CS-CM(k)	a cooperating system of k (two-way nondeterministic) counter machines
CS-DCM(k)	a cooperating system of k (two-way) deterministic counter machines
CS-1CM(k)	a cooperating system of k one-way (nondeterministic) counter machines
CS-1DCM(k)	a cooperating system of k one-way deterministic counter machines

Symbol	Interpretation
CS-CM(k)[Time($T(n)$)]	a cooperating system of k (two-way nondeterministic) counter machines with time bound $T(n)$
CS-DCM(k)[Time($T(n)$)]	a cooperating system of k (two-way) deterministic counter machines with time bound $T(n)$
CS-1CM(k)[Time($T(n)$)]	a cooperating system of k one-way (nondeterministic) counter machines with time bound $T(n)$
CS-1DCM(k)[Time($T(n)$)]	a cooperating system of k one-way deterministic counter machines with time bound $T(n)$
CS-CM(k)[Space($S(n)$)]	a cooperating system of k (two-way nondeterministic) counter machines with space bound $S(n)$
CS-DCM(k)[Space($S(n)$)]	a cooperating system of k (two-way) deterministic counter machines with space bound $S(n)$
CS-1CM(k)[Space($S(n)$)]	a cooperating system of k one-way (nondeterministic) counter machines with space bound $S(n)$
CS-1DCM(k)[Space($S(n)$)]	a cooperating system of k one-way deterministic counter machines with space bound $S(n)$

CONTENTS

CHAPTER 1

Introduction

In computer science the theory of automata and the theory of formal languages play the important roles for understanding and exploiting basic concepts and mechanisms in computing and information processing, because they provide the fundamental ideas and models underlying computing for computer science.

Since Alan Turing introduced his famous Turing machine in 1936 to answer a fundamental problems of computer science—"What kind of logical work can we effectively perform?", that is, what kind of problems can be solved by computers (or effective procedures), the Turing machine has become a simple mathematical model of a computer or the formalization of an effective procedure. Despite its simplicity, the Turing machine models the computing capability of a general-purpose computer.

At the same time, many researchers made serious efforts to investigate another fundamental problems of computer science—"How complicated is it to perform a given logical work?" The concept of "computational complexity" is a formalization of such "difficulty of logical works". The theory of computational complexity is an attempt to show that certain problems cannot be solved efficiently by establishing lower bounds on their inherent computational difficulty. Except in a few special circumstances, it has been unable to demonstrate that particular problems are computationally difficult. For example, the famous P-versus-NP question still remains far beyond our present abilities. One way to make some progress on this is to limit the capabilities of the computational model, thereby limiting the class of potential algorithms. In this way it

has been possible to achieve some interesting results. It is hoped that these may lead the way to lower bounds for more powerful computational models.

A variety of other models of computation, of which some appear to much weaker than Turing machines, are introduced and showed to be equivalent to Turing machines in computing power. For example, Baker and Book [1] proved that every recursively enumerable language (which is defined by a Turing machine) can be recognized by a one-way two-pushdown machine that operates in such a way that in every computation each pushdown makes at most one reversal. Minsky [2] showed that a one-way deterministic two-counter machine can simulate an arbitrary Turing machine. Either pushdown machine or counter machine can be considered as the restricted Turing machine, but the study of these models can lead to a better understanding of computation and bring some new, useful techniques to the theory of automata and the theory of formal languages.

Another basic simplest model is the finite-state automaton, which was originally developed with neuron nets and switching circuits in mind [3]. Nevertheless, the theory of finite-state automaton has preserved from its origins a great diversity of aspects. From one point of view, it is a branch of mathematics connected with the algebraic theory of semigroups and associative algebras [4, 5]. From another point of view, it is a branch of algorithm design connected with string manipulation and sequence processing. In contrast to the Turing machine models, the finite-state automaton requires no work tape in its computation (i.e., is a nonwriting model). In the simulation of this model for the processing of a language, one need not establish and maintain the often costly list structures that are usually employed to simulate the work tape of an Turing machine model (writing model). The nonwriting model is, in a sense, more efficient than any writing model that will do the same job.

In view of the computational weakness of the model, from the above motivation, some of models which preserve the nonwriting character have been introduced and investigated. In the following we review some important models as the nonwriting extensions of finite-state automata.

Piatkowski [6] first introduced the definition of multihead finite automata in the early 60's, and soon after Rosenberg [7] investigated the closure properties of the languages defined by one-way multihead finite automata (called one-way multihead languages) under the Boolean (union, intersection, and complementation) and Kleene (concatenation, closure, and reversal) operations, the relationship of one-way multihead languages to the regular, context-free, and context-sensitive languages, and several decision problems associated with one-way multihead finite automata. Yao and Rivest [8] completely proved in 1978 that for one-way multihead finite automata, $k+1$ heads are better than k (which was observed by Rosenberg [7], but Floyd [9] pointed out that Rosenberg's informal proof was incomplete). The proof was by a technique often called "cutting" and "pasting" or "fooling". Hromkovic [10] used suitably modified cutting and pasting to prove some non-closure properties of one-way multihead languages for the deterministic case. The similar problems for multihead finite automata were also considered by Ibarra [11, 12], Inoue [13], and Duris [14] et al. for so-called simple multihead finite automata, that is, for such automata that only one head sees the input symbols, the other heads can detect only the endmarkers.

The importance of two-way multihead finite automata results from the fact that they define two central complexity classes (DLOG and NLOG) [15, 16]. Also some complexity problems such as P-versus-NP and LBA, can be reduced to simple-looking problems about two-way multihead finite automata [16, 17, 18], even for the automata accepting only one-letter alphabet languages [19]. Two-way multihead finite automata are much more powerful than one-way ones, and enough to allow application of two frequently used techniques for proving separation results: diagonalization and padding [20]. The full hierarchy result based on the number of heads for two-way multihead finite automata was proved by Monien [21]. Ibarra et al. [22] gave some characterizations of two-way multihead finite automata in terms of multihead reversal-bounded pushdown automata and restricted checking stack automata. Engelfriet [23] improved some results of Ibarra et al. [22] and showed, in particular, that deterministic two-way two-head finite automata are equivalent to deterministic two-way checking stack automata. The other

interesting investigations about multihead finite automata can be found in [24, 25].

The concept of augmenting markers (also called pebbles sometimes) to finite-state automata was first introduced by Blum and Hewitt [26] in two-dimensional finite-state automata. Their work is the first attempt at approach to the problem of pattern recognition by serial computer, within the framework of automata theory. They have shown several interesting results about this model. For example, in the two-dimensional case one-marker finite automata are more powerful than finite-state automata, whereas in the one-dimensional case both of them accept the same regular languages. It was also shown that a one-marker finite automaton can decide if a pattern is pathwise-connected. However, whether one marker is necessary for a finite-state automaton doing so is not yet known. Ritchie and Springsteel [27, 28] investigated recognition of context-free languages by (one-dimensional) marker finite automata. Hsia and Yeh [29] investigated some fundamental properties of marker finite automata and studied their relationships to other types of automata and languages. One of main results in [29] is the establishment of an infinite hierarchy of languages recognizable by deterministic, halting marker finite automata. From this result, one can give the full hierarchy result based on the number of markers for deterministic marker finite automata by using the technique presented in [30]. As far as we know, it is unknown whether an analogous result holds for nondeterministic marker finite automata. Wang, Inoue and Takanami [31] considered a version of this problem for a new class of machines called multihead marker finite automata, and showed that an additional marker can add power for both the deterministic and nondeterministic versions of the machine that has at least two input heads, even if the alphabet is restricted to a one-letter. The maze (or labyrinth) search problem for finite automata was first investigated by Budach [32] and Shah [33]. Blum and Kozen [34] showed that the search can be implemented by a (deterministic) two-marker finite automaton. This result is "optimal", because Hoffmann [35] has proved that one-marker finite automata cannot search all finite mazes. Szepietowski [36] showed that a 5-marker finite automaton can search every infinite or finite maze.

The model that is called the cooperating system of finite-state automata (CS-FA) and will

be studied as an acceptor of languages in this thesis, was first considered by Blum and Sakoda [37] concerning the problem about searching a two- or three-dimensional obstructed space. The search algorithms for two-dimensional space are particularly interesting in view of the difficulty of searching more general graphs. It was shown in [37] that no CS-FA is capable of searching every finite three-dimensional maze. Rollik [38] and Hemmerling [39] investigated some other graph search problems on this model. Furthermore, Bull and Hemmerling [40, 41] essentially improved these results. However, it is little known what are the fundamental properties of CS-FA's as language acceptors. One of main purposes of this thesis is to answer this question.

Recently, Buda [42] introduced the notion of multiprocessor finite automata as one of the simplest models of parallel computation, and some interesting properties of this model have been investigated by Kakugawa et al. [43]. In view of parallel computation, it is interesting to compare this model with the cooperating system of finite-state automata for understanding the effects of data routing on parallel computation, since the later can be considered as another of the simplest models of parallel computation.

In this thesis we also introduce the cooperating system of counter machines (CS-CM) as a new type of language recognizers and investigate its basic properties. This model may be considered as a natural extension of the model CS-FA to counter machines.

Given a class of automata, the following problems are usually investigated:

(1) are there (finite or infinite) hierarchies in the classes of languages defined by the automata?

(2) are nondeterministic automata better than deterministic ones?

(3) what are the closure properties of the languages defined by the automata?

(4) the relationships to other types of automata and languages.

(5) the decision problems associated with the automata.

These problems will be considered for both CS-FA's and CS-CM's in this thesis.

The thesis has 5 chapters in addition to this Introduction.

Chapter 2 consists of a detailed study of the cooperating systems of finite-state automata as string acceptors. We first give the formal definition of cooperating system of finite-state automata, and then turn to an investigation on the effect of the number of finite-state automata in the system upon its computing power. We prove a full hierarchy result for the one-way case, but only a weak hierarchy result for the two-way case, based on the number of finite-state automata. It is shown that for each $k \geq 1$, cooperating systems of $k+1$ one-way (deterministic or nondeterministic) finite-state automata are more powerful than cooperating systems of k one-way (deterministic or nondeterministic) finite-state automata, and for each $k \geq 3$, cooperating systems of $k+2$ two-way (deterministic or nondeterministic) finite-state automata are more powerful than cooperating systems of k two-way (deterministic or nondeterministic) finite-state automata (even for languages over a one-letter alphabet). We next investigate the differences between the computing powers of the cooperating systems of one-way deterministic finite-state automata and the cooperating systems of one-way nondeterministic finite-state automata, and between the computing powers of the cooperating systems of one-way finite-state automata and the cooperating systems of two-way finite-state automata. We show that there is a language accepted by a cooperating system of 2 one-way nondeterministic finite-state automata, but not by any cooperating system of one-way deterministic finite-state automata, and there is a language accepted by a cooperating system of 2 two-way deterministic finite-state automata, but not by any cooperating system of one-way nondeterministic finite-state automata. We also prove some results concerning how the reversal number restriction to the cooperating system of two-way finite-state automata affects its computing power. It is shown that there is a language accepted by a cooperating system of 2 two-way deterministic finite-state automata, but not by any cooperating system of two-way nondeterministic finite-state automata with reversal bound n^a, for any $0 < a < 1/3$. Finally we investigate the closure properties of the classes of languages accepted by cooperating systems of one-way finite-state automata under the Boolean and Kleene operations. In particular, it is shown that the class of languages accepted by cooperating systems of 2 one-way nondeterministic finite-state automata is a full abstract family languages.

In Chapter 3, we establish a relationship of the cooperating systems of finite-state automata to other kinds of automata—multihead finite automata, marker finite automata, multiprocessor automata, and multicounter machines, as well as recalling their definitions. We mainly concentrate our attention on the one-way case. In particular, it is shown that one-way deterministic (nondeterministic) one-counter machines and cooperating systems of 2 one-way deterministic (nondeterministic) finite-state automata are equivalent in computing power. Several results concerning the decision problems associated with the cooperating systems of finite-state automata are obtained as the corollaries of this fact. It is shown that for any $k \geq 2$, the containment problem is undecidable for cooperating systems of k one-way deterministic finite-state automata, and the equivalence and universe problems are undecidable for cooperating systems of k one-way nondeterministic finite-state automata.

In Chapter 4, we study some properties of the cooperating systems of two-dimensional finite-state automata as two-dimensional language acceptors. A cooperating system of three-way two-dimensional finite-state automata is introduced as an extension of a cooperating system of one-way finite-state automata. We first show that cooperating systems of three-way two-dimensional (deterministic or nondeterministic) finite-state automata are less powerful than cooperating systems of four-way two-dimensional (deterministic or nondeterministic) finite-state automata, and cooperating systems of three-way two-dimensional nondeterministic finite-state automata are more powerful than cooperating systems of three-way two-dimensional deterministic finite-state automata, even if the input tapes are restricted to square ones. We then show that hierarchies of the classes of languages accepted by cooperating systems of two-dimensional finite-state automata can be obtained by varying the number of finite-state automata in both the four-way case and the three-way case. It is shown that for each $k \geq 1$, cooperating systems of $k + 1$ three-way two-dimensional (deterministic or nondeterministic) finite-state automata are more powerful than cooperating systems of k three-way two-dimensional (deterministic or nondeterministic) finite-state automata, and cooperating systems of $k + 2$ four-way two-dimensional (deterministic or nondeterministic) finite-state automata are more powerful than

cooperating systems of k four-way two-dimensional (deterministic or nondeterministic) finite-state automata (even for languages over a one-letter alphabet). We also compare the cooperating systems of three-way two-dimensional finite-state automata with other types of automata—two-dimensional (sensing) (simple) multihead finite automata, and two-dimensional multihead counter machines, when the input tapes are restricted to square ones. The results are sharply contrast with the corresponding results in the one-dimensional case obtained in Chapter 3. For example, it is shown (in Chapter 3) that for any $k, r \geq 2$, the class of languages accepted by cooperating systems of k one-way deterministic finite-state automata and the class of languages accepted by one-way simple (deterministic or nondeterministic) r-heads finite automata are incomparable, whereas it is shown (in Chapter 4) that the class of languages accepted by cooperating systems of three-way two-dimensional (deterministic or nondeterministic) finite-state automata is the same as the class of languages accepted by three-way two-dimensional simple (deterministic or nondeterministic) multihead finite automata. We finally examine the closure properties of the classes of languages accepted by cooperating systems of three-way two-dimensional finite-state automata with the square input tapes, under several two-dimensional language operations.

In Chapter 5, we introduce a new class of devices called cooperating systems of counter machines, which may be considered as an extended version of the cooperating system of finite-state automata where counter machines substitue for finite-state automata, and all questions about cooperating systems of finite-state automata as language acceptors are also investigated in this model. We first investigate a relationship between the accepting powers of cooperating systems of counter machines and multicounter machines with polynomial time or space bound. We then prove a full hierarchy result for both the one-way case and the two-way case, based on the number of counter machines. It is shown that for any $c, s, k \geq 1$, cooperating systems of $k+1$ one-way (deterministic or nondeterministic) counter machines with time (space) bound cn^s are more powerful than cooperating systems of k one-way (deterministic or nondeterministic) counter machines with time (space) bound cn^s, and cooperating systems of $k+1$ two-way (de-

terministic or nondeterministic) counter machines with space bound n are more powerful than cooperating systems of k two-way (deterministic or nondeterministic) counter machines with space bound n (even for languages over a one-letter alphabet). We also prove some hierarchy results for the one-way case, based on the time or space allowed. It is shown that for each $s \geq 1$ and each $k \geq 9$, cooperating systems of k one-way (deterministic or nondeterministic) counter machines with time bound n^{4s} are more powerful than cooperating systems of k one-way (deterministic or nondeterministic) counter machines with time bound n^s, and cooperating systems of k one-way (deterministic or nondeterministic) counter machines with space bound $n^{3(k-1)s+3}$ are more powerful than cooperating systems of k one-way (deterministic or nondeterministic) counter machines with space bound n^s. We next investigate the differences between the computing powers of the cooperating systems of one-way counter machines and the cooperating systems of two-way counter machines, and between the computing powers of the cooperating systems of one-way deterministic counter machines and the cooperating systems of one-way nondeterministic counter machines, where all the automata are polynomial time bounded. It is shown that there is a language accepted by a two-way deterministic one-counter machine with linear time bound, but not by any cooperating system of one-way nondeterministic counter machines with polynomial time bound, and there is a language accepted by a one-way nondeterministic one-counter machine in real time, but not by any cooperating system of one-way deterministic counter machines with polynomial time bound. Finally we investigate the closure properties of the classes of languages accepted by cooperating systems of one-way counter machines with polynomial time bound under the Boolean and Kleene operations.

In addition, we list some open problems in the end of every chapter (from Chapter 2) for suggesting some directions for further research.

CHAPTER 2

Cooperating Systems of Finite-state Automata as String Acceptors

In this chapter we first define the cooperating system of finite-state automata that can recognize given classes of strings or "one-dimensional tapes". After then, we turn to a detailed investigation of its properties as a string acceptor.

2.1 Definitions and Notation

A cooperating system of finite-state automata can be considered as one of the simplest models of parallel computation: there are more than one finite-state automata and an input tape where these finite-state automata operate simultaneously (in parallel) and can communicate with each other on the same cell of the input tape. More precisely, a cooperating system of k finite-state automata consists of k finite-state automata FA_1, FA_2, $\cdots$, FA_k, and a read-only input tape where these finite-state automata independently work step by step. Each step is assumed to require exactly one time for its completion. Those finite-state automata whose input heads scan the same cell of the input tape can communicate with each other, that is, every finite-state automaton is allowed to know the internal states of other finite-state automata on the cell it is scanning at the moment. The input tape holds a string of input symbols delimited by left and right endmarkers. The system starts with each FA_i on the left endmarker in its initial state and accepts the input tape if each FA_i enters an accepting state and halts when reading the right endmarker.

Formally, a cooperating system of k finite-state automata (CS-FA(k)) is denoted by

$$M = (\mathrm{FA}_1, \mathrm{FA}_2, \cdots, \mathrm{FA}_k),$$

where for each $1 \leq i \leq k$, FA_i is a finite-state automaton defined by a 9-tuple $(\Sigma, Q_i, X_i, \delta_i, q_{0_i}, F_i, ¢, \$, \phi)$ with

- Σ is a finite input alphabet ($¢, \$ \notin \Sigma$),
- Q_i is a finite set of states ($\phi \notin Q_i$),
- $X_i = (Q_1 \cup \{\phi\}) \times \cdots \times (Q_{i-1} \cup \{\phi\}) \times (Q_{i+1} \cup \{\phi\}) \times \cdots \times (Q_k \cup \{\phi\})$,
- δ_i is the transition function mapping $(\Sigma \cup \{¢, \$\}) \times X_i \times Q_i$ to $2^{Q_i \times \{-1,0,+1\}}$,
- q_{0_i} in Q_i is the initial state,
- $F_i \subseteq Q_i$ is the set of accepting states,
- $¢, \$$ are the left and right endmarkers, respectively.

An input to M is any string of the form $¢x\$$ where x is a string in Σ^*. The function of the endmarkers $¢$ and $\$$ is to let each automaton know when it is at the beginning or at the end of the input. At the start of the computation, every FA_i $(1 \leq i \leq k)$ is set to its initial state q_{0_i}, with its input head positioned on the left endmarker $¢$. A single move of M is described as follows: Let automata FA_1, FA_2, $\cdots$, FA_k be in states q_1, q_2, $\cdots$, q_k and scanning symbols a_1, a_2, $\cdots$, a_k (note that a_i may equal $¢$ or $\$$) at the moment t. If $\delta_i(a_i, (q'_1, \cdots, q'_{i-1}, q'_{i+1}, \cdots, q'_k), q_i) = \emptyset$, then FA_i has no next move (i.e., FA_i halts). If (p_i, d_i) is in $\delta_i(a_i, (q'_1, \cdots, q'_{i-1}, q'_{i+1}, \cdots, q'_k), q_i)$, then FA_i may enter state p_i, move right its input head d_i cells at the next moment $t+1$. Here for each $j \in \{1, \cdots, i-1, i+1, \cdots, k\}$

$$q'_j = \begin{cases} q_j \in Q_j & \text{if } \mathrm{FA}_i \text{ and } \mathrm{FA}_j \text{ are on the same cell at the moment } t; \\ \phi & \text{otherwise.} \end{cases}$$

The input $\not{c}x\$$ is accepted by M if there is a sequence of moves that leads every FA_i to an accepting state when scanning the right endmarker \$. We assume that no FA_i can fall off either end of the input. The language accepted by M is the set $T(M) = \{x \in \Sigma^* | \not{c}x\$ \text{ is accepted by } M\}$.

The CS-FA(k) M is called deterministic (denoted by CS-DFA(k)) if for each $1 \leq i \leq k$ and for each $(a_i, (q'_1, \cdots, q'_{i-1}, q'_{i+1}, \cdots, q'_k), q_i)$ in $(\Sigma \cup \{\not{c}, \$\}) \times X_i \times Q_i$,

$$|\delta_i(a_i, (q'_1, \cdots, q'_{i-1}, q'_{i+1}, \cdots, q'_k), q_i)| \leq 1,$$

that is, there exists only one possible sequence of moves for any input to M.

The CS-FA(k) (CS-DFA(k)) M is called one-way (denoted by CS-1FA(k) and CS-1DFA(k), respectively) if for each $1 \leq i \leq k$ and for each (p_i, d_i) in $\delta_i(a_i, (q'_1, \cdots, q'_{i-1}, q'_{i+1}, \cdots, q'_k), q_i)$, $d_i \neq -1$, that is, each FA_i in M may move its input head only to the right.

By $\mathcal{L}$[CS-FA(k)]= $\{T(M) | M \text{ is a CS-FA}(k)\}$, we denote the class of languages accepted by CS-FA(k)'s, and we will use the notation $\mathcal{L}[\,\cdot\,]$ in the same way throughout this thesis.

We say that the speed of a automaton, A, is $1/n$ if A moves its input head one cell every n steps.

2.2 Hierarchies Based on the Number of Finite-state Automata

In this section we investigate how the number of finite-state automata of the cooperating system affects its accepting power.

2.2.1 One-way Case

We begin with the following lemmas.

Lemma 2.1. Let $M = (A_1, A_2, \cdots, A_k)$ be a CS-1FA(k), $k \geq 1$. If x is any word in $T(M)$, then there exits a computation of M on x such that M accepts x at most in $O(|x|)$ steps.

Proof : For each $1 \leq i \leq k$, let Q_i denote the set of states of A_i. Consider a shortest accepting computation of M on x (in which no loop exists). It is obvious that during the com-

putation, at least one A_i moves its input head at least one cell to the right every $|Q_1||Q_2|\cdots|Q_k|$ steps. So M can accept x within $k|Q_1||Q_2|\cdots|Q_k||x|$ steps, and thus the lemma holds. □

Lemma 2.2. For each $k \geq 1$, let

$$T(k) = \{0^{m_1}10^{m_2}1\cdots10^{m_k}20^{m_1}10^{m_2}1\cdots10^{m_k} \in \{0,1,2\}^+ | \forall i(1 \leq i \leq k)[m_i \geq 1]\}.$$

Then

(1) $T(k) \in \mathcal{L}$[CS-1DFA$(k+1)$] and

(2) $T(k) \notin \mathcal{L}$[CS-1FA(k)].

Proof: (1) The language $T(k)$ is accepted by the CS-1DFA$(k+1)$ $M = (A_1, A_2, \cdots, A_k, A_{k+1})$ which acts as follows: Consider the case when an input word

$$\not{0}0^{m_1}10^{m_2}1\cdots10^{m_k}20^{m'_1}10^{m'_2}1\cdots10^{m'_k}\$$$

is presented to M. (Input words in the form different from the above can easily be rejected by M.)

(i) For each $1 \leq i \leq k$, A_i sweeps the subword 0^{m_i} at speed 1, the subword $0^{m'_i}$ at speed $1/3$, and the other parts at speed $1/2$.

(ii) A_{k+1} sweeps the input tape at the same speed $1/2$.

(iii) $A_1, A_2, \cdots, A_{k+1}$ can enter an accepting state when scanning the right endmarker if and only if for each $1 \leq i \leq k$, A_i and A_{k+1} scan the same cell just after A_{k+1} sweeps the subword $0^{m'_i}$. (See Fig. 2.1.)

Note that for each $1 \leq i \leq k$, $m_i = m'_i$ if and only if A_i and A_{k+1} scan the same cell just after A_{k+1} sweeps the subword $0^{m'_i}$. Thus M accepts the input word x if and only if $x \in T(k)$.

(2) Suppose that there is a CS-1FA(k) $M = (A_1, A_2, \cdots, A_k)$ accepting $T(k)$. For each $n \geq 1$, let

$$V(n) = \{0^{m_1}10^{m_2}1\cdots10^{m_k}20^{m_1}10^{m_2}1\cdots10^{m_k} | \forall i(1 \leq i \leq k)[1 \leq m_i \leq n]\}.$$

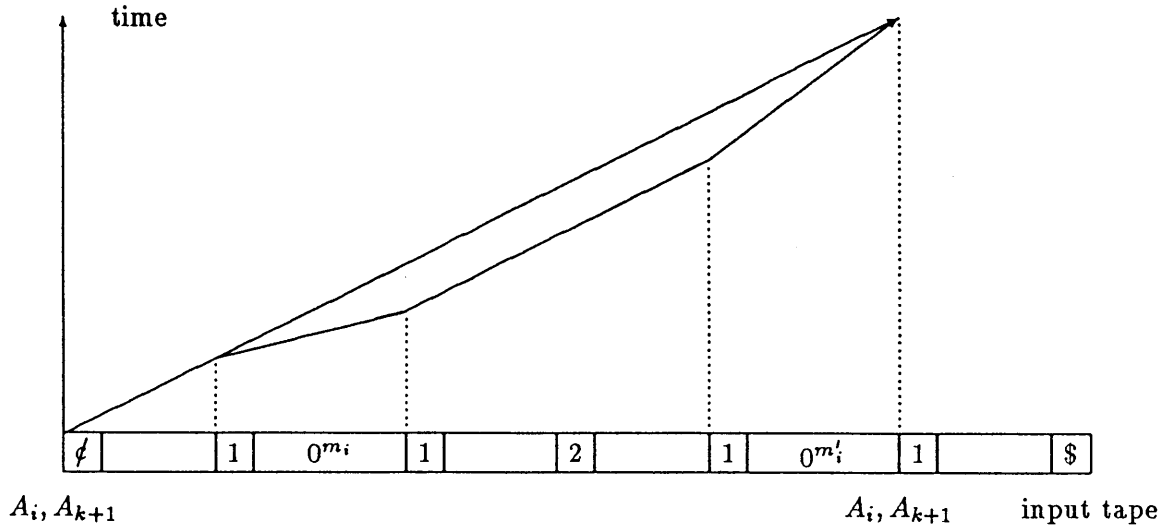

Fig. 2.1. Comparing m_i with m_i' by using A_i and A_{k+1}.

Clearly, each word w in $V(n)$ is in $T(k)$, so w is accepted by M. With each $w \in V(n)$, we associate one fixed accepting compuation, $c(w)$, of M on w. For each $1 \leq i \leq k$, let $t_i(w)$, $q_i(w)$ denote the time and the internal state, respectively, when A_i reads the symbol "2" of $w \in V(n)$ for the first time during the computation $c(w)$, and let $t_{\min}(w) = \min\{t_1(w), t_2(w), \cdots, t_k(w)\}$. Furthermore, for each $w \in V(n)$, let

$$u(w) = \langle (q_1(w), t_1(w) - t_{\min}(w)), (q_2(w), t_2(w) - t_{\min}(w)), \cdots, (q_k(w), t_k(w) - t_{\min}(w)) \rangle,$$

and for each $1 \leq i \leq k$, let

$$W_i(n) = \{w \in V(n) | t_i(w) = t_{\min}(w)\}.$$

Then, the following statement must hold :

Statement. For each $1 \leq i \leq k$ and any two different words w, w' in $W_i(n)$, $u(w) \neq u(w')$.

⌈ For otherwise suppose that

$$w = 0^{m_1}10^{m_2}1 \cdots 10^{m_k}20^{m_1}10^{m_2}1 \cdots 10^{m_k},$$

$$w' = 0^{m'_1}10^{m'_2}1\cdots10^{m'_k}20^{m'_1}10^{m'_2}1\cdots10^{m'_k},$$

$(m_1, m_2, \cdots, m_k) \neq (m'_1, m'_2, \cdots, m'_k)$, and $u(w) = u(w')$. Then, it follows that the word

$$w'' = 0^{m_1}10^{m_2}1\cdots10^{m_k}20^{m'_1}10^{m'_2}1\cdots10^{m'_k}$$

must be also accepted by M, because we can construct an accepting computation of M on w'' from $c(w)$ and $c(w')$, by the assumption $u(w) = u(w')$. This contradicts the fact that w'' is not in $T(k)$, since $(m_1, m_2, \cdots, m_k) \neq (m'_1, m'_2, \cdots, m'_k)$. ⌋

Clearly, for some $1 \leq j \leq k$,

$$|W_j(n)| \geq |V(n)|/k = n^k/k = \Omega(n^k).$$

Let $U_j(n) = \{u(w) | w \in W_j(n)\}$. Since for each $w \in V(n)$, $|w| = O(n)$, by Lemma 2.1, it follows that for each $i \in \{1, \cdots, j-1, j+1, \cdots, k\}$, $t_i(w) - t_{\min}(w) = O(n)$. Thus $|U_j(n)| = O(n^{k-1})$ (note that $t_j(w) = t_{\min}(w)$ for each $w \in W_j(n)$). Therefore, it follows that for large n, $|W_j(n)| > |U_j(n)|$, and thus there must exist two different words $w, w' \in W_j(n)$ such that $u(w) = u(w')$. This contradicts the above statement, and completes the proof. □

From Lemma 2.2, we can get the following theorem.

Theorem 2.1. For each $k \geq 1$,

(1) $\mathcal{L}$[CS-1FA(k)]$\subsetneq\mathcal{L}$[CS-1FA($k+1$)], and

(2) $\mathcal{L}$[CS-1DFA(k)]$\subsetneq\mathcal{L}$[CS-1DFA($k+1$)].

2.2.2 Two-way Case

Can we extend the result of Theorem 2.1 to the two-way case? That is, whether $\mathcal{L}$[CS-FA(k)] ($\mathcal{L}$[CS-DFA(k)]) $\subsetneq$ $\mathcal{L}$[CS-FA($k+1$)] ($\mathcal{L}$[CS-DFA($k+1$)]) for any $k \geq 2$? (Note that it is trivial to show $\mathcal{L}$[CS-FA(1)] ($\mathcal{L}$[CS-DFA(1)]) $\subsetneq$ $\mathcal{L}$[CS-FA(2)] ($\mathcal{L}$[CS-DFA(2)]).) Unfortunately, we were not able to solve this problem. However, we can give a relatively weak result about

hierarchies based on the number of finite-state automata. This result is obtained as a corollary of a hierarchy result on sensing two-way multihead finite automata [44].

A two-way k-head finite automaton (HA(k)) consists of a finite control and an input tape where k read-only heads may move independently in both directions. The input is placed between the left and right endmarkers. The automaton starts in a distinguished starting state with its k heads on the left endmarker. It accepts the input string if it stops in an accepting state. The automaton is called deterministic (denoted by DHA(k)) if its next move function is deterministic. (See Chapter 3 for the formal definition of multihead finite automaton.)

A sensing two-way k-head finite automaton (SeHA(k)) is the same device as a HA(k) except that the former can detect coincidence of the heads, that is, whose heads are allowed to sense the presence or absence of other heads on the same input position. A deterministic SeHA(k) is denoted by SeDHA(k).

The following fact (from Theorem 3.1 in [44]) is used.

Fact 2.1. For all $k \geq 3$,

(1) $\mathcal{L}$[SeHA(k)]$\subsetneq\mathcal{L}$[SeHA($k+1$)], and

(2) $\mathcal{L}$[SeDHA(k)]$\subsetneq\mathcal{L}$[SeDHA($k+1$)].

The above result, in fact, holds even though the alphabet is restricted to a one-letter.

Theorem 2.2. For each $k \geq 3$,

(1) $\mathcal{L}$[CS-FA(k)]$\subsetneq\mathcal{L}$[CS-FA($k+2$)], and

(2) $\mathcal{L}$[CS-DFA(k)]$\subsetneq\mathcal{L}$[CS-DFA($k+2$)].

Proof : It is easy to prove that every CS-FA(k) (CS-DFA(k)) can be simulated by a SeHA(k) (SeDHA(k)), and that every SeHA(k) (SeDHA(k)) can be simulated by a CS-FA($k+1$) (CS-DFA($k+1$)). From this observation and Fact 2.1, one can immediately derive the theorem. □

2.3 Determinism versus Nondeterminism

In this section, we investigate the difference between the accepting powers of CS-1FA(k)'s and CS-1DFA(k)'s. We give an example language which is acceptable by a CS-1FA(2), but not by any CS-1DFA(k).

Lemma 2.3. Let M be any CS-1DFA(k), $k \geq 1$. Then there exists a CS-1DFA M' such that

(1) $T(M') = T(M)$,

(2) for any input $\not x\$$, during the computation of M' on $\not x\$$, at least one finite-state automaton moves its input heads one cell to the right every c steps (except the finite-state automata whose input heads have reached the right endmarker \$), where c is some canstant dependent only on M, and

(3) given any input $\not x\$$ to M', all the finite-state automata eventually halt on the right endmarker \$.

Proof: Let $M = (A_1, A_2, \cdots, A_k)$ be a CS-1DFA(k) with $A_i = (\Sigma, Q_i, X_i, \delta_i, q_{0_i}, F_i, \not, \$, \phi)$ for $1 \leq i \leq k$. Let $c = \max_{1 \leq i \leq k}\{|Q_i| \times |X_i|\}$. From M, we construct a CS-1DFA(k) $M' = (A'_1, A'_2, \cdots, A'_k)$ which acts as follows: Given an input $\not x\$$, M' simulates the action of M on $\not x\$$ by making A'_i simulate each move of A_i and simultaneously remembering in its finite control whether it is the nearest one to the left endmarker $\not$ (i.e., whether it lags behind the others), for all $1 \leq i \leq k$. (Note that if there are more than two A'_i's that are the nearest ones to $\not$, then we refer to the finite-state automaton with the least index as the one that lags behind the others.) If A'_i lags behind the others and stays on the same cell more than $|Q_i| \times |X_i|$ steps (this means that A_i enters a infinite loop, so that the input can never be accepted by M), then A'_i moves to the right and makes all the finite-state automata halt without accepting the input. If each of A_1, A_2, $\cdots$, A_k eventually enters an accepting state when scanning the right endmarker \$, then M' accepts the input and halts. If A_1, A_2, $\cdots$, A_k reach the right endmarker

\$ and some A_i cannot enter an accepting state within $|Q_1||Q_2|\cdots|Q_k|$ steps (this means that A_i never enters an accepting state), then M' halts without accepting the input.

It is straightforward that M' satisfies the desired properties in the lemma. □

Lemma 2.4. Let $L_1 = \{w_1 1 w_2 | w_1, w_2 \in \{0,1\}^* \ \& \ |w_1| = |w_2|\}$. Then

(1) $L_1 \in \mathcal{L}$[CS-1FA(2)] and

(2) $L_1 \notin \bigcup_{1 \leq k < \infty} \mathcal{L}$[CS-1DFA($k$)].

Proof : (1) L_1 can be accepted by a CS-1FA(2) $M = (A_1, A_2)$ which acts as follows. Given an input $w \in \{0,1\}^+$, A_1 starts at speed 1 to sweep the input, and nondeterministically changes its speed to 1/3 when scanning the symbol "1". Once A_1 changes its speed, it will sweeps the remainder of the input at the same speed. On the other hand, A_2 sweeps the input tape at the same speed 1/2. If A_1 and A_2 reach the right endmarker at the same time, then M accepts the input. Otherwise, it rejects the input. (See Fig. 2.2.) It will be obvious that $T(M) = L_1$.

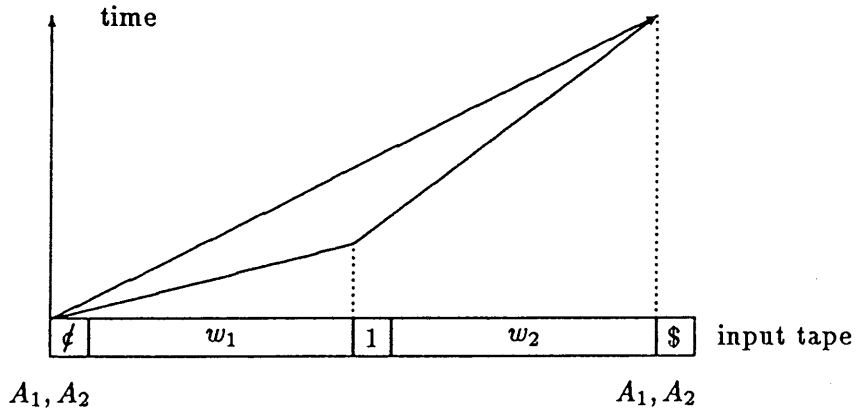

Fig. 2.2. Profile of an accepting computation of M.

(2) Suppose that there is a CS-1DFA(k) $M = (A_1, A_2, \cdots, A_k)$ accepting L_1 for some $k \geq 1$. For each $n \geq 1$, let

$$V(n) = \{0^n w | w \in \{0,1\}^* \ \& \ |w| = n\}.$$

For each $1 \leq i \leq k$ and each $x \in V(n)$, let $t_i(x)$, $q_i(x)$ denote the time and the internal state, respectively, just after A_i sweeps the subword x during the computation of M on the input $x0^p$ ($p \geq 1$), and let $t_{\min}(x) = \min\{t_1(x), t_2(x), \cdots, t_k(x)\}$. Furthermore, for each $x \in V(n)$, let

$$u(x) = \langle (q_1(x), t_1(x) - t_{\min}(x)), (q_2(x), t_2(x) - t_{\min}(x)), \cdots, (q_k(x), t_k(x) - t_{\min}(x)) \rangle.$$

Then, the following statement must hold:

Statement. For any two different words $x, y \in V(n)$, $u(x) \neq u(y)$.

⌈ For otherwise suppose that for some two $x, y \in V(n)$, $x \neq y$ and $u(x) = u(y)$. Let $w(i)$ denote the i-th symbol of w from the left. Then for some $n \leq i \leq 2n$, $x(i) \neq y(i)$, since $x, y \in V(n)$ and $x \neq y$. Without loss of generality, let $x(i) = 1$ and $y(i) = 0$.

Consider the following two words, $z = x0^{2i-2n-1}$ and $z' = y0^{2i-2n-1}$. It is easy to ascertain that $z \in L_1$. Hence z must be accepted by M. On the other hand, z' must be also accepted by M, because $u(x) = u(y)$. This contradicts the fact that z' is not in L_1. ⌋

For each $n \geq 1$, let $U(n) = \{u(x) | x \in V(n)\}$. By Lemma 2.3, $t_i(x) - t_{\min}(x) = O(n)$ for each $1 \leq i \leq k$ and $x \in V(n)$, so $|U(n)| = O(n^{k-1})$. On the other hand, $|V(n)| = 2^n$. Therefore, it follows that for large n, $|V(n)| > |U(n)|$, and thus there must exist two different words $x, y \in V(n)$ such that $u(x) = u(y)$. This contradicts the above statement, and thus (2) holds. □

From Lemma 2.4, we can get the following theorem.

Theorem 2.3. For each $k \geq 2$,

(1) $\mathcal{L}$[CS-1DFA(k)]$\subsetneq\mathcal{L}$[CS-1FA(k)], and

(2) $\bigcup_{1\leq k<\infty}\mathcal{L}$[CS-1DFA($k$)]$\subsetneq\bigcup_{1\leq k<\infty}\mathcal{L}$[CS-1FA($k$)].

2.4 One-way versus Two-way

In this section, we investigate the difference between the accepting powers of CS-1FA(k)'s and CS-FA(k)'s. We give an example language that is acceptable by a CS-DFA(2), but not by any CS-1FA(k).

Lemma 2.5. Let $L_2 = \{w\#w^R | w \in \{0,1\}^*\}$, where w^R denotes the reversal of word w. Then

(1) $L_2 \in \mathcal{L}$[CS-DFA(2)] and

(2) $L_2 \notin \bigcup_{1 \leq k < \infty} \mathcal{L}$[CS-1FA($k$)].

Proof : (1) We construct a CS-DFA(2) $M = (A_1, A_2)$ accepting language L_2. M executes the following steps :

Step 0. M checks if the input word is well formed, that is, if it is of the form $w\#w'$ with $w, w' \in \{0,1\}^*$ and $|w| = |w'|$. These actions can be easily done.

Step 1. Let $w = a_1 a_2 \cdots a_m$ and $w' = a'_m \cdots a'_2 a'_1$ (when the input word is well formed). Now A_1 and A_2 simultaneously start to move on the symbol "a_1", and A_1 stores a_1 in its finite control. A_1 moves left and returns at speed 1 when scanning the left endmarker, while A_2 moves right and returns at speed 1 when scanning the right endmarker. It is easy to see that when A_1 and A_2 simultaneously reach the same cell again, the symbol on the cell is "a'_1". Then A_1 checks if $a_1 = a'_1$.

Step i. $i = 2, \cdots, m$. Suppose that A_1 has already verified that $a_1 = a'_1, \cdots, a_{i-1} = a'_{i-1}$. Now if A_1 and A_2 are scanning the same symbol "a'_{i-1}" (this happens for even i's), then A_1 and A_2 simultaneously move left one cell, and A_1 stores a'_i in its finite control; if A_1 and A_2 are scanning the same symbol "a_{i-1}" (this happens for add i's), then A_1 and A_2 simultaneously move right one cell, and A_1 stores a_i in its finite control. After that, A_1 and A_2 acts as in Step 1 to check if $a_i = a'_i$. M accepts the input if and only if for each $1 \leq i \leq m$, $a_i = a'_i$. (See Fig. 2.3.)

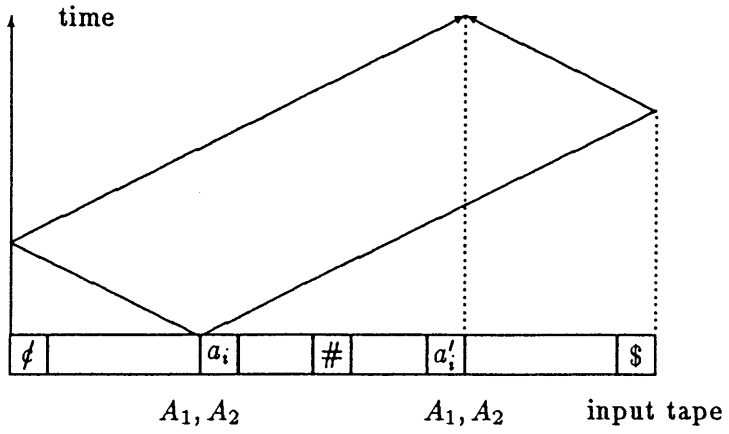

Fig. 2.3. Comparing a_i with a'_i.

(2) It is not difficult to prove, by using the technique in the proof of Lemma 2.2 (2), that for any $k \geq 1$, $L_2 \notin \mathcal{L}$[CS-1FA(k)]. □

The following theorem follows from Lemma 2.5.

Theorem 2.4. For each $k \geq 2$,

(1) $\mathcal{L}$[CS-1FA(k)]$\subsetneq\mathcal{L}$[CS-FA(k)],

(2) $\mathcal{L}$[CS-1DFA(k)]$\subsetneq\mathcal{L}$[CS-DFA(k)],

(3) $\bigcup_{1\leq k<\infty}\mathcal{L}$[CS-1FA($k$)]$\subsetneq\bigcup_{1\leq k<\infty}\mathcal{L}$[CS-FA($k$)],

(4) $\bigcup_{1\leq k<\infty}\mathcal{L}$[CS-1DFA($k$)]$\subsetneq\bigcup_{1\leq k<\infty}\mathcal{L}$[CS-DFA($k$)].

2.5 CS-FA's with Reversal Number Restriction

It has been known that the reversal complexity is intimately connected to parallel time complexity and uniform circuit depth [45, 46]. In this section, we consider the CS-FA(k)'s with reversal number restriction and investigate how this restriction affects the accepting power of CS-FA(k)'s. Our result mainly follows from the J. Hromkovic's work [47] where a language is

constructed to fool two-way nondeterministic (sensing) multihead finite automata with reversal number restriction, by using a generalized technique presented by Yao-Rivest [8].

Let f be a function from natural numbers to the positive real numbers. We denote by CS-FA(k)-R(f)'s the CS-FA(k)'s such that in their accepting computations each finite-state automaton may use at most $f(n)$ head reversals for input words of length n.

Lemma 2.6. Let

$$L_3' = \{w_1cw_2c\cdots cw_r\natural w_rc\cdots cw_2cw_1 | r \geq 1 \ \& \ w_i \in \{0,1\}^* \text{ for } i = 1,2,\cdots,r\}$$

and

$$L_3 = \{x_1\#x_2\#\cdots\#x_m\# | m \geq 1 \ \& \ x_i \in L_3' \text{ for } i = 1,2,\cdots,m\}.$$

Then

(1) $L_3 \in \mathcal{L}$[CS-DFA(2)] and

(2) $L_3 \notin \bigcup_{1\leq k<\infty} \mathcal{L}$[CS-FA($k$)-R($n^a$)], where $0 < a < 1/3$.

Proof : (1) To prove $L_3 \in \mathcal{L}$[CS-DFA(2)], it is enough to show that $L_3' \in \mathcal{L}$[CS-DFA(2)]. We construct a CS-DFA(2) $M = (A_1, A_2)$ accepting language L_3'. M executes the following steps :

Step 0. M checks if the input word is well formed, that is, if it is of the form

$$w_1cw_2c\cdots cw_r\natural w_r'c\cdots cw_2'cw_1'$$

with $w_i, w_i' \in \{0,1\}^*$ for each $1 \leq i \leq r$ and $|w_1cw_2c\cdots cw_r| = |w_r'c\cdots cw_2'cw_1'|$. These actions can be easily done.

Step 1. Let $w_i = a_{i,1}a_{i,2}\cdots a_{i,l_i}$ and $w_i' = a_{i,1}'a_{i,2}'\cdots a_{i,l_i'}'$ for each $1 \leq i \leq r$ (when the input word is well formed). Now A_1 and A_2 simultaneously start to move on the symbol of the left end of w_r, "$a_{r,1}$", and A_1 stores $a_{r,1}$ in its finite control. Both A_1 and A_2 move at speed 1. A_1 moves left, turns to the right when first scanning the symbol "c", and turns to

the left again when scanning the right endmarker, while A_2 moves right, turns to the left when (first) scanning the symbol "$\natural$" (or "c"), and turns to the right again when scanning the left endmarker. It is easy to observe that when A_1 and A_2 simultaneously reach the same cell again, the symbol on the cell is "$a'_{r,1}$". So A_1 can check whether $a_{r,1} = a'_{r,1}$. If this is the case, then A_1 and A_2 simultaneously move right one cell to the symbol "$a'_{r,2}$", and check, in the same way, whether $a_{r,2} = a'_{r,2}$. Clearly, A_1 and A_2 can check, by repeating the above oprations, whether $a_{r,j} = a'_{r,j}$ for each $1 \leq j \leq \min\{l_r, l'_r\}$. Thus M can verify that $w_r = w'_r$. (See Fig. 2.4.)

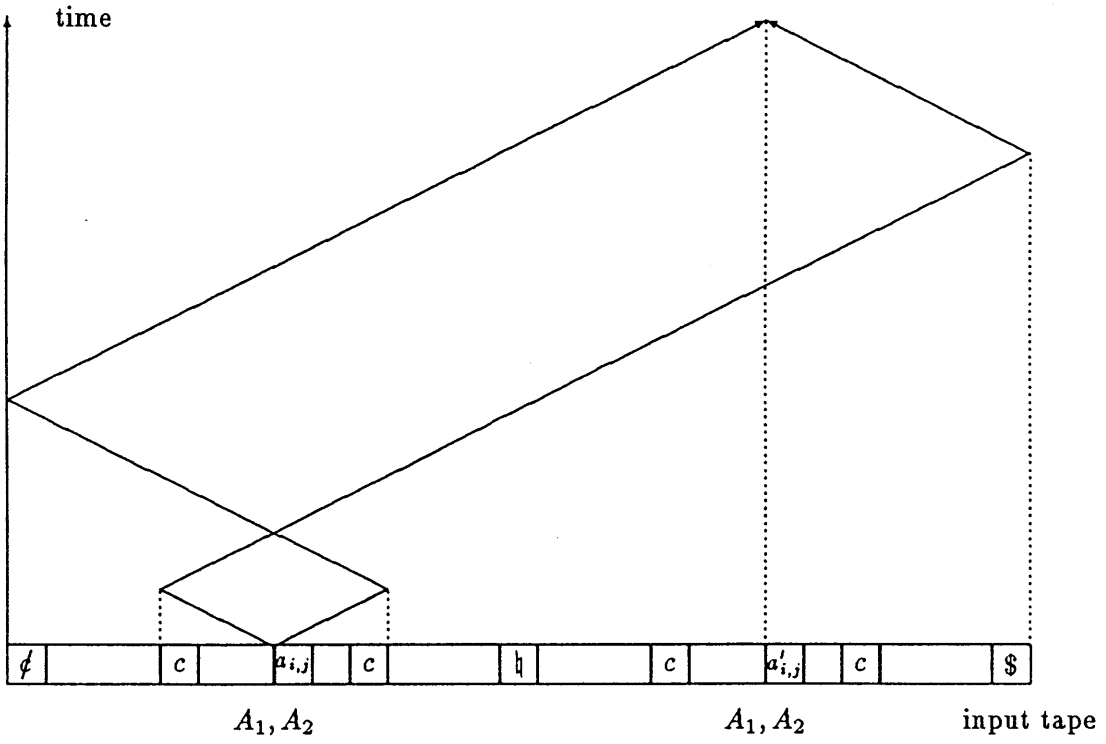

Fig. 2.4. Comparing $a_{i,j}$ with $a'_{i,j}$.

Step i. $i = 2, \cdots, r$. Suppose that M has already verified that $w_r = w'_r$, $\cdots$, $w_{r-i+2} = w'_{r-i+2}$. Now if A_1 and A_2 are scanning the same symbol "$a_{r-i+2,l_{r-i+2}}$", then A_1 and A_2 simultaneously move left to the symbol of the left end of w_{r-i+1}, "$a_{r-i+1,1}$", and A_1 stores "$a_{r-i+1,1}$" in its finite control; if A_1 and A_2 are scanning the same symbol "$a'_{r-i+2,l_{r-i+2}}$", then

A_1 and A_2 simultaneously move right to the symbol of the left end of w'_{r-i+1}, "$a'_{r-i+1,1}$", and A_1 stores "$a'_{r-i+1,1}$" in its finite control. After that, A_1 and A_2 acts as in Step 1 to check if $w_{r-i+1} = w'_{r-i+1}$. It will be obvious that M accepts the input if and only if for each $1 \leq i \leq r$, $w_i = w'_i$.

(2) It was shown in [47] that $L_3 \notin \bigcup_{1 \leq k < \infty} \mathcal{L}$[SeHA($k$)-R($n^a$)] for any $0 < a < 1/3$, where SeHA(k)-R(n^a) denote a SeHA(k) which may use in its accepting computations at most n^a head reversals for input words of length n. Clearly, (2) follows from this fact, since $\mathcal{L}$[CS-FA(k)-R(n^a)]$\subseteq\mathcal{L}$[SeHA(k)-R(n^a)]. □

The following theorem is derived from Lemma 2.6.

Theorem 2.5. For each $k \geq 2$, and $0 < a < 1/3$,

(1) $\mathcal{L}$[CS-FA(k)-R(n^a)]$\subsetneq\mathcal{L}$[CS-FA(k)],

(2) $\mathcal{L}$[CS-DFA(k)-R(n^a)]$\subsetneq\mathcal{L}$[CS-DFA(k)],

(3) $\bigcup_{1 \leq k < \infty} \mathcal{L}$[CS-FA($k$)-R($n^a$)]$\subsetneq \bigcup_{1 \leq k < \infty} \mathcal{L}$[CS-FA($k$)],

(4) $\bigcup_{1 \leq k < \infty} \mathcal{L}$[CS-DFA($k$)-R($n^a$)]$\subsetneq \bigcup_{1 \leq k < \infty} \mathcal{L}$[CS-DFA($k$)].

2.6 Closure Properties

In this section, we investigate closure properties of the classes of languages accepted by CS-1FA(k)'s.

We first examine closure properties for the deterministic case.

Lemma 2.7. For each $k \geq 1$, the class of languages accepted by CS-1DFA(k)'s is closed under union and intersection with arbitrary regular languages.

Proof : Let M be any CS-1DFA(k) and R any regular language. By Lemma 2.3, there exists a CS-1DFA(k) M' satisfying the properties described in Lemma 2.3. Consider a CS-1DFA(k) M'' which acts as follows. M'' simulates the action of M', and makes some finite-state

automaton try to accept R at the same time. It is easy to verify that M'' can be constructed to accept $T(M') \cup R$ or $T(M') \cap R$. □

Lemma 2.8. For each $k \geq 1$, let

$$A(k) = \{0^{m_1}10^{m_2}1\cdots10^{m_k}20^m \in \{0,1,2\}^+ | \forall i(1 \leq i \leq k)[m_i \geq 1] \\ \& \exists j(1 \leq j \leq k)[m_j = m]\},$$

$$A(\infty) = \{0^{m_1}10^{m_2}1\cdots10^{m_k}20^m \in \{0,1,2\}^+ | k \geq 1 \ \& \ \forall i(1 \leq i \leq k)[m_i \geq 1] \\ \& \exists j(1 \leq j \leq k)[m_j = m]\}.$$

Then,

(1) $A(k) \notin \mathcal{L}$[CS-1DFA(k)] and

(2) $A(\infty) \notin \bigcup_{1 \leq k < \infty} \mathcal{L}$[CS-1DFA($k$)].

Proof : (1) Suppose that there is a CS-1DFA(k) $M = (A_1, A_2, \cdots, A_k)$ accepting $A(k)$. For each $n \geq 1$, let

$$V(n) = \{0^{m_1}10^{m_2}1\cdots10^{m_k}2 | \forall i(1 \leq i \leq k)[1 \leq m_i \leq n]\}.$$

For each $1 \leq i \leq k$, let $t_i(x)$, $q_i(x)$ denote the time and the internal state, respectively, when A_i first reads the symbol "2" of $x \in V(n)$ during the computation of M on an arbitrary word w containing the prefix x, and let $t_{\min}(x) = \min\{t_1(x), t_2(x), \cdots, t_k(x)\}$. Furthermore, for each $x = 0^{m_1}10^{m_2}1\cdots10^{m_k}2 \in V(n)$, let

$$u(x) = \langle (q_1(x), t_1(x) - t_{\min}(x)), (q_2(x), t_2(x) - t_{\min}(x)), \cdots, (q_k(x), t_k(x) - t_{\min}(x)) \rangle,$$

$$N(x) = \{m | \exists j(1 \leq\leq k)[m_j = m]\}.$$

and for each $1 \leq i \leq k$, let

$$W_i(n) = \{x \in V(n) | t_i(x) = t_{\min}(x)\},$$

$$S_i(n) = \{N(x) | x \in W_i(n)\}.$$

Then, the following statement must hold :

Statement. For each $1 \leq i \leq k$ and any two words $x, y \in W_i(n)$ such that $N(x) \neq N(y)$, $u(x) \neq u(y)$.

⌈ For otherwise suppose that

$$x = 0^{m_1}10^{m_2}1\cdots10^{m_k}2,$$

$$y = 0^{m'_1}10^{m'_2}1\cdots10^{m'_k}2,$$

$x, y \in W_i(n)$, $N(x) \neq N(y)$, and $u(x) = u(y)$. Let $m \in N(x) - N(y)$ and $m = m_j$.

Consider the following two words,

$$z = x0^{m_j} = 0^{m_1}10^{m_2}1\cdots10^{m_k}20^{m_j},$$

and

$$z' = y0^{m_j} = 0^{m'_1}10^{m'_2}1\cdots10^{m'_k}20^{m_j}.$$

It is easy to see that $z \in A(k)$. Hence z must be accepted by M. On the other hand, z' must be also accepted by M, because $u(x) = u(y)$. This contradicts the fact that z' is not in $A(k)$ (since $m_j \notin N(y)$). ⌋

Clearly, for some $1 \leq j \leq k$,

$$|S_j(n)| \geq \left[\binom{n}{1} + \binom{n}{2} + \cdots + \binom{n}{k}\right] / k = \Omega(n^k).$$

Let $U_j(n) = \{u(x) | x \in W_j(n)\}$. Since for each $x \in W_j(n)$, $|w| = O(n)$, by Lemma 2.3, it follows that for each $i \in \{1, \cdots, j-1, j+1, \cdots, k\}$, $t_i(x) - t_{\min}(x) = O(n)$. Thus $|U_j(n)| = O(n^{k-1})$. (Note that $t_j(x) = t_{\min}(x)$ for each $x \in W_j(n)$.) Therefore, there must exist two different words $x, y \in W_j(n)$ such that $N(x) \neq N(y)$ and $u(x) = u(y)$, if $|S_j(n)| > |U_j(n)|$ which happens for n large enough. This contradicts the above statement, and thus (1) holds.

(2) For each $k \geq 1$, let

$$A'(k) = \{0^{m_1}10^{m_2}1\cdots10^{m_k}20^m \in \{0,1,2\}^+ | \forall i(1 \leq i \leq k)[m_i \geq 1] \ \& \ m \geq 1\}.$$

Clearly, the language $A'(k)$ is regular for each $k \geq 1$. Suppose that $A(\infty) \in \bigcup_{1 \leq k < \infty} \mathcal{L}$[CS-1DFA($k$)]. Then, there is a CS-1DFA(k) accepting $A(\infty)$ for some $k \geq 1$. By Lemma 2.7, $A(\infty) \cap A'(k) = A(k) \in \mathcal{L}$[CS-1DFA($k$)]. This contradicts (1) of the lemma, and thus (2) holds.

□

Theorem 2.6.

(1) For each $k \geq 2$, $\mathcal{L}$[CS-1DFA(k)] is closed under complementation, but not under union and intersection.

(2) $\bigcup_{1 \leq k < \infty} \mathcal{L}$[CS-1DFA($k$)] is closed under union, intersection and complementation.

Proof : (1) By Lemma 2.3, it is not difficult to prove that $\mathcal{L}$[CS-1DFA(k)] is closed under complementation for any $k \geq 1$. Without loss of generality, one can only consider the CS-1DFA(k)'s satisfying the properties (2), (3) described in Lemma 2.3. Then, For any CS-1DFA(k) M, one may define a new CS-1DFA(k) as the same as M except that the accepting and non-accepting states are switched.

For each $k \geq 1$ and $1 \leq j \leq k$, let

$$T'(k,j) = \{0^{m_1}1 \cdots 10^{m_k}20^{m'_1}1 \cdots 10^{m'_k} \in \{0,1,2\}^+ | \forall i(1 \leq i \leq k)[m_i, m'_i \geq 1] \ \& \ m_j = m'_j\}.$$

Then, it is easy to verify that $T'(k,j) \in \mathcal{L}$[CS-1DFA(2)]. On the other hand, by Lemma 2.2, $T'(k,1) \cap \cdots \cap T'(k,k) = T(k) \notin \mathcal{L}$[CS-1FA($k$)]. This means that $\mathcal{L}$[CS-1DFA(k)] is not closed under intersection for each $k \geq 2$.

Since $\overline{L_1 \cap L_2} = \overline{L_1} \cup \overline{L_2}$, where the overbar denotes complementation with respect to an alphabet including the alphabets of L_1 and L_2, it follows from the above facts that $\mathcal{L}$[CS-1DFA(k)] is not closed under union for each $k \geq 2$.

(2) We leave the proof to the reader. □

Lemma 2.9. For each $k \geq 1$, let

$$B(k) = \{1^j 0^{m_j} 20^{m_1} 10^{m_2} 1 \cdots 10^{m_k} \in \{0,1,2\}^+ | 1 \leq j \leq k$$

$$\& \forall i(1 \leq i \leq k)[m_i \geq 1]\},$$

$$B(\infty) = \{1^j 0^{m_j} 20^{m_1} 10^{m_2} 1 \cdots 10^{m_k} \in \{0,1,2\}^+ | k \geq 1 \ \& \ 1 \leq j \leq k$$

$$\& \forall i(1 \leq i \leq k)[m_i \geq 1]\}.$$

Then,

(1) $B(k) \in \mathcal{L}$[CS-1DFA(2)],

(2) $B(\infty) \in \mathcal{L}$[CS-1DFA(3)],

(3) $B(k)^R \notin \mathcal{L}$[CS-1DFA($k$)], and

(4) $B(\infty)^R \notin \bigcup_{1 \leq k < \infty} \mathcal{L}$[CS-1DFA($k$)].

Proof : (1) $B(k)$ can be accepted by a CS-1DFA(2) $M = (A_1, A_2)$ which acts as follows: Consider the case when an input word

$$\not\in 1^j 0^{m'_j} 20^{m_1} 10^{m_2} 1 \cdots 10^{m_k} \$, \ 1 \leq j \leq k,$$

is presented to M. (Input words in the form different from the above can easily be rejected by M.)

(i) A_1 sweeps the subwords 1^j and $0^{m'_j}$ at speed 1/2, and the other parts at speed 1; A_2 sweeps the subwords 1^j and 0^{m_j} at speed 1/2, and the other parts at speed 1.

(ii) A_1, A_2 can enter an accepting state when scanning the right endmarker \$ if and only if A_1 and A_2 scan the same cell just after they sweep the subword 0^{m_j}. (See Fig. 2.5.)

Note that $m'_j = m_j$ if and only if A_1 and A_2 scan the same cell just after they sweep the subword 0^{m_j}. So M accepts $B(k)$.

(2) $B(\infty)$ can be accepted by a CS-1DFA(3) $M = (A_1, A_2, A_3)$ which acts as follows: Consider the case when an input word

$$\not\in 1^j 0^{m'_j} 20^{m_1} 10^{m_2} 1 \cdots 10^{m_k} \$ \ \text{(with} \ j \geq 1, \ k \geq 1),$$

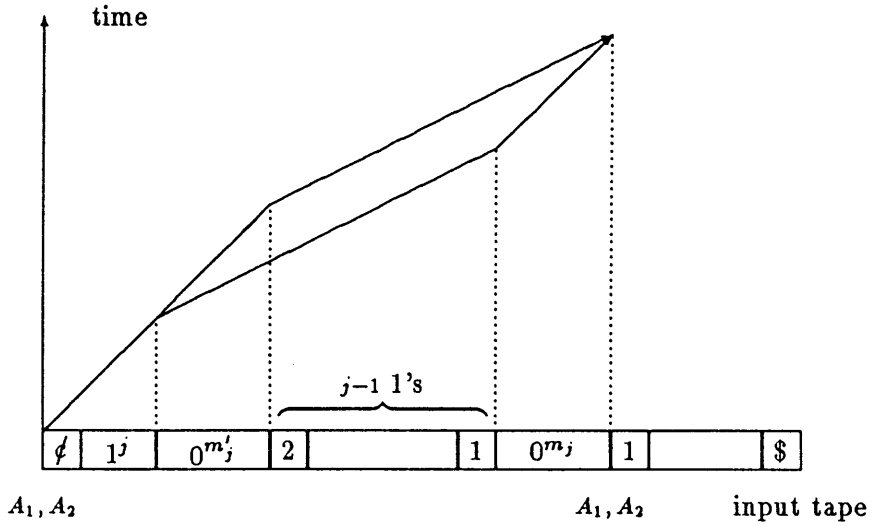

Fig. 2.5. Comparing m'_j with m_j.

is presented to M. (Input words in the form different from the above can easily be rejected by M.)

(i) A_1 sweeps the subword $1^j0^{m'_j}$ at speed 1, the symbols "2" and every "1" on the right hand of "2" at speed 1/2, and the other parts at speed 1.

(ii) A_2 sweeps the subword 1^j at speed 1/2, and after then changes its speed to 1. If A_2 and A_1 scan the same cell just after they read some symbol "1", then A_2 changes its speed to 1/2 and sweeps the remainder of the input tape (in this speed).

(iii) A_3 sweeps the subword $1^j0^{m'_j}$ at speed 1/2, and the other parts at speed 1.

(iv) A_1, A_2, A_3 can enter an accepting state when scanning the right endmarker \$ if and only if A_1 and A_2 scan the same cell just after they read some symbol "1", and after then A_2 and A_3 scan the identical symbol "1" just after A_2 sweeps the sequential 0's. Note that this happens if and only if $k \geq j$ and $m'_j = m_j$, that is, the input is in $B(\infty)$. (See Fig. 2.6.)

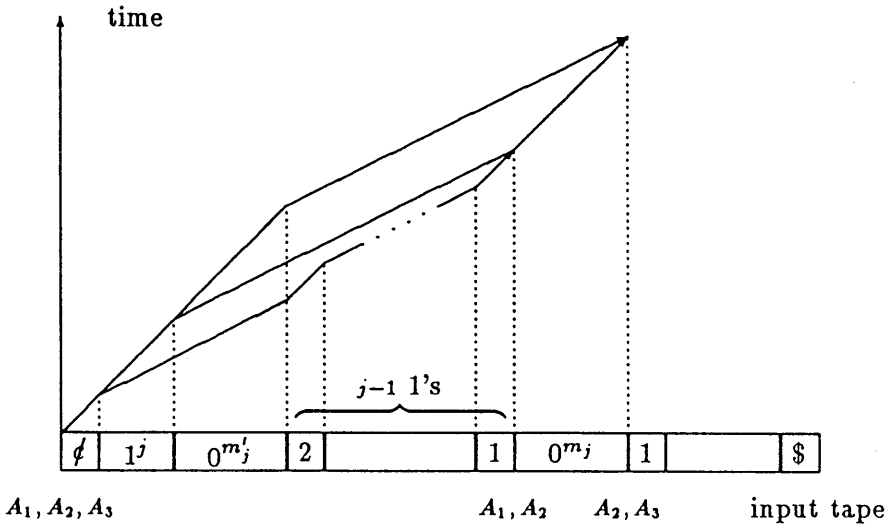

Fig. 2.6. Comparing m'_j with m_j (if $k \geq j$).

(3),(4) In the same way as in the proof of Lemma 2.8, we can prove (3), (4) of the Lemma, but we do not go into details. □

Theorem 2.7. For each $k \geq 2$, neither $\mathcal{L}$[CS-1DFA(k)] nor $\bigcup_{1 \leq k < \infty} \mathcal{L}$[CS-1DFA($k$)] is closed under the following operations :

(1) reversal "^{R}",

(2) concatenation,

(3) Kleene closure "*",

(4) nonerasing homomorphism.

Proof : (1) Nonclosure under reversal follows from Lemma 2.9.

(2) Let

$$H = \{0^{m_1}10^{m_2}1 \cdots 10^{m_k}20^{m_1} \in \{0,1,2\}^+ | k \geq 1 \ \& \ \forall i(1 \leq i \leq k)[m_i \geq 1]\},$$

and $T = \{0^n1 | n \geq 1\}^*$. It is easy to see that $H, T \in \mathcal{L}$[CS-1DFA(2)]. On the other hand, $TH = A(\infty)$, by Lemma 2.8, and thus $TH \notin \bigcup_{1 \leq k < \infty} \mathcal{L}$[CS-1DFA($k$)].

(3) Let $G = H \cup T$. It is easy to see that $G \in \mathcal{L}$[CS-1DFA(2)]. On the other hand, $G^* \cap T\{0\}^+\{2\}\{0\}^+ = A(\infty)$. Form this fact and Lemmas 2.7, 2.8, it follows that $G^* \notin \bigcup_{1 \leq k < \infty} \mathcal{L}$[CS-1DFA($k$)].

(4) Let

$$L_4 = \{w_1 2 w_2 | w_1, w_2 \in \{0,1\}^* \ \& \ |w_1| = |w_2|\}.$$

We can easily show that $L_4 \in \mathcal{L}$[CS-1DFA(2)]. On the other hand, let h be the nonerasing homomorphism defined by $h(0) = 0$ and $h(1) = h(2) = 1$. Then $h(L_4) = L_1$, where L_1 is the language given in Lemma 2.4, and it follows that $h(L_4) \notin \bigcup_{1 \leq k < \infty} \mathcal{L}$[CS-1DFA($k$)], by Lemma 2.4. □

We next examine closure properties for the nondeterministic case.

Lemma 2.10. Let $L_5 = \{w2w | w \in \{0,1\}^*\}$. Then,

(1) $\overline{L_5} \in \mathcal{L}$[CS-1FA(2)], and

(2) $L_5 \notin \bigcup_{1 \leq k < \infty} \mathcal{L}$[CS-1FA($k$)].

Proof : (1) $\overline{L_5}$ can be accepted by a CS-1FA(2) $M = (A_1, A_2)$ which acts as follows: Consider the case when an input word $\not\in w_1 2 w_2 \$$, $w_1, w_2 \in \{0,1\}^*$, $|w_1| = |w_2|$, is presented to M. (Input words in the form different from the above can easily be accepted or rejected by M.) A_1 moves at speed 1 until it nondeterministically stores in its finite control some symbol (0 or 1) scanned by its input head during the sweep of the subword "w_1", and then moves at speed 1/2, while A_2 sweeps the subword "$w_1 2$" at speed 1/2 and the subword "w_2" at speed 1. A_1, A_2 can enter an accepting state on the right endmarker if and only if A_1 and A_2 simultaneously reach the same cell and the symbol on the cell is different from the symbol stored in the finite control of A_1. (See Fig. 2.7.) It is easy to verify that $T(M) = \overline{L_5}$.

(2) The proof is similar to that of (2) of Lemma 2.2, and omited here. □

Theorem 2.8.

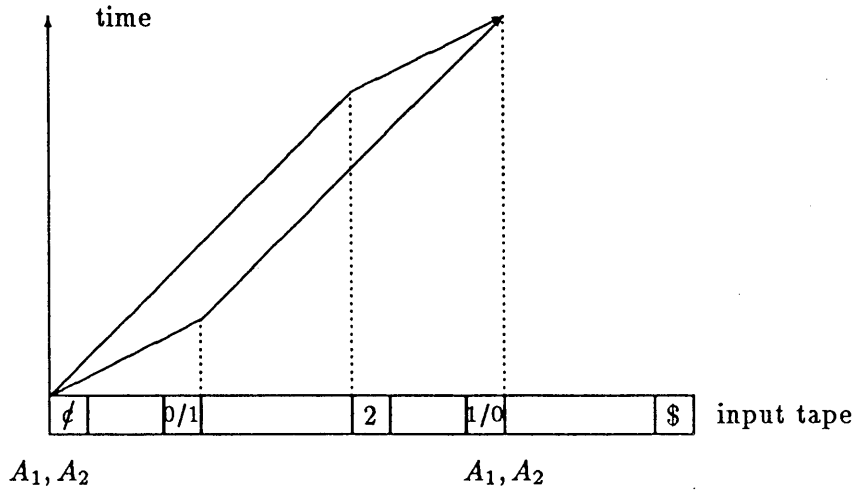

Fig. 2.7. An example for recognition of $w \in \overline{L_5}$ by M.

(1) For each $k \geq 2$, $\mathcal{L}$[CS-1FA(k)] is closed under union, but not under intersection and complementation.

(2) $\bigcup_{1 \leq k < \infty} \mathcal{L}$[CS-1FA($k$)] is closed under union and intersection, but not under complementation.

Proof : (1) It is straightforward that $\mathcal{L}$[CS-1FA(k)] is closed under union. The proof for nonclosure under intersection is the same as in the deterministic case (see the proof of (1) of Lemma 2.6). Nonclosure under complementation follows immediately from Lemma 2.10.

(2) Nonclosure under complementation follows from Lemma 2.10, and the proofs for other closure properties are left to the reader. □

Theorem 2.9. For each $k \geq 1$, $\mathcal{L}$[CS-1FA(k)] is closed under reversal "^{R}", concatenation and Kleene closure "*". Thus, so does $\bigcup_{1 \leq k < \infty} \mathcal{L}$[CS-1FA($k$)].

Proof : The proof for closure under reversal "^{R}" is similar to the proof for the case $k = 1$; a CS-1FA(k) can run a computation "in reverse". Let L be accepted by a CS-1FA(k) $M = (A_1, A_2, \cdots, A_k)$, where for each $1 \leq i \leq k$, $A_i = (\Sigma, Q_i, X_i, \delta_i, q_{0_i}, F_i, ¢, \$, \phi)$. Now

consider the CS-1FA(k) $M' = (A'_1, A'_2, \cdots, A'_k)$ which, on input $\not w^R\$$, simulates M on $\not w\$$, by using the following algorithm :

(1) For each $1 \leq i \leq k$, A'_i chooses some accepting state of A_i, $q_{if} \in F_i$, and stores it in the finite control (denote the state stored in the finite control q_i).

(2) **Repeat**

For each $1 \leq i \leq k$, A'_i chooses a $q'_i \in Q_i$, $a'_i \in \Sigma \cup \{\not, \$\}$, $d'_i \in \{0, +1\}$ and $x'_i \in X_i$ such that $(q_i, d'_i) \in \delta_i(a'_i, x'_i, q'_i)$, and moves right d'_i cells. If the values of a'_i and x'_i guessed by some A_i are not correct (this can be detected by comparing them with the symbol scanned by A_i and the states of the other finite-state automata on the same cell scanned by A_i at the moment, since A_i's can communicate with each other on the same cell), A'_i rejects the input. Otherwise, set $q_i = q'_i$.

Until the simulation cannot continue.

(3) If all A'_i's are scanning the right endmarker and $q_i = q_{0_i}$ for each $1 \leq i \leq k$, M' accepts the input.

It is straightforward to verify that M' accepts L^R. This shows that $\mathcal{L}$[CS-1FA(k)] is closed under reversal "^{R}".

To show closure under concatenation, let M' and M'' be two CS-1FA(k)'s. We consider the CS-1FA(k) M which acts as follows: While sweeping the first part of the input word x, M simulates the action of M' on the first part, and each finite-state automaton of M simultaneously remenbering in the finite control whether it is nearest to the right endmarker \$. Let A_i ($1 \leq i \leq k$) be one finite-state automaton that is nearest to \$. A_i nondeterministically guesses the arrival at the right end of the first part of x, and after that, without moving, detects the arrivals of other finite-state automata at the same cell. If M finds out that M' accepts the first part of x by simulating the action of M' in this way, M next proceeds to simulate the action of M'' on

the latter part of x, and accepts if and only if the latter part is also accepted by M''. It will be obvious that $T(M) = T(M')T(M'')$.

In a similar way, one can prove closure under Kleene closure "*". □

The following theorem is derived from Theorem 3.4 (in the next chapter) and the fact that the class of languages accepted by one-way nondeterministic (one-) counter machines is a full AFL (*Abstract Family of Languages*) [48].

Theorem 2.10. $\mathcal{L}$[CS-1FA(2)] is a full AFL.

The closure results obtained above are summarized in Table 2.1, where $\mathcal{L}(\mathrm{FA})_k^D = \mathcal{L}$[CS-1DFA($k$)], $\mathcal{L}(\mathrm{FA})_k^N = \mathcal{L}$[CS-1FA($k$)], $\mathcal{L}(\mathrm{FA})_\infty^D = \bigcup_{1\le k<\infty} \mathcal{L}$[CS-1DFA($k$)], $\mathcal{L}(\mathrm{FA})_\infty^N = \bigcup_{1\le k<\infty}$ $\mathcal{L}$[CS-1FA(k)], "$\surd$" means that the class is closed, "×" means that the class is not closed, and "?" means that the closure property is not known.

Table 2.1. Closure properties of CS-1FA(k)'s, $k \ge 2$

	$\mathcal{L}(\mathrm{FA})_k^D$	$\mathcal{L}(\mathrm{FA})_k^N$	$\mathcal{L}(\mathrm{FA})_\infty^D$	$\mathcal{L}(\mathrm{FA})_\infty^N$
complementation	$\surd$	×	$\surd$	×
union	×	$\surd$	$\surd$	$\surd$
intersection	×	×	$\surd$	$\surd$
concatenation	×	$\surd$	×	$\surd$
reversal	×	$\surd$	×	$\surd$
Kleene closure	×	$\surd$	×	$\surd$
nonerasing homomorphism	×	$\surd(k=2)$ $?(k\ge 3)$	×	?

2.7 Concluding Remarks

In this chapter, the cooperating system of finite-state automata was introduced as a new non-writing and parallel string acceptor, and its basic properties were investigated. Especially, we obtained some hierarchy results about CS-FA(k)'s with respect to the number of finite-state automata, nondeterminism–determinism, two way–one way, and the bounded reversal number. Fig. 2.8 is a summary of the inclusion relations, which hold for these CS-FA(k)'s, $k \ge 2$. All inclusions are proper. Looking back at the results in this chapter in more detail, we observe that

as the resources of computation, (1) nondeterminism cannot make up for an additional finite-state automaton for CS-1FA(k)'s (Lemma 2.2), additional finite-state automata cannot make up for the nondeterminism for CS-1DFA(k)'s (Lemma 2.4), (3) additional finite-state automata plus nondeterminism cannot make up for the two-way power for CS-1FA(k)'s (Lemma 2.5), and (4) additional finite-state automata plus nondeterminism cannot make up for the unlimitness to the number of reversals for CS-FA(k)-R(n^a)'s with $0 \leq a \leq 1/3$ (Lemma 2.6).

We also proved the closure results on CS-1FA(k)'s under the Boolean (union, intersection, complementation) and Kleene (concatenation, closure, and reversal) operations. In the next chapter, we will establish a relationship of CS-FA(k)'s to the other types of automata, and consider several decision problems associated with CS-FA(k)'s.

We conclude this chapter by listing some open problems related to the ones investigated in this chapter.

(1) Is $\mathcal{L}$[CS-FA(k)] ($\mathcal{L}$[CS-DFA(k)])$\subsetneq$ $\mathcal{L}$[CS-FA($k+1$)] ($\mathcal{L}$[CS-DFA($k+1$)]) for each $k \geq 2$? (We believe that the proof seems to require new techniques.)

(2) Does there exist a language $L \in \mathcal{L}$[CS-1FA(2)] such that $L \notin \mathcal{L}$[CS-DFA(k)] for any $k \geq 1$? (We will see in the next chapter that this problem is closely related to some open problem concerning deterministic and nondeterministic tape-bounded Turing computations.)

(3) Does Theorem 2.1 still hold even if the input alphabet is restricted to a one-letter? (We will see in the next chapter that this is true for CS-1FA(k)'s, $k \leq 3$.)

(4) Is $\mathcal{L}$[CS-1FA(k)] closed under nonerasing homomorphism for each $k \geq 3$?

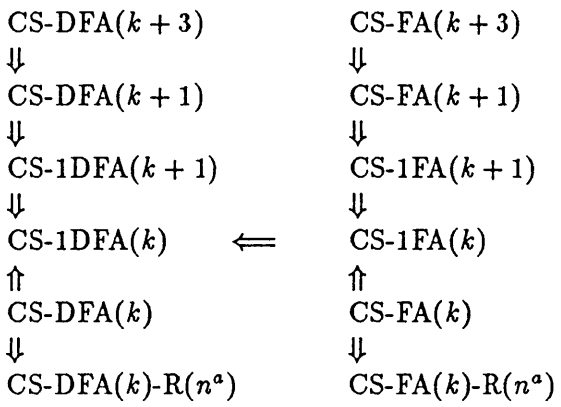

Fig. 2.8. The relations for accepting power of CS-FA(k)'s according to a variety of restrictions, where $k \geq 2$ and $a \leq 1/3$. (A⇒B indicates $\mathcal{L}$[B]$\subsetneq \mathcal{L}$[A].)

CHAPTER 3

A Relationship to Other Automata

In this chapter, we prove some results establishing a relationship of the cooperating systems of finite-state automata to the other automata of tape complexity $O(\log n)$. We will consider the following types of automata: (1) multihead finite automata, (2) marker finite automata, (3) multiprocessor automata, and (4) multicounter machines. In particular, it is shown that one-way (one-)counter machines and CS-1FA(2)'s are equivalent in accepting power. Several results concerning the decision problems associated with CS-1FA(k)'s are simple consequences of this fact.

3.1 Definitions and Notation

In this section, we recall the definitions of multihead finite automata, marker finite automata, multiprocessor automata, and multicounter machines.

3.1.1 Multihead Finite Automata

A two-way k-head finite automaton, HA(k), is an 8-tuple $(k, \Sigma, Q, \delta, q_0, F, \not{c}, \$)$ where

- $k \geq 1$ is the number of input heads,
- Σ is a finite input alphabet ($\not{c}, \$ \notin \Sigma$),
- Q is a finite set of states,
- δ is the transition function mapping $Q \times (\Sigma \cup \{\not{c}, \$\})^k$ to $2^{Q \times \{-1,0,+1\}^k}$,

- q_0 in Q is the initial state,

- $F \subseteq Q$ is the set of accepting states,

- $\not\!\!\$, \$$ are the left and right endmarkers, respectively.

A HA(k) M consists of a finite control, a read-only input tape with endmarkers and k input heads. At the start of the computation, M is set to its initial state, with all input heads positioned on the left endmarker. A single move of M is described as follows: Let M be in state q and the k input heads scanning symbols $a_1, a_2, \cdots, a_k$, respectively. If $\delta(q, a_1, a_2, \cdots, a_k) = \emptyset$, then M has no next move (i.e., M halts). If $(p, d_1, d_2, \cdots, d_k)$ is in $\delta(q, a_1, a_2, \cdots, a_k)$, then M may go to state p and move right its i-th head d_i cells. The input heads are prevented from going off either end of the input. The input is accepted by M if there is a sequence of moves that lands M in an accepting state.

A deterministic HA(k) is a HA(k) $= (k, \Sigma, Q, \delta, q_0, F, \not\!\!\$, \$)$ such that, for all $(q, a_1, a_2, \cdots, a_k)$ in $Q \times (\Sigma \cup \{\not\!\!\$, \$\})^k$, $\delta(q, a_1, a_2, \cdots, a_k)$ contains at most one element.

A one-way HA(k) is a HA(k) $= (k, \Sigma, Q, \delta, q_0, F, \not\!\!\$, \$)$ such that, for all $(q, a_1, a_2, \cdots, a_k)$ in $Q \times (\Sigma \cup \{\not\!\!\$, \$\})^k$, if $(p, d_1, d_2, \cdots, d_k) \in \delta(q, a_1, a_2, \cdots, a_k)$, then $d_1, d_2, \cdots, d_k \in \{0, +1\}$.

A simple HA(k) (denoted by SHA(k)) is a HA(k) whose only one head (called the "reading" head) is capable of distinguishing the symbol in the input alphabet, and whose other heads (called the "counting" heads) can only detect whether they are on the left endmarker $\not\!\!\$$, the right endmarker \$ or on a symbol in the input alphabet [11, 12].

We will use the following abbreviations.

- HA(k) : a two-way k-head (nondeterministic) finite automaton,
- DHA(k) : a two-way k-head deterministic finite automaton,
- SeHA(k) : a two-way k-head sensing (nondeterministic) finite automaton,
- SeDHA(k) : a two-way k-head sensing deterministic finite automaton,

- SHA(k) : a two-way k-head simple (nondeterministic) finite automaton,
- SDHA(k) : a two-way k-head simple deterministic finite automaton,
- SeSHA(k) : a two-way k-head sensing simple (nondeterministic) finite automaton,
- SeSDHA(k) : a two-way k-head sensing simple deterministic finite automaton,
- 1HA(k) : a one-way k-head (nondeterministic) finite automaton,
- 1DHA(k) : a one-way k-head deterministic finite automaton,
- 1SeHA(k) : a one-way k-head sensing (nondeterministic) finite automaton,
- 1SeDHA(k) : a one-way k-head sensing deterministic finite automaton,
- 1SHA(k) : a one-way k-head simple (nondeterministic) finite automaton,
- 1SDHA(k) : a one-way k-head simple deterministic finite automaton,
- 1SeSHA(k) : a one-way k-head sensing simple (nondeterministic) finite automaton,
- 1SeSDHA(k) : a one-way k-head sensing simple deterministic finite automaton.

3.1.2 Marker Finite Automata

A k-marker finite automaton (denoted by MA(k)) is an 8-tuple $(k, \Sigma, Q, \delta, q_0, F, \not{\epsilon}, \$)$ where $\Sigma, Q, q_0, F, \not{\epsilon}, \$$ have the same meanings as before. k is the number of markers. δ is the transition function mapping $Q \times K \times (\Sigma \cup \{\not{\epsilon}, \$\}) \times \{0,1\}$ to $2^{Q \times K \times \{L,R,N\} \times \{0,1\}}$, where "0" and "1" mean the absence and the presence of a marker on the cell scanned by the input head of MA(k), and $K = \{0, 1, 2, \cdots, k\}$. The integer $i \in K$ means the number of markers "carried" by MA(k) at the current moment. Furthermore, δ satisfies the following conditions: For each $(q, i, a, x) \in Q \times K \times (\Sigma \cup \{\not{\epsilon}, \$\}) \times \{0,1\}$, if $(p, j, d, y) \in \delta(q, i, a, x)$, then $i + x = j + y$, that is, only the following combinations of (i, x, j, y) are possible: $(i, 0, i, 0)$, $(i, 0, i-1, 1)$, $(i, 1, i+1, 0)$, and $(i, 1, i, 1)$, which have the interpretations shown in Table 3.1.

Table 3.1.

(i, x, j, y)	Meaning
$(i, 0, i, 0)$	MA(k) was carring $i(\geq 0)$ markers, and no marker was at the current position; after the transition, MA(k) did not put down any marker.
$(i, 0, i-1, 1)$	MA(k) was carring $i(\geq 1)$ markers, and no marker was at the current position; after the transition, MA(k) did put down some marker.
$(i, 1, i+1, 0)$	MA(k) was carring $i(\geq 0)$ markers, and there was some marker at the current position; after the transition, MA(k) did pick it up.
$(i, 1, i, 1)$	MA(k) was carring $i(\geq 0)$ markers, and there was some marker at the current position; but MA(k) did not pick it up after the transition.

A MA(k) M is a finite-state automaton with k distinguishable markers, $m_1, m_2, \cdots, m_k$. These markers can be thought of as labeled pebbles that can be placed on or removed from only the cell the input head is currently scanning. Furthermore, M can place at most one marker on any cell of the input tape. (Note that one can show that there is no difference in computing power between marker finite automata that place at most one marker on any cell of the input tape and those that can place up to some constant k markers on a cell.) At the start of the computation, M is set to its initial state with the input head positioned on the left endmarker, and all markers are "carried" by the finite control. Acceptance is defined in the obvious way. (Note that the markers here are assumed to be physical and labeled. This does not lose generality, since Blum and Hewitt [26] had shown that marker finite automata with abstract and physical markers are equivalent in computing power, as are those with labeled and unlabeled markers.)

A MA(k) $= (k, \Sigma, Q, \delta, q_0, F, \not{\epsilon}, \$)$ is called deterministic (denoted by DMA(k)) if $|\delta(q, i, a, x)| \leq 1$ for each (q, i, a, x) in $Q \times K \times (\Sigma \cup \{\not{\epsilon}, \$\}) \times \{0, 1\}$.

3.1.3 Multiprocessor Automata

A two-way finite automaton with k processors (denoted by PA(k)) is a structure $M = (Q, \Sigma, g, h, q_0, F, \not{\epsilon}, \$)$, where $\Sigma, Q, F, \not{\epsilon}, \$$ have the same meanings as before. g, the transition function, is a mapping from $Q \times (\Sigma \cup \{\not{\epsilon}, \$\})$ into $2^{Q \times \{-1,0,+1\}}$. h, the switching function, is a mapping from $\{1, 2, \cdots, k\} \times Q^k$ into $\{0, 1\}$. $q_0 \in Q^k$ is the k-tuple of the initial states.

A PA(k) consists of k processors that read information from the input tape simultaneously and the switching function that depends on the internal states of all processors on the step of computation. If processors $p_1, p_2, \cdots, p_k$ are in states $q_1, q_2, \cdots, q_k$ and scan $a_1, a_2, \cdots, a_k$ on the input tape, and $(q_i', d_i) \in g(q_i, a_i)$ for each $1 \leq i \leq k$, then each processor p_i such that $h(i, q_1, q_2, \cdots, q_k) = 1$ may enter state q_i' and move right itself d_i cells on the input tape. If $h(i, q_1, q_2, \cdots, q_k) = 0$, then processor p_i must miss the step. At the start of the computation, all processors are set to the initial state and scanning the left endmarker. An input is accepted by a PA(k) if all processors finally enter an accepting state. (Note that the definition of acceptance here is different from one in [42], where the acceptance is defined as follows: an input is accepted if processors $p_1, p_2, \cdots, p_k$ enter states $q_1, q_2, \cdots, q_k$ such that for all $i = 1, 2, \cdots, k$, $h(i, q_1, q_2, \cdots, q_k) = 0$. A relationship between two definitions of acceptance is discussed in [43].)

Let M be a PA(k) as defined above. If $|g(q_i, a_i)| \leq 1$ for any $1 \leq i \leq k$ and any $(q_i, a_i) \in Q \times \Sigma$, then M is a two-way deterministic finite automaton with k processors. If $d_i \neq -1$, for all (q_i', d_i) in $\delta(q_i, a_i)$, where (q_i, a_i) in $Q \times (\Sigma \cup \{\not{\epsilon}, \$\})$, then M is a one-way finite automaton with k processors. The following abbreviations will be used later.

- PA(k) : a two-way (nondeterministic) finite automaton with k processors,
- DPA(k) : a two-way deterministic finite automaton with k processors,
- 1PA(k) : a one-way (nondeterministic) finite automaton with k processors,
- 1DPA(k) : a one-way deterministic finite automaton with k processors.

3.1.4 Multicounter Machines

A two-way k-counter machine (denoted by CM(k)) is defined as a tuple $M = (k, \Sigma, Q, \delta, q_0, F, Z_0, Z, \not\epsilon, \$)$ where $\Sigma, Q, q_0, F, \not\epsilon, \$$ have the same meanings as before. k is the number of counters. δ is a mapping from $Q \times (\Sigma \cup \{\not\epsilon, \S\}) \times \{Z_0, Z\}^k$ into $2^{Q \times \{-1,0,+1\} \times \{+1,0,-1\}^k}$. Z_0 and Z (blank) are the two symbols of counters. Furthermore, the symbol Z_0, which serves as a bottom of the counter, appears initially on the bottom of counter and may never appear on any other place.

A CM(k) M consists of a finite control, a read-only input tape with endmarkers and k counters. At the start of the computation, M is set to its initial state, with the input head positioned on the left endmarker and with all counter heads positioned on the bottoms. A single move of M is described as follows: Let M be in state q, the input head scanning symbol a, and k counter heads scanning symbols a_1, a_2, $\cdots$, a_k, respectively. If $\delta(q, a, (a_1, a_2, \cdots, a_k)) = \emptyset$, then M has no next move (i.e., M halts). If $(p, d, d_1, d_2, \cdots, d_k)$ is in $\delta(q, a, (a_1, a_2, \cdots, a_k))$, then M may go to state p, move right its input head d cells, and move up the i-th counter head d_i cells. The input head and the counter heads are prevented from going off either end of the input and the bottoms of counters, respectively. Note that an integer x_i can be stored by moving up the counter head x_i cells from the bottom. The input is accepted by a CM(k) if there is a sequence of moves that lands it in an accepting state. (Note that there is no difference between acceptance by final state and by final state and empty counters except in the realtime case [49].)

A CM(k) $= (k, \Sigma, Q, \delta, q_0, F, Z_0, Z, \not\epsilon, \$)$ is called deterministic (denoted by DCM(k)) if $|\delta(q, a, (a_1, a_2, \cdots, a_k))| \leq 1$ for each $(q, a, (a_1, a_2, \cdots, a_k))$ in $Q \times (\Sigma \cup \{\not\epsilon, \S\}) \times \{Z_0, Z\}^k$. A CM($k$) (or DCM($k$)) is called one-way (denoted by 1CM(k) or 1DCM(k)) if for all $(p, d, (d_1, d_2, \cdots, d_k))$ in $\delta(q, a, (a_1, a_2, \cdots, a_k))$, where $(q, a, (a_1, a_2, \cdots, a_k))$ in $Q \times (\Sigma \cup \{\not\epsilon, \S\}) \times \{Z_0, Z\}^k$, $d \neq -1$.

A DCM(k) (or 1DCM(k)), M, accepts in time $T(n)$ if each input w accepted by M is accepted within $T(|w|)$ steps, and we denote it by DCM(k)-Time($T(n)$) (or 1DCM(k)-Time($T(n)$). A CM(k) (or 1CM(k)), M, accepts in time $T(n)$ if for each input w accepted by M there is a

computation of M on w which accepts in at most $T(|w|)$ steps, and we denote it by CM(k)-Time($T(n)$) (or 1CM(k)-Time($T(n)$)).

A DCM(k) (or 1DCM(k)), M, accepts in space $S(n)$ if for each input w accepted by M, each counter of M requires space not exceeding $S(|w|)$, and we denote it by DCM(k)-Space($S(n)$) (or 1DCM(k)-Space($S(n)$)). A CM(k) (or 1CM(k)), M, accepts in space $S(n)$ if for each input w accepted by M there is a computation of M on w in which each counter of M requires space not exceeding $S(|w|)$, and we denote it by CM(k)-Space($S(n)$) (or 1CM(k)-Space($S(n)$)).

3.2 Results

For the two-way case, it is straightforward to see that for each $k \geq 1$,

(1) $\mathcal{L}$[CS-FA(k)] ($\mathcal{L}$[CS-DFA(k)])
$\subseteq$ $\mathcal{L}$[SeHA(k)] ($\mathcal{L}$[DSeHA(k)])
$\subseteq$ $\mathcal{L}$[SHA($k+1$)] ($\mathcal{L}$[DSHA($k+1$)]),
$\mathcal{L}$[SeHA(k)] ($\mathcal{L}$[DSeHA(k)])
$\subseteq$ $\mathcal{L}$[CS-FA($k+1$)] ($\mathcal{L}$[CS-DFA($k+1$)]);

(2) $\mathcal{L}$[CS-FA(k)] ($\mathcal{L}$[CS-DFA(k)])
$\subseteq$ $\mathcal{L}$[MA(k)] ($\mathcal{L}$[DMA(k)]),
$\mathcal{L}$[MA(k)] ($\mathcal{L}$[DMA(k)])
$\subseteq$ $\mathcal{L}$[CS-FA($k+1$)] ($\mathcal{L}$[CS-DFA($k+1$)]);

(3) $\mathcal{L}$[CS-FA(k)] ($\mathcal{L}$[CS-DFA(k)])
$\subseteq$ $\mathcal{L}$[CM(k)-Space(n)] ($\mathcal{L}$[DCM(k)-Space(n)]),
$\mathcal{L}$[CM(k)-Space(n)] ($\mathcal{L}$[DCM(k)-Space(n)])
$\subseteq$ $\mathcal{L}$[CS-FA($k+1$)] ($\mathcal{L}$[CS-DFA($k+1$)]);

Moreover, it was shown in [43] that $\mathcal{L}$[PA(k)]=$\mathcal{L}$[HA(k)] for each $k \geq 1$. From this fact we get

(4) $\mathcal{L}[\text{CS-FA}(k)] \subseteq \mathcal{L}[\text{PA}(k+1)]$,

$\mathcal{L}[\text{PA}(k)] \subseteq \mathcal{L}[\text{CS-FA}(k+1)]$.

Thus,

$$\begin{aligned}
&\cup_{1\leq k<\infty}\mathcal{L}[\text{CS-FA}(k)] \\
&= \cup_{1\leq k<\infty}\mathcal{L}[\text{HA}(k)] \\
&= \cup_{1\leq k<\infty}\mathcal{L}[\text{MA}(k)] \\
&= \cup_{1\leq k<\infty}\mathcal{L}[\text{PA}(k)] \\
&= \cup_{1\leq k<\infty}\mathcal{L}[\text{CM}(k)\text{-Space}(n)] \\
&= \text{Space}(\log n),
\end{aligned}$$

and

$$\begin{aligned}
&\cup_{1\leq k<\infty}\mathcal{L}[\text{CS-DFA}(k)] \\
&= \cup_{1\leq k<\infty}\mathcal{L}[\text{DHA}(k)] \\
&= \cup_{1\leq k<\infty}\mathcal{L}[\text{DMA}(k)] \\
&= \cup_{1\leq k<\infty}\mathcal{L}[\text{DCM}(k)\text{-Space}(n)] \\
&= \text{DSpace}(\log n),
\end{aligned}$$

where Space($\log n$) (DSpace($\log n$)) denotes the class of languages accepted by nondeterministic (deterministic) Turing machines within space bound $\log n$.

Next we concentrate our attention on the one-way case.

Lemma 3.1. $\mathcal{L}[\text{1DHA}(2)] - \cup_{1\leq k<\infty}\mathcal{L}[\text{CS-1FA}(k)] \neq \emptyset$.

Proof : Let $L_5 = \{w2w | w \in \{0,1\}^*\}$ be the language defined in Lemma 2.10. It is obvious that $L \in \mathcal{L}[\text{1DHA}(2)]$. Thus the Lemma follows from this fact and Lemma 2.10 (2). □

From Lemma 3.1, we have at once the following theorem.

Theorem 3.1.

(1) For each $k \geq 2$, $\mathcal{L}$[CS-1FA(k)] ($\mathcal{L}$[CS-1DFA(k)]) $\subsetneq \mathcal{L}$[1SeHA(k)] ($\mathcal{L}$[1SeDHA(k)]), and

(2) $\bigcup_{1\leq k<\infty} \mathcal{L}$[CS-1FA($k$)] ($\bigcup_{1\leq k<\infty} \mathcal{L}$[CS-1DFA($k$)]) $\subsetneq \bigcup_{1\leq k<\infty} \mathcal{L}$[1SeHA($k$)] ($\bigcup_{1\leq k<\infty} \mathcal{L}$[1SeDHA($k$)]).

Theorem 3.2.

(1) For any $k, r \geq 2$, $\mathcal{L}$[CS-1DFA(k)] is incomparable with $\mathcal{L}$[1SHA(r)] ($\mathcal{L}$[1SDHA(r)]), and

(2) $\bigcup_{1\leq k<\infty} \mathcal{L}$[CS-1DFA($k$)] is incomparable with $\bigcup_{1\leq k<\infty} \mathcal{L}$[1SHA($k$)] ($\bigcup_{1\leq k<\infty} \mathcal{L}$[1SDHA ($k$)]).

Proof : Let $L_1 = \{w_1 1 w_2 | w_1, w_2 \in \{0,1\}^* \ \& \ |w_1| = |w_2|\}$ be the language defined in Lemma 2.4. By Lemma 2.4, $L_1 \notin \bigcup_{1\leq k<\infty} \mathcal{L}$[CS-1DFA($k$)]. It is shown, however, in [13] that $L_1 \in \mathcal{L}$[1SDHA(2)]. On the other hand, let $L_6 = \{a^n b^n | n \geq 1\}^*$, then one can easily show that L_6 is in $\mathcal{L}$[CS-1DFA(2)], but not in $\bigcup_{1\leq k<\infty} \mathcal{L}$[1SHA($k$)] (see Lemma 4.1 in [13]). The lemma follows from those facts. □

Lemma 3.2. $\mathcal{L}$[1DCM(2)-Time(cn)]$-\bigcup_{1\leq k<\infty} \mathcal{L}$[CS-1FA($k$)]$\neq \emptyset$ for some positive constant $c > 1$.

Proof : For a word w in $\{1\}\{0,1\}^*$, let $n(w)$ be the integer represented by w as a binary number. It is shown in [47] that $L_7 = \{w 2 0^{n(w)} | w \in \{1\}\{0,1\}^*\}$ is accepted by a 1DCM(2)-Time(cn), where $c > 1$ is a positive constant. On the other hand, using the technique in the proof of Lemma 2.2 (2), we can prove that L_7 cannot be accepted by any CS-1FA(k) for any $k \geq 1$. □

Lemma 3.3. For each $k \geq 1$ and each CS-1FA($k+1$) (CS-1DFA($k+1$)), M, there exists some positive constant $c \geq 1$ such that we can construct a 1CM(k)-Time(cn) (1DCM(k)-Time (cn)) to simulate M.

Proof: For each $k \geq 1$, let $M = (A_1, A_2, \cdots, A_{k+1})$ be a CS-1FA(k+1) (CS-1DFA(k+1)). We will construct a 1CM(k)-Time(cn) (1DCM(k)-Time(cn)) M' to simulate M, where c is some positive constant dependent only on M. Let $c_1, c_2, \cdots, c_k$ denote k counters of M'. M' acts as follows:

(1) M' keeps track in its finite control of what states A_1, A_2, $\cdots$, A_{k+1} are in when they read the cell of input tape scanned by M'.

(2) For each cell of input tape:

(a) M' stores in its finite control the internal state of each A_i ($1 \leq i \leq k+1$) when A_i leaves the cell, and the order $\langle t_1, t_2, \cdots, t_{k+1} \rangle$ in which A_1, A_2, $\cdots$, A_{k+1} leave the cell subsequently (i.e., A_{t_1} firstly leaves the cell, A_{t_2} secondly leaves the cell, and so on).[1]

(b) Furthermore, for each i ($1 \leq i \leq k$), the interval between the times at which A_{t_i} and $A_{t_{i+1}}$ leave the cell is stored by counter c_i. That is, M' adds the difference of steps between $A_{t_{i+1}}$ and A_{t_i} when they stay at the cell to the counter c_i.

In addition, by Lemma 2.1, if M accepts its input tape, it can do so in linear time. Thus, it is easy to verify that M' can simulate M in linear time. □

Note that the value of c (> 1) in Lemma 3.3 is not important, because any linear time multicounter machine can be replaced by an equivalent one that operates in time $(1+\epsilon)n$ for any $0 < \epsilon < 1$ [49]. Furthermore, we observe that for every 1CM(k)-Time(cn) (1DCM(k)-Time(cn)), M, one can efficiently use two one-way sensing counting heads to simulate a counter of M. This can be done as follows: when the counter contains the integer m, the distance between two counting heads is $\left\lfloor \frac{m}{c} \right\rfloor$, and the residue $m - c\left\lfloor \frac{m}{c} \right\rfloor$ is retained in the finite control. If the counter increases, move the leading head right without moving the lagging one; if the counter decreases, move the lagging head right without moving the leading one. So the distance

[1] If $A_{i_1}, A_{i_2}, \cdots, A_{i_r}$ ($1 \leq i_1 < i_2 < \cdots < i_r \leq k+1$) leave the cell simultaneously, we refer the order on them as $(i_1, i_2, \cdots, i_r)$.

between two counting heads will correspond to the contents of the counter. Thus one can easily construct a 1SeSFA($2k+1$) (1DSeSFA($2k+1$)) M' to simulate a 1CM(k)-Time(cn) (1DCM(k)-Time(cn)) M by using its $2k$ counting heads to simulate k counters of M in this way. From this observation and Lemmas 3.2, 3.3, the following theorem is obtained.

Theorem 3.3. For any $0 < \epsilon < 1$ and each $k \geq 2$,

(1) $\mathcal{L}$[CS-1FA($k+1$)]$\subsetneq\mathcal{L}$[1CM(k)-Time($(1+\epsilon)n$)],

(2) $\mathcal{L}$[CS-1DFA($k+1$)]$\subsetneq\mathcal{L}$[1DCM(k)-Time($(1+\epsilon)n$)],

(3) $\mathcal{L}$[CS-1FA($k+1$)]$\subsetneq\mathcal{L}$[1SeSHA($2k+1$)],

(4) $\mathcal{L}$[CS-1DFA($k+1$)]$\subsetneq\mathcal{L}$[1SeSDHA($2k+1$)],

(5) $\bigcup_{1\leq k<\infty}\mathcal{L}$[CS-1FA($k$)]$\subsetneq\bigcup_{1\leq k<\infty}\mathcal{L}$[1CM($k$)-Time($(1+\epsilon)n$)],

(6) $\bigcup_{1\leq k<\infty}\mathcal{L}$[CS-1DFA($k$)]$\subsetneq\bigcup_{1\leq k<\infty}\mathcal{L}$[1DCM($k$)-Time($(1+\epsilon)n$)],

(7) $\bigcup_{1\leq k<\infty}\mathcal{L}$[CS-1FA($k$)]$\subsetneq\bigcup_{1\leq k<\infty}\mathcal{L}$[1SeSHA($k$)],

(8) $\bigcup_{1\leq k<\infty}\mathcal{L}$[CS-1DFA($k$)]$\subsetneq\bigcup_{1\leq k<\infty}\mathcal{L}$[1SeSDHA($k$)].

The results of (1), (2) of Theorem 3.3 are "optimal" in the sense of Lemma 3.4. Moreover, the result of (2) of Theorem 3.3 cannot be "tightened" by setting the "ϵ" to zero (Lemma 3.5). It is open whether we can sharpen the results of (3), (4) in Theorem 3.3.

Lemma 3.4. For each $k \geq 1$, $\mathcal{L}$[CS-1DFA($k+1$)]$-\bigcup_{1\leq c<\infty}\mathcal{L}$[1CM($k-1$)-Time($cn$)]$\neq\emptyset$.

Proof: By using simple counting arguments (see [50] for details), we can show that $T(k)$ (defined in Lemma 2.2) cannot be accepted by any 1CM($k-1$)-Time(cn). Thus, the lemma follows from this fact and Lemma 2.2 (1). □

Lemma 3.5. $\mathcal{L}$[CS-1DFA(2)]$-\bigcup_{1\leq k<\infty}\mathcal{L}$[1DCM($k$)-Time($n$)]$\neq\emptyset$.

Proof : It is shown in [51] that $\{0^p1^m|p \geq m \geq 1\}^*$ is not in $\bigcup_{1\leq k<\infty}\mathcal{L}$[1DCM($k$)-Time($n$)]. On the other hand, it is easy to prove that $\{0^p1^m|p \geq m \geq 1\}^*$ can be accepted by some CS-1DFA(2). □

Theorem 3.4.

(1) $\mathcal{L}$[CS-1DFA(2)]=$\mathcal{L}$[1DCM(1)]$\subsetneq\mathcal{L}$[1SeSDHA(2)], and

(2) $\mathcal{L}$[CS-1FA(2)]=$\mathcal{L}$[1CM(1)]=$\mathcal{L}$[1SeSHA(2)].

Proof : (1) It is not difficult to prove that $\mathcal{L}$[1DCM(1)]$\subseteq\mathcal{L}$[1SeSDHA(2)]. We, next, show that $\mathcal{L}$[1DCM(1)]$\subseteq\mathcal{L}$[CS-1DFA(2)]. Let M be a 1DCM(1) with s internal states. We will construct a CS-1DFA(2) $M' = (A_1, A_2)$ to simulate M. M' acts as follows:

1. A_1 and A_2 keep track of the state of M on the input tape in their finite controls.

2. For each cell of the input tape:

   (a) If M reaches the cell with the memory configuration (q_0, c_0), where q_0 denotes the internal state and $c_0 \leq s$ denotes the contents of the counter, then

      (i) if M leaves the cell (within $2s^2$ steps) with the contents of the counter $s + c$ $(1 \leq c < s)$, then A_2 moves at speed $1/(1+c)$ on the cell and A_1 moves at speed 1 on the cell;

      (ii) if M leaves the cell (within $2s^2$ steps) with the contents of the counter c $(0 \leq c \leq s)$, then A_1 and A_2 leave the cell simultaneously (at speed 1) and store the contents of the counter of M in their finite controls.

      (iii) otherwise A_1 (A_2) rejects the input tape (because M enters a loop, that is, M will never leave the cell).

   (b) If M reaches the cell with the memory configuration (q_0, c_0), where $c_0 > s$, (in this case, A_1 and A_2 are on the different cells) then

(i) if M leaves the cell (within s steps) with the contents of the counter c_0, then A_1 and A_2 move at speed 1 on the cell;

(ii) if M leaves the cell (within s steps) with the contents of the counter $c_0 + \delta$ $(1 \leq \delta < s)$, then A_2 moves at speed $1/(\delta+1)$ on the cell and A_1 moves at speed 1 on the cell;

(iii) if M leaves the cell (within s steps) with the contents of the counter $c_0 - \delta$ $(1 \leq \delta < s)$, then A_1 moves at speed $1/(\delta+1)$ on the cell and A_2 moves at speed 1 on the cell;

(iv) otherwise there must exist a sequence of memory configurations $(q_0, c_0), (q_1, c_1), \cdots, (q_j, c_j)$, $j \leq s$, of M on the cell such that $q_i = q_j$ for some i $(0 \leq i < j)$. If $c_i \leq c_j$ (it means that M enters a loop), then A_1 (A_2) rejects the input tape; If $c_i > c_j$, then A_1 stays on the cell until A_2 reaches the cell, and afterwords A_1 (A_2) simulates the action of M on the cell with the contents of the counter s as in the case 2.(a).

Note that (A) if the contents of the counter of M exceeds s when M leaves a cell of the input tape, then M' stores it by using the difference between the times at which A_1 and A_2 leave the cell, and (B) otherwise M' can simulate the action of M using the finite control. It is easy to verify that M' is able to simulate M, (since M is deterministic).

So we get $\mathcal{L}$[1DCM(1)]=$\mathcal{L}$[CS-1DFA(2)] by Lemma 3.3, and $\mathcal{L}$[CS-1DFA(2)]$\subsetneq\mathcal{L}$[1SeSDH A(2)] by applying the proof of Theorem 3.2.

(2) It is shown in [52] that every 1CM(1) is equivalent to some 1CM(1)-Time(n) (i.e., some 1CM(1) which accepts in real-time). Let M be a 1CM(1)-Time(n). We will construct a CS-1FA(2) $M' = (A_1, A_2)$ to simulate M. M' acts as follows:

1. A_2 sweeps the input tape at the same speed 1/2.

2. A_1 keeps track of the state of M on the input tape in the finite control.

3. For each cell of the input tape:

(a) If M does not change the counter on the cell, then A_1 moves at speed 1/2 on the cell.

(b) If M increases the counter by 1 on the cell, then A_1 moves at speed 1 on the cell.

(c) If M decreases the counter by 1 on the cell, then FA_1 moves at speed 1/3 on the cell.

So the contents of the counter of M on each cell of the input tape corresponds to the difference between the times at which A_1 and A_2 leave the cell. It is easy to verify that M' is able to simulate M.

Consequently, (2) follows from the above fact, Lemma 3.3 and Lemma 7.1 of [13] (where it was shown that $\mathcal{L}$[1CM(1)]=$\mathcal{L}$[1SeSHA(2)]). □

Lemma 3.6. Let $L_7 = \{0^{2^n} | n \geq 2\}$. Then there exists a CS-1DFA(3) M such that M accepts L_7.

Proof : A CS-1DFA(3) to recognize the language L_7 need only check the input is of a form $b_0 b_1 \cdots b_k$ with $b_0 = 00$ and $b_i = \underbrace{0 \cdots 0}_{2^i}$ for each $1 \leq i \leq k$, since $2^{k+1} = 2 + 2^1 + \cdots + 2^k$ for any $k \geq 1$.

Now consider the CS-1DFA(3) $M = (A_1, A_2, A_3)$ which creates successively larger "blocks" of length a power of 2 on the input using the following algorithm:

(1) Locate A_1 on the 3rd cell, A_2 and A_3 on the 4th cell, from the left endmarker of the input tape, and then make A_1, A_2, A_3 start to move (to the right) simultaneously.

(2) **Repeat the following:**

A_1 moves at speed 1/4; A_2 moves at speed 1/3; A_3 stays on the cell (which is visited simultaneously by A_2 and A_3) until A_1 reaches the cell, and then moves at speed 1/2 until A_2 and A_3 reach the same cell again.

Until all of A_1, A_2 and A_3 reach the right endmarker of the input.

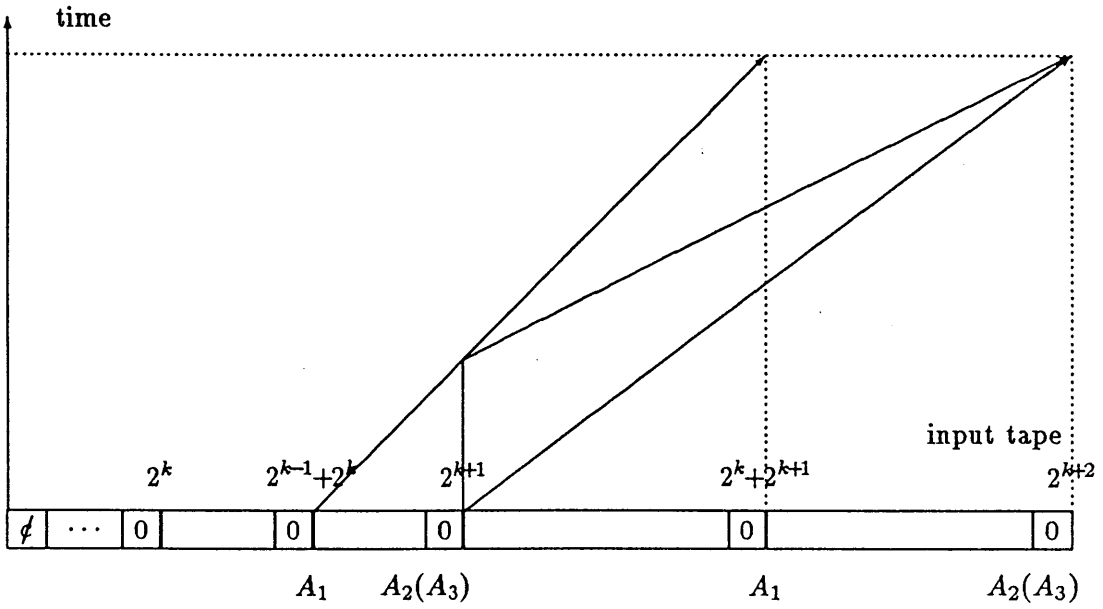

Fig. 3.1. Moves of M.

(3) If A_2 and A_3 reach the same cell simultaneously and the next cell is just the right end-marker, then M accepts the input. Otherwise M rejects the input. (See Fig. 3.1.)

It is easy to verify that M accepts the language L_7. □

It is well known [53] that the languages accepted by 1HA(k)'s over a one-letter alphabet are regular for any $k \geq 1$. From this fact and Lemmas 3.1 and 3.6, we get the following theorem.

Theorem 3.5.

(1) For each $r \geq 2$ and $k \geq 3$, $\mathcal{L}$[CS-1FA(k)] and $\mathcal{L}$[CS-1DFA(k)] are incomparable with $\mathcal{L}$[1HA(r)] (=$\mathcal{L}$[1PA(r)]) and $\mathcal{L}$[1DHA(r)].

(2) $\bigcup_{1\leq k<\infty}\mathcal{L}$[CS-1FA($k$)] and $\bigcup_{1\leq k<\infty}\mathcal{L}$[CS-1HA($k$)] are incomparable with $\bigcup_{1\leq k<\infty}\mathcal{L}$[1HA($k$)] (= $\bigcup_{1\leq k<\infty}\mathcal{L}$[1PA($k$)]) and $\bigcup_{1\leq k<\infty}\mathcal{L}$[1DHA($k$)].

It is also known [54] that context-free languages over a one-letter alphabet are regular, so are the languages in $\mathcal{L}$[1CM(1)]. From Theorem 3.4 and Lemma 3.6, we have the following corollary.

Corollary 3.1. Even over a one-letter alphabet, CS-1FA(2)'s (CS-1DFA(2)'s) are strictly less powerful than CS-1FA(3)'s (CS-1DFA(3)'s).

If a problem is undecidable for CS-1FA(2)'s (CS-1DFA(2)'s), then it is also undecidable for CS-1FA(k)'s (CS-1DFA(k)'s), for any $k \geq 2$. From this observation and the fact [55, 56] that the containment problem is undecidable for 1DCM(1)'s and the equivalence and universe problems are undecidable for 1NCM(1)'s, we get the next corollary of Theorem 3.4.

Corollary 3.2. For any $k \geq 2$, the containment problem is undecidable for CS-1DFA(k)'s, and the equivalence and universe problems are undecidable for CS-1FA(k)'s.

The following corollary relates to an important open problem of whether the classes of languages accepted by deterministic and nondeterministic $L(n)$ tape-bounded Turing machines are the same for $L(n) \geq \log n$, and is obtained by combining Theorem 3.4 with the result in [13]. This perhapes gives another possibility to investigate the above problem.

Corollary 3.3. $\mathcal{L}$[CS-1FA(2)] is contained in the class of languages accepted by deterministic $\log(n)$ tape-bounded Turing machines if and only if the classes of languages accepted by deterministic and nondeterministic $L(n)$ tape-bounded Turing machines are the same for $L(n) \geq \log n$.

Finally, we give some results concerning a relationship between CS-FA(k)'s and reversal-bounded CM(k)'s.

Let $f(n)$ be a real function defined on natural numbers, and 1CM(k)-R($f(n)$) denote a 1CM(k) which operates in such a way that for any input w of length n, if w is accepted, then there is an accepting computation on w in which each counter head reverses its direction (i.e., changes from increasing to decreasing of a counter or vice versa) at most $f(n)$ times.

For any nonnegative integers c, let DCM(k)-T(c) denote a DCM(k) which operates in such a way that in every accepting computation the input head reverses its direction at most c times.

Theorem 3.6.

(1) $\mathcal{L}$[CS-1DFA(2)]$-\bigcup_{1\leq k<\infty}\mathcal{L}$[1CM($k$)-R($f(n)$)]$\neq\emptyset$, where $f(n)=o(n)$ be a function from natural numbers to positive reals.

(2) For any nonnegative integer c, $\mathcal{L}$[DCM(1)-T(c)]$\not\subseteq\mathcal{L}$[CS-DFA(2)].

Proof : (1) Let

$$L_8=\{w|w\in\{a,b\}^* \ \&\ \#_a(w)=\#_b(w)\ \&\ \#_a(x)\geq\#_b(x) \text{ for any prefix } x \text{ of } w\},$$

where $\#_a(w)$ denotes the number of a's in w and similarly for $\#_b(w)$. It is easy to see that L_8 can be accepted by some CS-1DFA(2). On the other hand, it was shown in [57] that $L_8\notin\bigcup_{1\leq k<\infty}\mathcal{L}$[1CM($k$)-R($f(n)$)].

(2) Using the technique in the proof of Theorem 3.4 (1), one can easily prove that every DCM(1)-T(c) can be simulated by some CS-DFA(2)-R(c). Thus (2) follows from this fact and Lemma 2.6. □

3.3 Concluding Remarks

In this chapter, we have established some relationships of language acceptabilities between CS-FA(k)'s and other automata (HA(k)'s, MA(k)'s, PA(k)'s and CM(k)'s). We summarize the main results for the one-way case in Fig. 3.2.

By the proof of Theorem 3.4 and Lemma 3.3, we can see that every CS-1FA(2) is equivalent in computing power to some CS-1FA(2) in which one of automata is deterministic. In general, let CS-1FA(i,j) denote a CS-1FA($i+j$) M such that i automata of M operate nondeterministically and the other j automata operate deterministically. Then, we have $\mathcal{L}$[CS-1FA(2)]=$\mathcal{L}$[CS-1FA(1,1)].

In light of the above result, a natural problem arises as to whether $\mathcal{L}$[CS-1FA(k)] =$\mathcal{L}$[CS-1FA(i,j)] for $k\geq 3$ and some $i,j\geq 1$ such that $i+j=k$? Furthermore, it is natural to ask

whether the number of nondeterministic finite-state automata is a natural measure of computational complexity for CS-1FA(k)'s. Nondeterminism is one of the most elusive concepts in computing, and much remains to be learned about the basic notion of nondeterminism. We believe that it will be meaningful to solve the above problems, towards an analysis of the power inherent in nondeterminism.

By Theorem 3.4, we know that the equivalence problem is decidable for CS-1DFA(2)'s. It is very interesting to ask whether the equivalence problem is decidable for CS-1DFA(k)'s, for each $k \geq 3$.

Another interesting open problem is "what is the relationship between $\mathcal{L}$[CS-1FA($k+1$)] and $\mathcal{L}$[1CM(k)-Time(n)] for $k \geq 2$?".

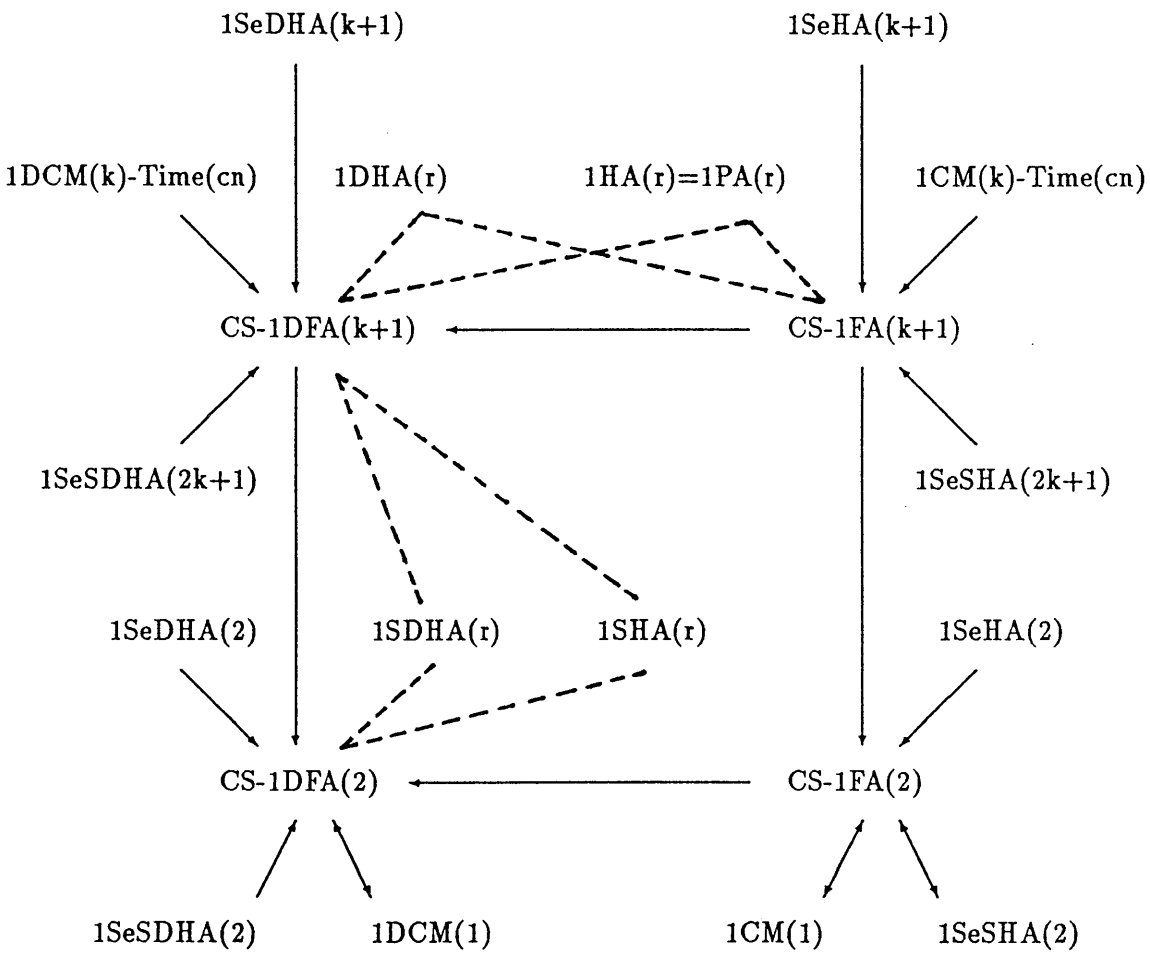

Fig. 3.2. Main relationships between the cooperating of system of one-way finite-state automata and other kinds of one-way automata, where $k, r \geq 2$, $c > 1$, and A$\longleftrightarrow$B means $\mathcal{L}$[A]=$\mathcal{L}$[B], A$\longrightarrow$B means $\mathcal{L}$[B]$\subsetneq\mathcal{L}$[A] and A- - -B means $\mathcal{L}$[A] and $\mathcal{L}$[B] are incomparable.

CHAPTER 4

Cooperating Systems of Two-Dimensional Finite-state Automata

In this chapter, we investigate some properties of cooperating systems of two-dimensional finite-state automata as two-dimensional language acceptors. We mainly propose a cooperating system of three-way two-dimensional finite-state automata, which can be considered as an extension of a cooperating system of one-way finite-state automata to two dimensions, and give some of its properties.

4.1 Definitions and Notation

Let Σ be a finite set of symbols. A two-dimensional tape over Σ is a two-dimensional rectangular array of elements of Σ.

The set of all two-dimensional tapes over Σ is denoted by $\Sigma^{(2)}$. Given a tape $w \in \Sigma^{(2)}$, we let $l_1(w)$ be the number of rows of w and $l_2(w)$ be the number of columns of w. If $1 \leq i \leq l_1(w)$ and $1 \leq j \leq l_2(w)$, we let $w(i,j)$ denote the symbol in w with coordinates (i,j). Furthermore, we define

$$w[(i,j),(i',j')],$$

only when $1 \leq i \leq i' \leq l_1(w)$ and $1 \leq j \leq j' \leq l_2(w)$, as the two-dimensional tape z satisfying the following:

(1) $l_1(z) = i' - i + 1$ and $l_2(z) = j' - j + 1$.

(2) for each k, r, $[1 \leq k \leq l_1(z),\ 1 \leq r \leq l_2(z)]$, $z(k,r) = w(k+i-1, r+j-1)$.

A cooperating system of k two-dimensional finite-state automata (CS-2-FA(k)) is denoted by

$$M = (\mathrm{FA}_1, \mathrm{FA}_2, \cdots, \mathrm{FA}_k),$$

such that for each $1 \leq i \leq k$, FA_i is a two-dimensional finite-state automaton defined by an 8-tuple $(\Sigma, Q_i, X_i, \delta_i, q_{0_i}, F_i, \#, \phi)$, where

- Σ is a finite input alphabet ($\phi, \$ \notin \Sigma$),
- Q_i is a finite set of states ($\phi \notin Q_i$),
- $X_i = (Q_1 \cup \{\phi\}) \times \cdots \times (Q_{i-1} \cup \{\phi\}) \times (Q_{i+1} \cup \{\phi\}) \times \cdots \times (Q_k \cup \{\phi\})$,
- δ_i is the transition function mapping $(\Sigma \cup \{\#\}) \times X_i \times Q_i$ to $2^{Q_i \times \{(0,-1),(0,+1),(-1,0),(+1,0),(0,0)\}}$,
- q_{0_i} in Q_i is the initial state,
- $F_i \subseteq Q_i$ is the set of accepting states,
- $\#$ not in Σ is the boundary symbol.

An input to M is a two-dimensional tape over Σ surrounded by boundary symbols $\#$. The action of M is similar to that of a CS-FA(k), except that every FA_i can move left ($=(0,-1)$), right ($=(0,+1)$), up ($=(-1,0)$) or down ($=(+1,0)$). That is, when an input tape $w \in \Sigma^{(2)}$ is presented to M, M starts its computation with every FA_i ($1 \leq i \leq k$) positioned on $w(1,1)$. A single move of M is described as follows: Let automata FA_1, FA_2, $\cdots$, FA_k be in states q_1, q_2, $\cdots$, q_k and scanning $w(x_1, y_1)$, $w(x_2, y_2)$, $\cdots$, $w(x_k, y_k)$ (note that $w(x_i, y_i)$ may equal $\#$) at the moment t. If $\delta_i(w(x_i, y_i), (q'_1, \cdots, q'_{i-1}, q'_{i+1}, \cdots, q'_k), q_i) = \emptyset$, then FA_i has no next move (i.e., FA_i halts). If $(p_i, (d_{i1}, d_{i2}))$ is in $\delta_i(w(x_i, y_i), (q'_1, \cdots, q'_{i-1}, q'_{i+1}, \cdots, q'_k), q_i)$, then FA_i may enter state p_i, move its input head to $w(x_i + d_{i1}, y_i + d_{i2})$ at the next moment $t+1$. Here for

each $j \in \{1, \cdots, i-1, i+1, \cdots, k\}$

$$q_j' = \begin{cases} q_j \in Q_j & \text{if } FA_i \text{ and } FA_j \text{ are on the same cell at the moment } t, \\ \phi & \text{otherwise.} \end{cases}$$

The input $\not\in x\$$ is accepted by M if there is a sequence of moves that leads all FA_i's to an accepting state when scanning one of the bottom boundary symbols. We assume that no FA_i can fall off the input tape beyond boundary symbols.

A cooperating system of k three-way two-dimensional finite-state automata (CS-TR2-FA(k)) is a CS-2-FA(k) M=(FA_1,FA_2,$\cdots$,FA_k) such that every FA_i can only move left, right, or down, but not up.

By CS-2-DFA(k) (respectively, CS-TR2-DFA(k)), we denote a deterministic version of CS-2-FA(k) (respectively, CS-TR2-FA(k)). For a two-dimensional automaton M, let M^S denote a M whose input tapes are restricted to square ones.

4.2 Three-Way versus Four-Way

In this section, we investigate the difference between the accepting powers of cooperating systems of two-dimensional finite-state automata and cooperating systems of three-way two-dimensional finite-state automata, and show that cooperating systems of two-dimensional finite-state automata are more powerful than cooperating systems of three-way two-dimensional finite-state automata.

We will use the fact that cooperating systems of three-way two-dimensional (deterministic) finite-state automata and three-way two-dimensional (deterministic) simple multihead finite automata accept the same family of two-dimensional languages when the input tapes are restricted to square ones (Theorem 4.7), which will be proved in Section 4.5.

Theorem 4.1. $\mathcal{L}$[CS-2-DFA(1)^S]$-\bigcup_{1\leq k<\infty}\mathcal{L}$[CS-TR2-FA($k$)^S]$\neq\emptyset$.

Proof : Let $T_1 = \{x \in \{0,\ 1\}^{(2)} \mid (\exists m \geq 2)\ [l_1(x) = l_2(x) = m\ \&\ x[(1,1),(1,m)] = x[(2,1),(2,m)]]\}$. Clearly, $T_1 \in \mathcal{L}$[CS-2-DFA(1)^S]. As shown in [58], T_1 cannot be accepted by any

three-way two-dimensional simple multihead finite automata. From this fact and Theorem 4.7, the theorem follows. □

From Theorem 4.1, we can get the following corollary.

Corollary 4.1. For each $k \geq 1$,

(1) $\mathcal{L}[\text{CS-TR2-FA}(k)^S] \subsetneq \mathcal{L}[\text{CS-2-FA}(k)^S]$,

(2) $\mathcal{L}[\text{CS-TR2-DFA}(k)^S] \subsetneq \mathcal{L}[\text{CS-2-DFA}(k)^S]$,

(3) $\bigcup_{1 \leq k < \infty} \mathcal{L}[\text{CS-TR2-FA}(k)^S] \subsetneq \bigcup_{1 \leq k < \infty} \mathcal{L}[\text{CS-2-FA}(k)^S]$,

(4) $\bigcup_{1 \leq k < \infty} \mathcal{L}[\text{CS-TR2-DFA}(k)^S] \subsetneq \bigcup_{1 \leq k < \infty} \mathcal{L}[\text{CS-2-DFA}(k)^S]$.

Note that the corresponding result obviously holds for general input tapes.

4.3 Nondeterminism versus Determinism

It is well known [26] that a nondeterministic finite-state automaton is more powerful than a deterministic finite-state automata in two dimensions (even if the input tapes are restricted to square ones). That is, $\mathcal{L}[\text{CS-2-DFA}(1)^S] \subsetneq \mathcal{L}[\text{CS-2-FA}(1)^S]$. In this section, we investigate the difference between the accepting powers of cooperating systems of three-way two-dimensional (nondeterministic) finite-state automata and cooperating systems of three-way two-dimensional deterministic finite-state automata whose input tapes are restricted to square ones.

Theorem 4.2. $\mathcal{L}[\text{CS-TR2-FA}(1)^S] - \bigcup_{1 \leq k < \infty} \mathcal{L}[\text{CS-TR2-DFA}(k)^S] \neq \emptyset$.

Proof : Let $T_2 = \{x \in \{0, 1\}^{(2)} \mid (\exists m \geq 2)\ [l_1(x) = l_2(x) = m]\ \&\ \exists i (1 \leq i \leq m)\ [x(1,i) = x(2,i) = 1]\}$. Clearly, $T_2 \in \mathcal{L}[\text{CS-TR2-FA}(1)^S]$. As shown in [58], T_2 cannot be accepted by any three-way two-dimensional deterministic simple multihead finite automata. From this fact and Theorem 4.7, the theorem follows. □

From Theorem 4.2, we can get the following corollary.

Corollary 4.2. For each $k \geq 1$,

(1) $\mathcal{L}$[CS-TR2-DFA(k)^S]$\subsetneq\mathcal{L}$[CS-TR2-FA(k)^S], and

(2) $\bigcup_{1\leq k<\infty}\mathcal{L}$[CS-TR2-DFA($k$)^S]$\subsetneq\bigcup_{1\leq k<\infty}\mathcal{L}$[CS-TR2-FA($k$)^S].

4.4 Hierarchies Based on the Number of Finite-State Automata

This section investigates how the number of finite-state automata affects the accepting power of the cooperating system of two-dimensional finite-state automata.

4.4.1 Four-Way Case

We first give hierarchies for the four-way case.

Theorem 4.3. For each $k \geq 1$,

(1) $\mathcal{L}$[CS-2-FA(k)^S]$\subsetneq$ $\mathcal{L}$[CS-2-FA($k+2$)^S], and

(2) $\mathcal{L}$[CS-2-DFA(k)^S]$\subsetneq$ $\mathcal{L}$[CS-2-DFA($k+2$)^S],

even if the alphabet is restricted to a one-letter.

Proof : It is easy to prove that every CS-2-FA(k) (CS-2-DFA(k)) can be simulated by a two-dimensional (deterministic) sensing k-head finite automaton (described in Section 4.5), and every two-dimensional (deterministic) sensing k-head finite automaton can be simulated by a CS-2-FA($k+1$) (CS-2-DFA($k+1$)). As shown in [44], for sets of square tapes over a one-letter alphabet, two-dimensional (deterministic) sensing $(k+1)$-head finite automata are more powerful tnan the corresponding k-head finite automata. From these facts, we immediately get the theorem. □

Unfortunately, it is unknown whether $\mathcal{L}$[CS-2-XFA(k)^S]$\subsetneq\mathcal{L}$[CS-2-FA($k+1$)^S] or $\mathcal{L}$[CS-2-XFA(k)^S]$\subsetneq\mathcal{L}$[CS-2-FA($k+1$)^S] for any $k \geq 2$.

Note that it is easy to show that $\mathcal{L}$[CS-2-FA(1)^S]$\subsetneq\mathcal{L}$[CS-2-FA(2)^S] and $\mathcal{L}$[CS-2-FA(1)^S]$\subsetneq\mathcal{L}$[CS-2-FA(2)^S]. In fact, it was shown in [59] that two-dimensional alternating Turing machines with

sublogarithmic space bound cannot accept the set of square tapes, T, which have the same upper and lower halves. On the other hand, it is easy to see that T can be accepted by a CS-2-DFA(2) (using a similar technique in the proof of Lemma 2.5 (1)).

4.4.2 Three-Way Case

We next give hierarchies for the three-way case.

For each $r \geq 1$, let

$$T(r) = \{x \in \{0,\ 1\}^{(2)} | \exists m \geq n[l_1(x) = l_2(x) = m\ \&$$
$$x[(1,1),(1,m)] = x[(2,1),(2,m)] \in R_r(m)\ \&\ x[(3,1),(m,m)] \in \{0\}^{(2)}]\},$$

where $R_r(m) = \{x \in \{0,\ 1\}^{(2)} \mid l_1(x) = 1,\ l_2(x) = m\ \&\ x$ has exactly r 1's$\}$ for each $m \geq r$.

It is obvious that for any fixed positive integer r, $T(r)$ can be accepted by a CS-TR2-DFA(r). We first consider the following problem: Given a fixed positive integer r, find a cooperating system of three-way two-dimensional (deterministic) finite-state automata that accepts $T(r)$ using the minimum number of finite-state automata. Unfortunately, we cannot generally solve the problem in the present thesis, but we can give the lower and upper bounds for this problem.

Let $\underline{f(r)}$ denote the minimum number of finite-state automata required for cooperating systems of three-way two-dimensional finite-state automata to accept $T(r)$, and $\underline{g(r)}$ denote the minimum number of finite-state automata required for cooperating system of three-way two-dimensional *deterministic* finite-state automata to accept $T(r)$. Clearly, $g(r) \geq f(r)$ for any $r \geq 1$.

Theorem 4.4. For each $k \geq 1$,

(1) $g(k^2 + k - 1) \leq 2k - 1$,

(2) $g(k^2 + 2k) \leq 2k$, and

(3) $g(k(k+1)/2 + 1) \geq k + 1$.

Before giving the proof of Theorem 4.4, we will give an example for showing how some CS-TR2-DFA(2) accepts $T(3)$, which will be used as a basis step in the proof of (1) (or (2)) of

Theorem 4.4 below.

Example. $T(3) \in \mathcal{L}$[CS-TR2-DFA(2)^S].

Proof : Let $T'(3) = \{x[(1,1),(2,l_2(x))] \mid x \in T(3)\}$ (i.e., $T'(3)$ is obtained from the first and second rows of every tape in $T(3)$). We actually show that there exists a CS-TR2-DFA(2) $M(2) = (A_1, A_2)$ accepting $T'(3)$, since one can easily make $M(2)$ accept $T(3)$. Let $h_i(t)$ denote the position of input head h_i of A_i at time t for each $i \in \{1,2\}$.

Consider the case when an input tape x with 2 rows and m columns such that $x[(1,1),(1,m)]$, $x[(2,1),(2,m)] \in R_3(m)$ is presented to $M(2)$ which acts as follows. (Input tapes in the form different from the above can easily be rejected by $M(2)$.) What we have to show is that how $M(2)$ checks whether the symbol on $p'_2(i)$ is 1 for each $i \in \{1,2,3\}$, where $p'_2(i)$ denotes the position just under the position, $p_1(i)$, of the i-th 1 in the first row (counting from left-to-right).

(1) A_1 and A_2 move h_1 and h_2 simultaneously to the position $p_1(1)$ (at some time t_0^x). Thus, $h_1(t_0^x) = h_2(t_0^x) = p_1(1)$. After that,

(2) (a) A_1 moves h_1 down one cell at speed 1 (thus, $h_1(t_1^x) = p'_2(1)$, $h_2(t_1^x - 1) = p_1(1)$, where $t_1^x = t_0^x + 1$), and then checks whether the symbol on $p'_2(1)$ is 1. If this is the case, then A_1 moves h_1 to the right at speed 1 from $p'_2(1)$ to the position of the next symbol 1 (denoted by $p_2(2)$) if it really exists, and then moves h_1 to the right at speed 1/2 from $p_2(2)$ to the position of next symbol 1 (denoted by $p_2(3)$) if it really exists. Otherwise A_1 halts forever (on the right boundary symbol attached to the second row of x), that is, $M(2)$ rejects x.

(b) A_2 moves h_2 to the right from $p_1(1)$ to $p_1(2)$ at speed 1 and from $p_1(2)$ to $p_1(3)$ at speed 1/2, and moves h_2 down one cell at speed 1.

(3) The time at which h_2 reaches $p'_2(3)$ is denoted by t_2^x. If h_1 and h_2 simultaneously reach $p_2(3)$ at time t_2^x (i.e., $p'_2(3) = p_2(3)$), $M(2)$ accepts x. Otherwise $M(2)$ rejects x. Note that h_1 and h_2 simultaneously reach $p_2(3)$ at time t_2^x if and only if $l_1 + l_2 = l'_1 + l'_2$ and

$l_1 + 2l_2 = l'_1 + 2l'_2$, where, for each $i \in \{1, 2\}$, l_i denotes the distance from $p_1(i)$ to $p_1(i+1)$, and l'_1 and l'_2 denote the distance from $p'_2(1)$ to $p_2(2)$ and from $p_2(2)$ to $p_2(3)$, respectively. That is, $l_1 = l'_1$ and $l_2 = l'_2$.

It will be obvious that $M(2)$ accepts $T'(3)$. □

Proof of Theorem 4.4 : The proofs of (1) and (2) are similar. We only give the proof of (2) here. To prove (2) is equivalent to proving that: for each $k \geq 1$, $T(k^2 + 2k) \in \mathcal{L}$[CS-TR2-DFA$(2k)^S$].

For each $r \geq 1$, let $T'(r) = \{x[(1,1),(2,l_2(x))] \mid x \in T(r)\}$ (i.e., $T'(r)$ is obtained from the first and second rows of every tape in $T(r)$).

For convenience, we prove by induction on k that $T'(k^2 + 2k) \in \mathcal{L}$[CS-TR2-DFA$(2k)$]. It will be obvious that (2) follows from this fact.

Basis step: k=1. There exists a CS-TR2-DFA(2) $M(2) = (A_1, A_2)$ accepting $T'(3)$ which satisfies the property that for each $x \in T'(3)$, there are three times $t_0^x < t_1^x < t_2^x$ during the accepting computation of $M(2)$ on x such that

[0.] $h_1(t_0^x) = h_2(t_0^x) = p_1(1)$,

[1.] $h_1(t_1^x) = p'_2(1)$, $h_2(t_1^x - 1) = p_1(1)$, and

[2.] $h_1(t_2^x) = h_2(t_2^x) = p'_2(3)$,

where $h_i(t)$ denotes the position of input head h_i of A_i at time t, $p_1(i)$ denotes the position of the i-th 1 in the first row (counting from left-to-right), and $p'_2(i)$ denotes the position just under $p_1(i)$. This has been shown in the above example.

Inductive Hypothesis: Suppose that for each $1 \leq j \leq k$, there exists a CS-TR2-DFA$(2j)$ $M(2j) = (A_1, A_2, \cdots, A_{2j})$ accepting $T'(j^2 + 2j)$ which satisfies the property that for each $x \in T'(j^2 + 2j)$, there are $2j + 1$ times $t_0^x < t_1^x < \cdots < t_{2j}^x$ during the accepting computation of $M(2j)$ on x such that

[0.] $h_1(t_0^x) = h_2(t_0^x) \cdots = h_{2j}(t_0^x) = p_1(d_j[1])$,

[1.] $h_1(t_1^x) = p_2'(d_j[1])$, $h_2(t_1^x - 1) = h_3(t_1^x - 1) = \cdots = h_{2j}(t_1^x - 1) = p_1(d_j[1])$,

$\vdots$

[i.] $h_1(t_i^x) = h_2(t_i^x) = \cdots = h_i(t_i^x) = p_2'(d_j[i])$,

$h_{i+1}(t_i^x - 1) = h_{i+2}(t_i^x - 1) = \cdots = h_{2j}(t_i^x - 1) = p_1(d_j[i])$,

$\vdots$

[2j.] $h_1(t_{2j}^x) = h_2(t_{2j}^x) = \cdots = h_{2j}(t_{2j}^x) = p_2'(d_j[2j])$,

where

$$d_r[i] = \begin{cases} i(i+1)/2 & \text{for } 1 \le i \le r+1, \\ i(4r-i+3)/2 - r(r+1) & \text{for } r+1 < i \le 2r. \end{cases}$$

(See Fig. 4.1.)

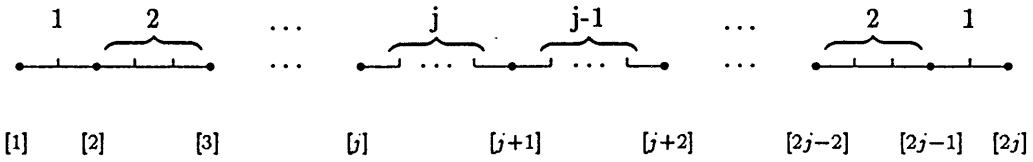

Fig. 4.1. '•' and 'ı' denote $j(j+2)$ 1's in the first row of x, and '[i]' denotes the position $p_1(d_j[i])$ for each $1 \le i \le 2j$.

Inductive Step: We show that there exists a CS-TR2-DFA($2(k+1)$) $M(2(k+1)) = (A_1, A_2, \cdots, A_{2k+2})$ accepting $T'(k^2+4k+3)$ which satisfies the property that for each $x \in T'(k^2+4k+3)$, there are $2k+3$ times $t_0^x < t_1^x < \cdots < t_{2k+2}^x$ during the accepting computation of $M(2(k+1))$ on x such that

[0.] $h_1(t_0^x) = h_2(t_0^x) \cdots = h_{2k+2}(t_0^x) = p_1(d_{k+1}[1])$,

[1.] $h_1(t_1^x) = p_2'(d_{k+1}[1])$, $h_2(t_1^x - 1) = h_3(t_1^x - 1) = \cdots = h_{2k+2}(t_1^x - 1) = p_1(d_{k+1}[1])$,

$\vdots$

[i.] $h_1(t_i^x) = h_2(t_i^x) = \cdots = h_i(t_i^x) = p_2'(d_{k+1}[i])$,

$h_{i+1}(t_i^x - 1) = h_{i+2}(t_i^x - 1) = \cdots = h_{2k+2}(t_i^x - 1) = p_1(d_{k+1}[i])$,

$\vdots$

[2k+2.] $h_1(t_{2k+2}^x) = h_2(t_{2k+2}^x) = \cdots = h_{2k+2}(t_{2k+2}^x) = p_2'(d_{k+1}[2k+2])$.

Consider the case when an input tape x with 2 rows and m columns such that $x[(1,1),(1,m)]$, $x[(2,1),(2,m)] \in R_{(k^2+4k+3)}(m)$ is presented to $M(2(k+1))$ which acts as follows. (Input tapes in the form different from the above can easily be rejected by $M(2(k+1))$.)

1. $M(2(k+1))$ first verifies in a similar way as $M(2k)$ that the symbol on $p_2'(i)$ is 1 for each $i \in \{1, 2, \cdots, d_{k+1}[k+1]\}$. That is, by using the corresponding $2k$ automata A_1, A_2, $\cdots$, A_{2k}, $M(2(k+1))$ simulates the action of $M(2k)$ for checking whether the symbol just under the i-th 1 in the first row is 1 when an input tape y with 2 rows and m columns such that $y[(1,1),(1,m)]$, $y[(2,1),(2,m)] \in R_{(k^2+2k)}(m)$ is presented to $M(2k)$, while FA_{2k+1} and FA_{2k+2} are idle, and move in the same way as FA_{2k} dose. If this verification is successful, then goto 2. Otherwise $M(2(k+1))$ rejects x.

2. (a) Inductive hypothesis implies that h_1, h_2, $\cdots$, h_{k+1} simultaneously reach $p_2'(d_{k+1}[k+1])$ at some time t_{k+1}^x, and h_{k+2}, h_{k+3}, $\cdots$, h_{2k+2} reach $p_1(d_{k+1}[k+1])$ at time $t_{k+1}^x - 1$. Then A_{k+1} moves h_{k+1} to the right from $p_2'(d_{k+1}[k+1])$ to the position (denoted by $p_2(d_{k+1}[k+1]+1)$) of the next symbol 1 at speed $1/(n^k+n^{k-1}+\cdots+n+1)$, from $p_2(d_{k+1}[k+1]+1)$ to the position (denoted by $p_2(d_{k+1}[k+1]+2)$) of the next symbol 1 at speed $1/(n^{k-1}+n^{k-2}+\cdots+n+1)$, $\cdots$, from $p_2(d_{k+1}[k+1]+k)$ to the position (denoted by $p_2(d_{k+1}[k+1]+k+1)$) of the next symbol 1 at speed 1, and from $p_2(d_{k+1}[k+1]+k+1)$ to the position (denoted by $p_2(d_{k+1}[k+1]+k+2) = p_2(d_{k+1}[k+2])$) of the next symbol 1 at speed $1/2$ if there really exist at least $k+2$ 1's to the right hand of $p_2'(d_{k+1}[k+1])$, where $n = 2(m+2)$. Moreover, make h_1, h_2, $\cdots$, h_{k+1} reach $p_2(d_{k+1}[k+2])$ at the same time if it really exists. For otherwise, $M(2(k+1))$ rejects x. FA_{k+1} may do this by means of A_1, A_2, $\cdots$, A_k.

Note that A_{k+1} can move h_{k+1} at speed $1/(n^i + n^{i-1} + \cdots + n + 1)$ for any $i \leq k$ by means of A_1, A_2, $\cdots$, A_k. For example, suppose that h_1 and h_{k+1} are positioned on the same cell, $x(2, j)$, and A_{k+1} moves right one cell together with A_1 just when h_1 meets h_{k+1} again after A_1 moved h_1 one cycle of the second row ($2(m+2)$ cells) at speed 1. This makes h_{k+1} move at speed $1/(n+1)$ (on $x(2, j)$). Similarly, by means of A_1 and A_2 which are on the same cell scanned by A_{k+1}, A_{k+1} can move (together with A_1 and A_2) at speed $1/((n+1)n+1) = 1/(n^2+n+1)$ and so on.

(b) In a similar way as A_{k+1} dose, A_{k+2} moves right h_{k+2} from $p_1(d_{k+1}[k+1])$ to $p_1(d_{k+1}[k+1]+1)$ at speed $1/(n^k + n^{k-1} + \cdots + n + 1)$, from $p_1(d_{k+1}[k+1]+1)$ to $p_1(d_{k+1}[k+1]+2)$ at speed $1/(n^{k-1} + n^{k-2} + \cdots + n + 1)$, $\cdots$, from $p_1(d_{k+1}[k+1]+k)$ to $p_1(d_{k+1}[k+1]+k+1)$ at speed 1, and from $p_1(d_{k+1}[k+1]+k+1)$ to $p_1(d_{k+1}[k+1]+k+2) = p_1(d_{k+1}[k+2])$ at speed $1/2$, by means of A_{k+3}, A_{k+4}, $\cdots$, A_{2k+2}. Moreover, make h_{k+2}, h_{k+3}, $\cdots$, h_{2k+2} reach $p_1(d_{k+1}[k+2])$ simultaneously. Then FA_{k+2} moves h_{k+2} down one cell at speed 1.

3. The time at which h_{k+2} reaches $p_2'(d_{k+1}[k+2])$ is denoted by t^x_{k+2}. $M(2(k+1))$ continues the computation on x if h_1, h_2, $\cdots$, h_{k+2} reach $p_2(d_{k+1}[k+2])$ at time t^x_{k+2} (i.e., $p_2'(d_{k+1}[k+2]) = p_2(d_{k+1}[k+2])$), that is, if $h_1(t^x_{k+2}) = h_2(t^x_{k+2}) = \cdots = h_{k+2}(t^x_{k+2}) = p_2'(d_{k+1}[k+2])$, and $h_{k+3}(t^x_{k+2}-1) = h_{k+4}(t^x_{k+2}-1) = \cdots = h_{2k+2}(t^x_{k+2}-1) = p_1(d_{k+1}[k+2])$. Otherwise $M(2)$ rejects x.

Note that h_1, h_2, $\cdots$, h_{k+2} reach $p_2(d_{k+1}[k+2])$ at time t^x_{k+2} if and only if

(a) $l_{d_{k+1}[k+1]} + l_{d_{k+1}[k+1]+1} + \cdots + l_{d_{k+1}[k+1]+k+1}$
$= l'_{d_{k+1}[k+1]} + l'_{d_{k+1}[k+1]+1} + \cdots + l'_{d_{k+1}[k+1]+k+1}$, and

(b) $l_{d_{k+1}[k+1]} \cdot n^k +$
$(l_{d_{k+1}[k+1]} + l_{d_{k+1}[k+1]+1}) \cdot n^{k-1} + \cdots +$
$(l_{d_{k+1}[k+1]} + l_{d_{k+1}[k+1]+1} + \cdots + l_{d_{k+1}[k+1]+k-1}) \cdot n +$
$l_{d_{k+1}[k+1]} + l_{d_{k+1}[k+1]+1} + \cdots + l_{d_{k+1}[k+1]+k} + 2l_{d_{k+1}[k+1]+k+1}$

$$= l'_{d_{k+1}[k+1]} \cdot n^k +$$
$$(l'_{d_{k+1}[k+1]} + l'_{d_{k+1}[k+1]+1}) \cdot n^{k-1} + \cdots +$$
$$(l'_{d_{k+1}[k+1]} + l'_{d_{k+1}[k+1]+1} + \cdots + l'_{d_{k+1}[k+1]+k-1}) \cdot n +$$
$$l'_{d_{k+1}[k+1]} + l'_{d_{k+1}[k+1]+1} + \cdots + l'_{d_{k+1}[k+1]+k} + 2l'_{d_{k+1}[k+1]+k+1},$$

where l_i denotes the distance from $p_1(i)$ to $p_1(i+1)$, $l'_{d_{k+1}[k+1]}$ denotes the distance from $p'_2(d_{k+1}[k+1])$ to the $p_2(d_{k+1}[k+1]+1)$, and $l'_{d_{k+1}[k+1]+i}$ denotes the distance from $p_2(d_{k+1}[k+1]+i)$ to the $p_2(d_{k+1}[k+1]+i+1)$. That is,

$$l_{d_{k+1}[k+1]} = l'_{d_{k+1}[k+1]},\ l_{d_{k+1}[k+1]+1} = l'_{d_{k+1}[k+1]+1},\ \cdots\ l_{d_{k+1}[k+1]+k+1} = l'_{d_{k+1}[k+1]+k+1}.$$

This means that the symbol on $p'_2(i)$ is 1 for each $i \in \{d_{k+1}[k+1]+1, d_{k+1}[k+1]+2, \cdots, d_{k+1}[k+2]\}$.

4. $M(2(k+1))$ finally verifies in a similar way as $M(2k)$ dose that the symbol on $p'_2(i)$ is 1 for each $i \in \{d_{k+1}[k+2]+1, d_{k+1}[k+2]+2, \cdots, d_{k+1}[2k+2]\}$. That is, by using A_1, A_2, $\cdots$, A_k, A_{k+3}, A_{k+4}, $\cdots$, A_{2k+2} corresponding to the $2k$ automata of $M(2k)$, $M(2(k+1))$ simulates the action of $M(2k)$ for checking whether the symbol under the j-th 1 in the first row is 1 for each $j \in \{d_k[k]+1, d_k[k]+2, \cdots, d_k[2k]\}$ when an input tape y with 2 rows and m columns such that $y[(1,1),(1,m)]$, $y[(2,1),(2,m)] \in R_{(k^2+2k)}(m)$ is presented to $M(2k)$, while A_{k+1} and A_{k+2} are idle, and move in the same way as A_k dose. If this is the case, $M(2(k+1))$ accepts x. Otherwise $M(2(k+1))$ rejects x.

It is easy to see that by inductive hypothesis the action of $M(2(k+1))$ described above satisfies the required property. This completes the proof of (2).

We now prove (3). Suppose that there is a CS-TR2-DFA$(k)^S$ $M(k) = (A_1, A_2, \cdots, A_k)$ accepting $T(k(k+1)/2+1)$. Let h_i denote the input head of FA$_i$ for each $i \in \{1, 2, \cdots, k\}$.

For each $m \geq k(k+1)/2+1$, let $V(m) = \{x \in T(k(k+1)/2+1) | l_1(x) = l_2(x) = m\}$, and for each permutation $\sigma : \{1, 2, \cdots, k\} \to \{1, 2, \cdots, k\}$, let $W_\sigma(m)$ be the set of all input tapes $x \in V(m)$ such that during the accepting computation of $M(k)$ on x, input heads $h_{\sigma(1)}$, $h_{\sigma(2)}$,

$\cdots$, $h_{\sigma(k)}$ leave the first row of x in this order.[1] Then there must exist some permutation τ such that

$$|W_\tau(m)| \geq |V(m)|/k! = \Omega(m^{k(k+1)/2+1}).$$

For each $x \in W_\tau(m)$ and each $1 \leq i \leq k$, let $q_{\tau(i)}(x)$, $p_{\tau(i)}(x)$ and $t_{\tau(i)}(x)$ denote the internal state of $\mathrm{FA}_{\tau(i)}$, the position of $h_{\tau(i)}$ and the time, respectively, when $h_{\tau(i)}$ leaves the first row during the accepting computation of $M(k)$ on x. Furthermore, let

$$t(x) = (t_{\tau(2)}(x) - t_{\tau(1)}(x), t_{\tau(3)}(x) - t_{\tau(2)}(x), \cdots, t_{\tau(k-1)}(x) - t_{\tau(k-2)}(x)),$$

and

$$u(x) = \langle (q_{\tau(1)}(x), p_{\tau(1)}(x)), \cdots, (q_{\tau(k)}(x), p_{\tau(k)}(x)), t(x) \rangle.$$

Clearly, for each $2 \leq i \leq k$, $t_{\tau(i)}(x) - t_{\tau(i-1)}(x) = O(m^{k-i})$, because for otherwise $A_{\tau(i)}$, $\cdots$, $A_{\tau(k)}$ would loop forever on the first row, and thus $M(k)$ would never accept x. So

$$| \{u(x) \mid x \in W_\tau(m)\} | = O(m^{k(k+1)/2}).$$

Therefore, it follows that for large m,

$$| W_\tau(m) | > | \{u(x) \mid x \in W_\tau(m)\} |$$

and there exist two different input tapes $x, y \in W_\tau(m)$ such that $u(x) = u(y)$. Let z be the tape obtained from x by replacing the second row of x with the second row of y. It follows that z is also accepted by $M(k)$. This is a contradiction, because z is not in $T(k(k+1)/2+1)$. This completes the proof of (3). □

Lemma 4.1. For each $k \geq 1$,

(1) if x is a tape with m rows and m columns accepted by a CS-TR2-DFA(k) M, then M accepts x in at most $O(m^{k+1})$ steps, and

[1] If $h_{i_1}, h_{i_2}, \cdots, h_{i_r}$ $(1 \leq i_1 < i_2 < \cdots < i_r \leq k)$ leave the first row simultaneously, we refer the order on them as $\langle i_1, i_2, \cdots, i_r \rangle$.

(2) if x is a tape with m rows and m columns accepted by a CS-TR2-FA(k) M, then there exists a computation of M on x such that M accepts in at most $O(m^{k+1})$ steps.

Proof : (1) Consider the accepting computation of M on x. It is obvious that each finite-state automaton of M can stay each row of x at most $O(m^k)$. (For otherwise M would enter a loop and never accept x). Thus the accepting time is at most $O(m^{k+1})$.

(2) Consider a shortest accepting computation of M on x (in which no loop exists). We can observe that for every cm^k steps, where c is some constant depending only on M, there is at least one finite-state automaton of M moves down from the present row. (For otherwise M would enter a loop). Thus, the accepting computation consists of at most $O(m^{k+1})$ steps. □

Theorem 4.5. For each $k \geq 1$, $f(k^2+k) \geq k+1$.

Proof : The proof is very similar to that of Theorem 4.4 (3). Suppose, to the contrary, that there is a CS-TR2-FA(k)^S $M(k) = (A_1, A_2, \cdots, A_k)$ accepting $T(k^2+k)$. Let h_i denote the input head of A_i for each $i \in \{1, 2, \cdots, k\}$.

For each $m \geq k^2+k$, let $V(m) = \{x \in T(k^2+k) | l_1(x) = l_2(x) = m\}$. With each $x \in V(m)$, We associate a shortest accepting computation, $c(x)$, of $M(k)$ on x (in which no loop exists). Furthermore, let $W_\tau(m)$, $t(x)$ and $u(x)$ (for each $x \in W_\tau(m)$) have the same meanings as in the proof of Theorem 4.4 (3).

By Lemma 4.1 (2),

$$t_{\tau(i)}(x) - t_{\tau(i-1)}(x) = O(m^{k+1}),$$

for each $2 \leq i \leq k$.

So $| \{u(x) | x \in W_\tau(m)\} | = O(m^{k^2+k-1})$. Therefore, it follows that for large m,

$$| W_\tau(m) | > | \{u(x) | x \in W_\tau(m)\} |$$

and there exist two different input tapes $x, y \in W_\tau(m)$ such that $u(x) = u(y)$. Let z be the tape obtained from x by replacing the second row of x with the second row of y. Clearly, from $c(x)$

and $c(y)$, we can construct an accepting computation of $M(k)$ on z. This is a contradiction, because z is not in $T(k^2+k)$. □

Combining Theorem 4.4 with Theorem 4.5, we can get the following theorem.

Theorem 4.6. For each $k \geq 1$,

(1) $\mathcal{L}$[CS-TR2-FA$(k)^S$]$\subsetneq\mathcal{L}$[CS-TR2-FA$(k+1)^S$], and

(2) $\mathcal{L}$[CS-TR2-DFA$(k)^S$]$\subsetneq\mathcal{L}$[CS-TR2-DFA$(k+1)^S$].

Proof : We only prove (1), since (2) can be proved in the same way. For each $k \geq 1$, let $N(k) = \max\{r \mid f(r) = k\}$. Since $f(k^2+k) \geq k+1$ (by Theorem 4.5), we have $N(k) \leq k^2+k-1$.

Let M be a CS-TR2-FA$(k)^S$ accepting $T(N(k))$. From M, we can easily construct a CS-TR2-FA$(k+1)^S$ M' which accepts $T(N(k)+1)$. Thus, $T(N(k)+1) \in \mathcal{L}$[CS-TR2-FA$(k+1)^S$]. From this fact and the definition of $N(k)$, it follows that $T(N(k)+1) \in \mathcal{L}$[CS-TR2-FA$(k+1)^S$]$-\mathcal{L}$[CS-TR2-XFA$(k)^S$]. This completes the proof. □

Remark. It is easy to see that $f(1) = g(1) = 1$, $f(2) = g(2) = f(3) = g(3) = 2$, $g(4) = g(5) = 3$; but it is unknown whether or not $f(r) < g(r)$ for $n \geq 4$.

4.5 Comparisons with Other Types of Acceptors

In Chapter 3, we have shown some relationships between cooperating systems of one-way finite-state automata and other automata, such as (simple) multihead finite automata and (time- or space-bounded) one-way multicounter machines. Since a cooperating system of three-way two-dimensional finite-state automata can be considered as an extension of a cooperating system of one-way finite-state automata to two dimensions, some results obtained in the one-dimensional case, such as (1) $\bigcup_{1\leq k<\infty}\mathcal{L}$[CS-1DFA$(k)$] and $\bigcup_{1\leq k<\infty}\mathcal{L}$[1SHA$(k)$] ($\bigcup_{1\leq k<\infty}\mathcal{L}$[1SDHA$(k)$]) are incomparable, and (2) $\bigcup_{1\leq k<\infty}\mathcal{L}$[CS-1FA$(k)$] ($\bigcup_{1\leq k<\infty}\mathcal{L}$[CS-1DFA$(k)$])$\subsetneq\bigcup_{1\leq k<\infty}\mathcal{L}$[1SeSHA$(k)$] ($\bigcup_{1\leq k<\infty}\mathcal{L}$[1SeSDHA$(k)$]), clearly hold for the two-dimensional case (if the input tapes are ar-

bitrary). A different situation, however, occurs when we restrict the input tapes to square ones.

We will be concerned only with the three-way case, since for the four-way case, the similar results as for the one-dimensional two-way case can be easily obtained.

A two-dimensional (simple) k-head finite automaton M is a finite automaton with k read-only input heads operating on a two-dimensional input tape surrounded by boundary symbols #. The action of M is very similar to that of one-dimensional two-way (simple) k-head finite automaton except that the input head of M can move up, down, right or left.

A two-dimensional sensing (simple) k-head finite automaton, is the same device as a two-dimensional (simple) k-head finite automaton except that the former can detect coincidence of the input heads.

A two-dimensional k-counter machine M has a read-only two-dimensional input tape with boundary symbols # and k counters. (Of course, M has a finite control, an input head, and k counter heads.) The action of M is very similar to that of the one-dimensional two-way k-counter machine except that the input head of M can move up, down, right or left.

A two-dimensional k-counter machine M whose input tapes are restricted to square ones is said to be $L(m)$ time-bounded if for each input w tape with m rows and m columns that is accepted by M, there is an accepting computation of M on w which consists of at most $L(m)$ steps; M is said to be $L(m)$ space-bounded if for each input tape w with m rows and m columns that is accepted by M, there is an accepting computation of M on w in which each counter of M requires space not exceeding $L(m)$.

As usual, we define the deterministic and three-way versions of two-dimensional (sensing) (simple) multihead finite automata and two-dimensional multicounter machines.

The following abbreviations will be used later.

- TR2-HA(k) : a three-way two-dimensional k-head (nondeterministic) finite automaton,
- TR2-DHA(k) : a three-way two-dimensional k-head deterministic finite automaton,

- TR2-SeHA(k) : a three-way two-dimensional k-head sensing (nondeterministic) finite automaton,
- TR2-SeDHA(k) : a three-way two-dimensional k-head sensing deterministic finite automaton,
- TR2-SHA(k) : a three-way two-dimensional k-head simple (nondeterministic) finite automaton,
- TR2-SDHA(k) : a three-way two-dimensional k-head simple deterministic finite automaton,
- TR2-SeSHA(k) : a three-way two-dimensional k-head sensing simple (nondeterministic) finite automaton,
- TR2-SeSDHA(k) : a three-way two-dimensional k-head sensing simple deterministic finite automaton,
- TR2-CM(k)^S-Time($L(m)$) : a three-way two-dimensional (nondeterministic) k-counter machine with $L(m)$ time bound whose input tapes are restricted to square ones,
- TR2-DCM(k)^S-Time($L(m)$) : a three-way two-dimensional deterministic k-counter machine with $L(m)$ time bound whose input tapes are restricted to square ones,
- TR2-CM(k)^S-Space($L(m)$) : a three-way two-dimensional (nondeterministic) k-counter machine with $L(m)$ space bound whose input tapes are restricted to square ones,
- TR2-DCM(k)^S-Space($L(m)$) : a three-way two-dimensional deterministic k-counter machine with $L(m)$ space bound whose input tapes are restricted to square ones.

Lemma 4.2. For any $k \geq 1$,

(1) $\mathcal{L}$[TR2-SeSHA(k)^S]$\subseteq\mathcal{L}$[CS-TR2-FA($2k$)^S], and

(2) $\mathcal{L}$[TR2-SeSDHA$(k)^S$]$\subseteq\mathcal{L}$[CS-TR2-DFA$(2k)^S$].

Proof : Let M be a TR2-SeSHA$(k)^S$ (TR2-SeSDHA$(k)^S$). We will construct a CS-TR2-FA$(2k)^S$ (CS-TR2-DFA$(2k)^S$) M' to simulate M. M' acts as follows:

1. M' simulates the moves of the reading head of M and all the left or right moves of counting heads of M by using its $(k+1)$ finite automata.

2. M' simulates all the down moves of counting heads of M by making the right moves of input heads of its other $(k-1)$ finite automata.

3. During the simulation, if M moves its reading head down, then M' must make all the input heads of finite automata of M' move down so that all the automata of M' can keep their input heads on the same row and can communicate with each other in that row.

It is easy to see that M' can simulate M. □

Lemma 4.3. For any $k \geq 1$,

(1) $\mathcal{L}$[CS-TR2-FA$(k)^S$]$\subseteq\mathcal{L}$[TR2-SeSHA$(k^2+k)^S$],

(2) $\mathcal{L}$[CS-TR2-DFA$(k)^S$]$\subseteq\mathcal{L}$[TR2-SeSDHA$(k^2+k)^S$],

(3) $\mathcal{L}$[CS-TR2-FA$(k)^S$]$\subseteq\mathcal{L}$[TR2-CM$(2k-1)^S$-Time(m^{k+1})],

(4) $\mathcal{L}$[CS-TR2-DFA$(k)^S$]$\subseteq\mathcal{L}$[TR2-DCM$(2k-1)^S$-Time(m^{k+1})].

Proof : (1), (2): Let $M=(A_1, A_2, \cdots, A_k)$ be a CS-TR2-FA$(k)^S$ (CS-TR2-DFA$(k)^S$). We will construct a TR2-SeSHA$(k^2+k)^S$ (TR2-SeSDHA$(k^2+k)^S$) M' to simulate M. Let r denote the reading head of M', and $h_1, h_2, \cdots, h_{k^2+k-1}$ denote the k^2+k-1 counting heads of M'. M' acts as follows:

1. In its finite control M' keeps track of what states A_1, A_2, $\cdots$, A_k are in when leaving the row of input tape scanned by M'.

2. For each row of the input tape:

   (a) M' simulates the left or right moves of input heads of A_1, A_2, $\cdots$, A_k by using r and h_1, h_2, $\cdots$, h_k.

   (b) M' stores in its finite control the internal state of each A_i when A_i leaves the row, and the order $\langle d_1, d_2, \cdots, d_k \rangle$ in which A_1, A_2, $\cdots$, A_k leave the row subsequently (i.e., A_{d_1} firstly leaves the row, A_{d_2} secondly leaves the row, and so on).[2] Moreover, M' keeps the horizontal position where A_i leaves the row, by using $h_1, h_2, \cdots, h_k$.

   (c) Furthermore, for each $1 \leq i \leq k-1$, the interval between the times at which A_{d_i} and $A_{d_{i+1}}$ leave the row is stored by a counter with $O(m^{k+1})$ space bound, which can be realized by using other $k+1$ counting heads of M', where m is the number of rows (or columns) of the input tape. This requires $k^2 - 1$ counting heads (h_{k+1}, h_{k+2}, $\cdots$, h_{k^2+k-1}) in total.

Note that M works in $O(m^{k+1})$ time (by Lemma 4.1), that is, if an input tape with m rows (or columns) is accepted by M, then it can be accepted by M within $O(m^{k+1})$ steps. Thus, it is easy to verify that M' can simulate M.

(3), (4): We leave it to the reader. (See the corresponding proof for the one-dimensional one-way case.) □

It was shown in [58] that

(1) $\bigcup_{1 \leq k < \infty} \mathcal{L}[\text{TR2-SHA}(k)^S]$
$= \bigcup_{1 \leq k < \infty} \mathcal{L}[\text{TR2-SeSHA}(k)^S]$
$= \bigcup_{1 \leq k < \infty} \mathcal{L}[\text{TR2-CM}(k)^S\text{-Space}(m)]$ and,

(2) $\bigcup_{1 \leq k < \infty} \mathcal{L}[\text{TR2-SDHA}(k)^S]$
$= \bigcup_{1 \leq k < \infty} \mathcal{L}[\text{TR2-SeSDHA}(k)^S]$
$= \bigcup_{1 \leq k < \infty} \mathcal{L}[\text{TR2-DCM}(k)^S\text{-Space}(m)]$.

[2] If A_{i_1}, A_{i_2}, $\cdots$, A_{i_r} $(1 \leq i_1 < i_2 < \cdots < i_r \leq k)$ leave the row simultaneously, we refer the order on them as $\langle i_1, i_2, \cdots, i_r \rangle$.

Combining this result with Lemmas 4.2 and 4.3, we have the following theorem.

Theorem 4.7.

(1) $\bigcup_{1\leq k<\infty}\mathcal{L}$[CS-TR2-FA$(k)^S$]
$= \bigcup_{1\leq k<\infty}\mathcal{L}$[TR2-SHA$(k)^S$]
$= \bigcup_{1\leq k<\infty}\mathcal{L}$[TR2-SeSHA$(k)^S$]
$= \bigcup_{1\leq k<\infty}\mathcal{L}$[TR2-CM$(k)^S$-Space$(m)$] and,

(2) $\bigcup_{1\leq k<\infty}\mathcal{L}$[CS-TR2-DFA$(k)^S$]
$= \bigcup_{1\leq k<\infty}\mathcal{L}$[TR2-SDHA$(k)^S$]
$= \bigcup_{1\leq k<\infty}\mathcal{L}$[TR2-SeSDHA$(k)^S$]
$= \bigcup_{1\leq k<\infty}\mathcal{L}$[TR2-DCM$(k)^S$-Space$(m)$].

A long-standing unsolved problem concerning two-dimensional finite-state automata [26] is whether or not a two-dimensional finite-state automata can accept the set of connected patterns, where a pattern on a tape $x \in \{0,1\}^{(2)+}$ is the set of all cells of x which have symbol 1's, two cells of a pattern are adjacent if they share a common edge (not just a common vertex), and a pattern is connected if and only if any two cells of it are joined by a string of adjacent cells on it. It is easily seen that there is a CS-2-DFA(2) (in fact, a two-dimensional deterministic finite-state automaton with one marker [26]) which can accept the set of connected patterns. On the other hand, we have

Corollary 4.3. For any $k \geq 1$, CS-TR2-FA(k)'s cannot accept the set of connected patterns.

Proof : It is shown in [60] that the set of connected patterns is not in $\bigcup_{1\leq k<\infty}\mathcal{L}$[TR2-SeSHA$(k)$]. From this result and Theorem 4.7, the corollary follows. □

4.6 Closure Properties

This section investigates closure properties of the classes of languages accepted by CS-TR2-FA$^S(k)$'s under several two-dimensional language operations.

We first examine closure properties under Boolean operations. It was shown in [61] that $\bigcup_{1\leq k<\infty}\mathcal{L}$[TR2-CM($k$)^S-Space($m$)] is closed under union and intersection, but not under complementation; $\bigcup_{1\leq k<\infty}\mathcal{L}$[TR2-DCM($k$)^S-Space($m$)] is closed under union, intersection, and complementation. From this fact and Theorem 4.7, we get the following theorem.

Theorem 4.8 (1) $\bigcup_{1\leq k<\infty}\mathcal{L}$[CS-TR2-FA($k$)^S] is closed under union and intersection, but not under complementation, and (2) $\bigcup_{1\leq k<\infty}\mathcal{L}$[CS-TR2-DFA($k$)^S] is closed under union, intersection, and complementation.

Lemma 4.5 For each $k \geq 1$ and each $1 \leq i \leq k$, let $C(k,i)$ be the set of all square tapes $x \in \{0,1\}^{(2)}$ such that

(i) x has at least k rows and columns,

(ii) there are exactly k 1's in the first row,

(iii) there is exactly one 1 in in the second row, and the symbol (in the second row) just below the i-th 1 (counting from left-to-right) in the first row is 1,

(iv) all other rows contain only 0.

For each $k \geq 1$, let $C(k) = C(k,1) \cup C(k,2) \cup \cdots \cup C(k,k)$. Then, for eack $k \geq 1$, each $1 \leq i \leq k$,

(1) $C(k,i) \in \mathcal{L}$[CS-TR2-DFA(1)^S],

(2) $C(2k^2) \notin \mathcal{L}$[CS-TR2-DFA($k$)^S].

Proof : The proof of (1) is obvious, and (2) follows from Lemma 4.3 and Lemma 6.2 (2) in [62], where it was shown that $C(2k^2) \notin \mathcal{L}$[TR2-DCM($2(k-1)$)^S-Space($m^{k+1}$)]. □

Lemma 4.6 For each $k \geq 1$ and each $1 \leq i \leq k$, let $T(k,i)$ be the set of all square tapes $x \in \{0,1\}^{(2)}$ such that

(i) x has at least k rows and columns,

(ii) there are exactly k 1's in the first and second rows, and the symbol (in the second row) just below the i-th 1 (counting from left to right) in the first row is 1,

(iii) all other rows contain only 0.

For each $k \geq 1$, let $T(k) = T(k,1) \cap T(k,2) \cap \cdots \cap T(k,k)$. (Note that $T(k)$ is identical with the set $T(k)$ described in Section 4.3.) Then, for eack $k \geq 1$, each $1 \leq i \leq k$,

(1) $T(k,i) \in \mathcal{L}$[CS-TR2-DFA(1)^S],

(2) $T(k^2+k) \notin \mathcal{L}$[CS-TR2-FA($k$)^S].

Proof : The proof is obvious, and (2) follows from Theorem 4.5. □

Now we can get the following theorem.

Theorem 4.9 For each $k \geq 1$, (1) $\mathcal{L}$[CS-TR2-DFA(k)^S] is not closed under union and intersection, and (2) $\mathcal{L}$[CS-TR2-FA(k)^S] is closed under union, but not closed under intersection and complementation.

Proof : (1) follows from Lemmas 4.5 and 4.6. It also follows from Lemma 4.6 that $\mathcal{L}$[CS-TR2-FA(k)^S] is not closed under intersection, while closure under union for $\mathcal{L}$[CS-TR2-FA(k)^S] is obvious.

Let T_1 be the set described in Theorem 4.1. Consider the complement, $\overline{T_1}$, of T_1. Clearly, $\overline{T_1}$ is in $\mathcal{L}$[CS-TR2-FA(1)^S]. On the other hand, T_1 is not in $\bigcup_{1 \leq k < \infty} \mathcal{L}$[CS-TR2-FA($k$)^S] (see the proof of Theorem 4.1). This completes the proof. □

We next investigate closure properties under rotation, row reflection, row and column cyclic closures, and projection.

Definition. If

$$x = \begin{matrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn}, \end{matrix}$$

the rotation x^R of x is given by

$$x = \begin{matrix} a_{m1} & \cdots & a_{11} \\ \vdots & \ddots & \vdots \\ a_{mn} & \cdots & a_{1n}, \end{matrix}$$

the row reflection x^{RR} of x is given by

$$x = \begin{matrix} a_{m1} & \cdots & a_{mn} \\ \vdots & \ddots & \vdots \\ a_{11} & \cdots & a_{1n}, \end{matrix}$$

a row cyclic shift of x is any two-dimensional tape of the form

$$x = \begin{matrix} a_{k+1,1} & \cdots & a_{k+1,n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \\ a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{k1} & \cdots & a_{kn}, \end{matrix}$$

for some $1 \leq k \leq m$, and a column cyclic shift of x is any two-dimensional tape of the form

$$x = \begin{matrix} a_{1,k+1} & \cdots & a_{1n} & a_{11} & \cdots & a_{1k} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_{m,k+1} & \cdots & a_{mn} & a_{m1} & \cdots & a_{mk}, \end{matrix}$$

for some $1 \leq k \leq n$.

Definition. Let T be a set of two-dimensional tapes. Then

- $T^R = \{x^R | x \in T\}$ (rotation of T),
- $T^{RR} = \{x^{RR} | x \in T\}$ (row reflection of T),
- $T^{RC} = \{y | y$ is a row cyclic shift of some $x \in T\}$ (row cyclic closure of T),
- $T^{CC} = \{y | y$ is a column cyclic shift of some $x \in T\}$ (column cyclic closure of T).

Definition. Let Σ_1, Σ_2 be finite sets of symbols. A projection is a mapping $\overline{\tau} : \Sigma_1^{(2)} \rightarrow \Sigma_2^{(2)}$ which is obtained by extending a mapping $\tau : \Sigma_1 \rightarrow \Sigma_2$ as follows: $\overline{\tau}(x) = x'$ if and only if

(i) $l_k(x) = l_k(x')$ for each $k = 1, 2$, and

(ii) $\tau(x(i,j)) = x'(i,j)$ for each $1 \leq i \leq l_1(x)$ and each $1 \leq j \leq l_2(x)$.

If $T \in \Sigma^{(2)}$, we let $\overline{\tau}(T) = \{\overline{\tau}(x) | x \in T\}$.

Theorem 4.10 (1) For each $k \geq 1$, neither $\mathcal{L}$[CS-TR2-FA$(k)^S$] nor $\mathcal{L}$[CS-TR2-DFA$(k)^S$] is closed under rotation. (2) Neither $\bigcup_{1 \leq k < \infty} \mathcal{L}$[CS-TR2-FA$(k)^S$] nor $\bigcup_{1 \leq k < \infty} \mathcal{L}$[CS-TR2-DFA$(k)^S$] is closed under rotation.

Proof : Let $T_3 = \{x \in \{0,1\}^{(2)} | \exists m \geq 2[l_1(x) = l_2(x)$ & $x[(1,1),(m,1)] = x[(1,2),(m,2)]]\}$. Clearly, T_3 is in $\mathcal{L}$[CS-TR2-DFA$(k)^S$]. On the other hand, $T_3^R = T_1$, which is described in Theorem 4.1, is not in $\bigcup_{1 \leq k < \infty} \mathcal{L}$[CS-TR2-FA$(k)^S$] (see the proof of Theorem 4.1). This completes the proof. □

Let T_4 be the set of all square tapes $x \in \{0,1\}^{(2)}$ such that

(i) x has m (≥ 2) rows and columns,

(ii) there is exactly one 1 in the first row,

(iii) there is a j, $1 \leq j \leq m$, such that $x(1,j) = x(m,j) = 1$.

Let T_5 be the set of all square tapes $x \in \{0,1\}^{(2)}$ such that

(i) x has m (≥ 2) rows and columns,

(ii) there is exactly one 1 in the last row,

(iii) there is a j, $1 \leq j \leq m$, such that $x(1,1) = x(m,1) = 1$.

Let T_6 be the set of all square tapes $x \in \{0,1\}^{(2)}$ such that

(i) x has m (≥ 2) rows and columns,

(ii) there is exactly one 1 in the first row,

(iii) there is a j, $1 \leq j \leq m$, such that $x(1,j) = x(2,j) = 1$.

Let T_7 be the set of all square tapes $x \in \{0,1,2\}^{(2)}$ such that

(i) x has m (≥ 2) rows and columns,

(ii) there is exactly one 1 in the first row,

(iii) there is a j, $1 \leq j \leq m$, such that $x(1,j) = x(m,j) = 2$, and for each $1 \leq i \leq m$, $i \neq j$, $x(1,i) \neq 2$ and $x(m,i) = 0$.

It was shown in [61] that $T_4, T_5, T_6, T_7 \in \mathcal{L}$[CS-TR2-DFA(1)^S], and $T_4^{RR} = T_5^{CC} = \overline{h}(T_7), T_6^{RC} \notin \bigcup_{1 \leq k < \infty} \mathcal{L}$[TR2-DCM($k$)^S-Space($m$)], where $\overline{h}$ is the projection obtained by extending the mapping $h : \{0,1,2\} \rightarrow \{0,1\}$ with $h(0) = 0$, $h(1) = 1$ and $h(2) = 1$. From this fact and Theorem 4.7, we get the following theorem.

Theorem 4.11 (1) For each $k \geq 1$, $\mathcal{L}$[CS-TR2-DFA(k)^S] is not closed under row reflection, row and column cyclic closure, or projection. (2) $\bigcup_{1 \leq k < \infty} \mathcal{L}$[CS-TR2-DFA($k$)^S] is not closed under row reflection, row and column cyclic closures, or projection, either.

Theorem 4.12 For each $k \geq 1$, $\mathcal{L}$[CS-TR2-FA(k)^S] is closed under row reflection, but not closed under column cyclic closure.

Proof: Closure under row reflection can be obtained by extending the corresponding results (i.e., closure under reversal) of the one-dimensional one-way case to the two-dimensional case, whose proof is left for the reader.

We now prove nonclosure under column cyclic closure. For eack $n \geq 1$, let $T'(n)$ be the set of all square tapes $x \in \{0,1\}^{(2)}$ such that

(i) x has m (≥ 3) rows and columns,

(ii) there are exactly $(n+1)$ 1's in the first row,

(iii) $x(1,1) = 1$,

(iv) the second row is equal to the first,

(v) all other rows contain only 0's.

Clearly, for each $k \geq 1$, $T'(N(k)+1)$ is in $\mathcal{L}$[CS-TR2-FA$(k)^S$], where $N(k)$ is defined in the proof of Theorem 4.6. On the other hand, $T'(N(k)+1)^{CC} = T(N(k)+1) \notin \mathcal{L}$[CS-TR2-FA$(k)^S$] (see the proof of Theorem 4.6). This completes the proof. □

Theorem 4.13 $\bigcup_{1 \leq k < \infty} \mathcal{L}$[CS-TR2-NFA$(k)^S$] is closed under row reflection, row and column cyclic closures.

Proof : This directly follows from Theorem 4.7 and the result (Theorem 6.6 in [59]) that $\bigcup_{1 \leq k < \infty} \mathcal{L}$[TR2-CM$(k)^S$-Space$(m)$] is closed under row reflection, row and column cyclic closures. □

The closure results obtained above are summarized in Table 4.1, where $\mathcal{L}(\mathrm{FA}^S)_k^D$ =$\mathcal{L}$[CS-TR2-DFA$(k)^S$], $\mathcal{L}(\mathrm{FA}^S)_k^N$ =$\mathcal{L}$[CS-TR2-FA$(k)^S$], $\mathcal{L}(\mathrm{FA}^S)_\infty^D = \bigcup_{1 \leq k < \infty} \mathcal{L}$[CS-TR2-DFA$(k)^S$], $\mathcal{L}(\mathrm{FA}^S)_\infty^N = \bigcup_{1 \leq k < \infty} \mathcal{L}$[CS-TR2-FA$(k)^S$], "$\surd$" means that the class is closed, "$\times$" means that the class is not closed, and "?" means that the closure property is not known.

Table 4.1. Closure properties of CS-TR2-FA$(k)^S$'s

	$\mathcal{L}(\mathrm{FA}^S)_k^D$	$\mathcal{L}(\mathrm{FA}^S)_k^N$	$\mathcal{L}(\mathrm{FA}^S)_\infty^D$	$\mathcal{L}(\mathrm{FA}^S)_\infty^N$
complementation	$\surd(k=1)$?$(k \geq 2)$	$\times$	$\surd$	$\times$
union	$\times$	$\surd$	$\surd$	$\surd$
intersection	$\times$	$\times$	$\surd$	$\surd$
rotation	$\times$	$\times$	$\times$	$\times$
row reflection	$\times$	$\surd$	$\times$	$\surd$
row cyclic closure	$\times$	$\times(k=1)$?$(k \geq 2)$	$\times$	$\surd$
column cyclic closure	$\times$	$\times$	$\times$	$\surd$
projection	$\times$	?	$\times$	$\surd$

4.7 Concluding Remarks

We have investigated some properties of cooperating systems of two-dimensional finite-state automata whose input tapes are restricted to square ones, and exhibited some features that sharply contrast them from one-dimensional languages. We conclude this chapter by giving

several open problems.

(1) $\mathcal{L}$[CS-2-FA(k)^S]$\subsetneq\mathcal{L}$[CS-2-FA($k+1$)^S] for each $k \geq 2$?

(2) $\mathcal{L}$[CS-2-DFA(k)^S]$\subsetneq\mathcal{L}$[CS-2-DFA($k+1$)^S] for each $k \geq 2$?

(3) Is $\mathcal{L}$[CS-TR2-DFA(k)^S] closed under complementation for each $k \geq 2$? (It was shown in [62] that $\mathcal{L}$[CS-TR2-DFA(k)] is closed under complementation.)

(4) Is $\mathcal{L}$[CS-TR2-FA(k)^S] closed under row cyclic closure for each $k \geq 2$? (It was shown in [63] that $\mathcal{L}$[CS-TR2-FA(1)] is not closed under row cyclic closure.) We believe that the answer is negative, but we have no proof at present.

(5) Is $\mathcal{L}$[CS-TR2-FA(k)^S] closed under projection for each $k \geq 1$?

CHAPTER 5

Cooperating Systems of Counter Machines

In the previous chapters we studied in detail cooperating systems of finite-state automata as language acceptors. It is suggested that this simple model for parallel computations seems to give us a newer hierarchy of languages than other types of automata, and to be useful in connection with computational complexity classes. All questions about cooperating systems of finite-state automata may also be asked for cooperating systems of other type of automata. In this chapter, we introduce such a new type of device, called the "cooperating system of counter machines" (CS-CM), and investigate some of its properties. We mainly concentrate on the one-way case.

5.1 Definitions and Notation

Informally, a cooperating system of k counter machines consists of k (one-)counter machines CM_1, CM_2, $\cdots$, CM_k, and a read-only input tape where these counter machines independently work step by step. Each step is assumed to require exactly one time for its completion. Those counter machines whose input heads scan the same cell of the input tape can communicate with each other, that is, every counter machine is allowed to know the internal states and the signs (positive or zero) of counters of other counter machines on the cell it is scanning at the moment. The input tape holds a string of input symbols delimited by left and right endmarkers. The system starts with each CM_i on the left endmarker in its initial state with the counter empty, and accepts the input tape if each CM_i enters an accepting state and halts with the counter

empty when reading the right endmarker. (Note that we let CS-CM's accept by final states and empty counters in this thesis. We conjecture that acceptance by final states and empty counters is equivalent to acceptance by final states alone except in the realtime for CS-CM's.)

Formally, a cooperating system of k counter machines (CS-CM(k)) is denoted by

$$M = (\mathrm{CM}_1, \mathrm{CM}_2, \cdots, \mathrm{CM}_k).$$

For each $1 \leq i \leq k$, CM_i is a counter machine defined by a 9-tuple $(\Sigma, Q_i, X_i, \delta_i, q_{0_i}, F_i, \not\!\!, \$, \phi)$, where Σ, Q_i, q_{0_i}, F_i, $\not\!\!$, $\$$, ϕ have the same meanings as in the definition of cooperating system of finite-state automata, and

- $X_i = (Q_1 \times \{0,1\} \cup \{\phi\}) \times \cdots \times (Q_{i-1} \times \{0,1\} \cup \{\phi\}) \times (Q_{i+1} \times \{0,1\} \cup \{\phi\}) \times \cdots \times (Q_k \times \{0,1\} \cup \{\phi\})$,
- δ_i is the transition function mapping $(\Sigma \cup \{\not\!\!, \$\}) \times X_i \times Q_i \times \{0,1\}$ to $2^{Q_i \times \{L,H,R\} \times \{-1,0,+1\}}$.

An input to M is any string of the form $\not\!\!x\$$ where x is a string in Σ^*. A single move of M is described as follows: Let machines CM_1, CM_2, $\cdots$, CM_k be in states q_1, q_2, $\cdots$, q_k with counter's signs c_1, c_2, $\cdots$, c_k, and scanning symbols a_1, a_2, $\cdots$, a_k (note that a_i may equal $\not\!\!$ or $\$$) at time t, respectively. If $\delta_i(a_i, (q'_1, \cdots, q'_{i-1}, q'_{i+1}, \cdots, q'_k), q_i, c_i) = \emptyset$, then CM_i has no next move (i.e., CM_i halts). If (p_i, d_i, u_i) is in $\delta_i(a_i, (q'_1, \cdots, q'_{i-1}, q'_{i+1}, \cdots, q'_k), q_i, c_i)$, then CM_i may enter state p_i, move its input head in the direction d_i (where $d_i = L, H$, or R indicates the head is to move one cell to the left, remain stationary, or move one cell to the right, respectively) and decrease the counter by 1 if $u_i = -1$, increase the counter by 1 if $u_i = +1$ or not change the counter if $u_i = 0$ at the next time $t+1$. Here for each $j \in \{1, \cdots, i-1, i+1, \cdots, k\}$,

$$q'_j = \begin{cases} (q_j, c_j) \in Q_j \times \{0,1\} & \text{if } \mathrm{CM}_i \text{ and } \mathrm{CM}_j \text{ are on the same cell at time } t, \\ \phi & \text{otherwise.} \end{cases}$$

The input $\not\!\!x\$$ is accepted by M if there is a sequence of moves that leads every CM_i to an accepting state with the counter empty when scanning the right endmarker $\$$. We assume that no CM_i can fall off either end of the input.

If for no $(a_i, (q'_1, \cdots, q'_{i-1}, q'_{i+1}, \cdots, q'_k), q_i, c_i)$ in the domain of δ_i does $\delta_i(a_i, (q'_1, \cdots, q'_{i-1}, q'_{i+1}, \cdots, q'_k), q_i, c_i)$ contain an element of the form (p_i, L, u_i), then M is said to be one-way.

If $\delta_i(a_i, (q'_1, \cdots, q'_{i-1}, q'_{i+1}, \cdots, q'_k), q_i, c_i)$ never contains more than one element, then M is deterministic.

Let $L(n)$ be a function from $\mathcal{N}$ to $\mathcal{N}$, where $\mathcal{N}$ is the set of natural numbers. M is said to be $L(n)$-time bounded if for each input w accepted by M, there is a computation of M on w which accepts in at most $L(|w|)$ steps. M is said to be $L(n)$-space bounded if for each input w accepted by M, there is a computation of M on w in which each counter requires space not exceeding $L(|w|)$.

We denote a one-way CS-CM(k), deterministic CS-CM(k) and one-way deterministic CS-CM(k) by CS-1CM(k), CS-DCM(k) and CS-1DCM(k), respectively, and for each M$\in$ {CS-CM(k), CS-DCM(k), CS-1CM(k), CS-1DCM(k)}, if M is $L(n)$-time (space) bounded, we denote it by M[Time($L(n)$)] (M[Space($L(n)$)]). Moreover, let

- $\mathcal{L}$[M[Time(poly)]]= $\bigcup_{1\leq c,s<\infty} \mathcal{L}$[M[Time($cn^s$)]],
- $\mathcal{L}$[M[Space(poly)]]= $\bigcup_{1\leq c,s<\infty} \mathcal{L}$[M[Space($cn^s$)]].

5.2 Relationship between Cooperating Systems of Counter Machines and Multicounter Machines

It is well-known that without time or space limitations, 1DCM(2)'s have the same accepting power as Turing machines [2]. Clearly, CS-1DCM(2)'s can simulate 1DCM(2)'s. Thus, CS-1DCM(2)'s (without time or space limitations) are equivalent to Turing machines in accepting power. In this chapter, we show some relationships between cooperating systems of counter machines and multicounter machines regarding the polynomial time (space) complexity.

The following theorem is straightforward.

Theorem 5.1. For each $k \geq 1$, CS-CM(k)'s (respectively, CS-DCM(k)'s, CS-1CM(k)'s,

CS-1DCM(k)'s) can simulate CM(k)'s (respectively, DCM(k)'s, 1CM(k)'s, 1DCM(k)'s) under the same time (space) bound.

From now on, for each integer $k \geq 1$, by $L(k)$ we denote the language that is defined as follows:

- $L(1) = \{0^i 2 0^i | i \geq 1\}$,
- $L(k+1) = \{0^i 1 w 1 0^i | i \geq 1 \ \& \ w \in L(k)\}$.

Furthermore, let $L'(k) = a^* L(k) a^*$.

Lemma 5.1. For each $k \geq 1$,

(1) $L'(k+1) \notin \mathcal{L}$[1CM($k$)-Space($n$)],

(2) $L'(2k-1) \in \mathcal{L}$[CS-1DCM(k)[Time($2n$)]],

(3) $L'(2k-1) \in \mathcal{L}$[CS-1DCM(k)[Space($2n$)]].

Proof : (1) follows from Lemma 2.1 of [48]. To prove (2) and (3), we construct a CS-1DCM(k)[Time(2n)] $M = (A_1, A_2, \cdots, A_k)$ which acts as follows: Consider the case when an input word

$$\not\in a^p 0^{m_1} 1 0^{m_2} 1 \cdots 1 0^{m_{2k-1}} 2 0^{m'_{2k-1}} 1 \cdots 1 0^{m'_2} 1 0^{m'_1} a^q \$$$

is presented to M. (Input words in the form different from the above can easily be rejected by M.)

(i) For each $1 \leq i \leq k-1$, A_i sweeps the subword $0^{m_{2k-i}}$ at speed 1, the subword $0^{m'_{2k-i}}$ at speed 1/3, and the other parts at speed 1/2. Moreover, when it reads the subword 0^{m_i}, A_i increases its counter by 1 for each move; and when it reads the subword $0^{m'_i}$, A_i decreases its counter by 1 for each move (unless its counter is empty).

(ii) While sweeping the input tape at the same speed 1/2, A_k increases its counter by 1 in every move when reading the subword 0^{m_k}, and decreases its counter by 1 in every move when reading the subword $0^{m'_k}$ (unless its counter is empty).

(iii) Each of A_1, A_2, $\cdots$, A_k can enter an accepting state when scanning the right endmarker if and only if for each $1 \leq i \leq k-1$, A_i and A_k scan the same cell just after A_k sweeps the subword $0^{m'_{2k-i}}$ and for each $1 \leq j \leq k$, the contents of counter of A_j becomes empty just after A_j sweeps the subword $0^{m'_j}$.

Note that for each $1 \leq i \leq k-1$, $m_{2k-i} = m'_{2k-i}$ if and only if A_i and A_k scan the same cell just after A_k sweeps the subword $0^{m'_{2k-i}}$ and that for $1 \leq j \leq k$, $m_j = m'_j$ if and only if the contents of counter of A_j becomes empty just after A_j sweeps the subword $0^{m'_j}$. Thus M accepts the input word x with time bound $2|x|$ if and only if $x \in L'(2k-1)$. Furthermore, it is obvious that M accepts $L'(2k-1)$ with space bound n. □

In addition, we have the usual speed-up results concerning multicounter machines [49] as follows.

Lemma 5.2. For each $k \geq 1$ and any $c, s > 1$,

(1) $\mathcal{L}$[CM(k)-Time(cn)] ($\mathcal{L}$[DCM(k)-Time(cn)])
$=\mathcal{L}$[CM(k)-Time(linear)] ($\mathcal{L}$[DCM(k)-Time(linear)]),

(2) $\mathcal{L}$[1CM(k)-Time(cn)] ($\mathcal{L}$[1DCM(k)-Time(cn)])
$=\mathcal{L}$[1CM(k)-Time(linear)] ($\mathcal{L}$[1DCM(k)-Time(linear)]),

(3) $\mathcal{L}$[CM(k)-Time(n^s)] ($\mathcal{L}$[DCM(k)-Time(n^s)])
$=\mathcal{L}$[CM(k)-Time(cn^s)] ($\mathcal{L}$[DCM(k)-Time(cn^s)]),

(4) $\mathcal{L}$[1CM(k)-Time(n^s)] ($\mathcal{L}$[1DCM(k)-Time(n^s)])
$=\mathcal{L}$[1CM(k)-Time(cn^s)] ($\mathcal{L}$[1DCM(k)-Time(cn^s)]),

where $\mathcal{L}$[CM(k)-Time(linear)]$= \bigcup_{1 \leq c < \infty} \mathcal{L}$[CM($k$)-Time($cn$)]], and $\mathcal{L}$[DCM(k)-Time(linear)], $\mathcal{L}$[1 CM(k)-Time(linear)], $\mathcal{L}$[1DCM(k)-Time(linear)] have similar meanings.

From Theorem 5.1 and Lemmas 5.1, 5.2, we get the following corollary.

Corollary 5.1. For each $k \geq 2$,

(1) $\mathcal{L}$[1CM(k)-Time(linear)]$\subsetneq\mathcal{L}$[CS-1CM(k)[Time($2n$)]],

(2) $\mathcal{L}$[1DCM(k)-Time(linear)]$\subsetneq\mathcal{L}$[CS-1DCM(k)[Time($2n$)]],

(3) $\mathcal{L}$[1CM(k)-Space(n)]$\subsetneq\mathcal{L}$[CS-1CM(k)[Space(n)]],

(4) $\mathcal{L}$[1DCM(k)-Space(n)]$\subsetneq\mathcal{L}$[CS-1DCM(k)[Space(n)]].

For each $s, k \geq 1$, let $f_s(k)$ ($f'_s(k)$) denote the minimum number of counters required for one-way multicounter machines with time (space) bound $O(n^s)$ to accept $L'(k)$, and $g_s(k)$ ($g'_s(k)$) denote the minimum number of counters required for one-way deterministic multicounter machines with time (space) bound $O(n^s)$ to accept $L'(k)$. Furthermore, let

$$N_s(k) = \max\{m | f_s(m) = k\},$$

$$N'_s(k) = \max\{m | f'_s(m) = k\},$$

$$D_s(k) = \max\{m | g_s(m) = k\},$$

$$D'_s(k) = \max\{m | g'_s(m) = k\}.$$

Lemma 5.3. For each $k, s \geq 1$,

(1) $D_s(k) \leq N_s(k) \leq ks$, and

(2) $D'_s(k) \leq N'_s(k) \leq ks$.

Proof : It follows from the definitions that $D_s(k) \leq N_s(k)$ and $D'_s(k) \leq N'_s(k)$ for each $k, s \geq 1$. Below we only establish by a contradiction that $N'_s(k) \leq ks$ for each $k, s \geq 1$, since $N_s(k) \leq N'_s(k)$.

Suppose that there exists some 1CM(k)-Space(n^s) M accepting $L'(ks+1)$. For each $m \geq 1$, let

$$V(m) = \{a^j 0^{i_1} 1 0^{i_2} 1 \cdots 1 0^{i_{ks+1}} 2 0^{i_{ks+1}} 1 0^{i_{ks}} 1 \cdots 1 0^{i_1} a^j \mid 1 \leq i_1, i_2, \cdots, i_{ks+1} \leq m \ \&$$
$$(j + i_1 + i_2 + \cdots + i_{ks+1}) = (ks+1)m\}.$$

Clearly, $V(m) \subseteq L'(ks+1)$ and $|V(m)| = m^{ks+1}$. With each $w \in V(m)$, we associate one fixed accepting computation, $c(w)$, of M on w which accepts in space $|w|^s$. Since the number of distinct *memory configurations* of M just after reading the symbol 2 during $c(x2x^R)$ for words $x2x^R \in V(m)$ cannot exceed $O(m^{ks})$,[1] it follows that for large m, there exist two different words $x2x^R$, $y2y^R \in V(m)$ such that the memory configuration of M just after reading 2 during $c(x2x^R)$ is the same as that of M after reading the symbol 2 during $c(y2y^R)$. Clearly, from $c(x2x^R)$ and $c(y2y^R)$, we can construct an accepting computation (in space $|x2y^R|^s$) of M on $x2y^R$. This is a contradiction, because $x2y^R$ is not in $L'(ks+1)$. Thus the lemma follows. □

Lemma 5.4. For each $k \geq 1$ and any $s > 1$,

(1) $L'(D_s(k)+k-1) \in \mathcal{L}$[CS-1DCM($k$)[Time($n^s$)]],

(2) $L'(N_s(k)+k-1) \in \mathcal{L}$[CS-1CM($k$)[Time($n^s$)]],

(3) $L'(D'_s(k)+k-1) \in \mathcal{L}$[CS-1DCM($k$)[Space($n^s$)]].

(4) $L'(N'_s(k)+k-1) \in \mathcal{L}$[CS-1CM($k$)[Space($n^s$)]],

Proof : It is obvious that the results are true for $k = 1$. Now suppose $k \geq 2$. We only give the proof of (1), since the proofs of (2), (3) and (4) are just the same. Let M be a 1DCM(k)-Time(n^s) that accepts $L'(D_s(k))$. We construct a CS-1DCM(k) M' from M. M' acts as follows: Consider the case when an input word

$$\underline{a^p 0^{n_1} 1 \cdots 10^{n D_s(k)}} 10^{m_1} 1 \cdots 10^{m_{k-1}} 20^{m'_{k-1}} 1 \cdots 10^{m'_1} \underline{10^{n' D_s(k)} 1 \cdots 10^{n'_1} a^q}\$$

is presented to M'. (Input words in the form different from the above can easily be rejected by M'.)

(a) When reading the part of $a^p 0^{n_1} 1 \cdots 10^{n D_s(k)} 1$, M' simulates the action of M when it reads the input $a^p 0^{n_1} 1 \cdots 10^{n D_s(k)} 2$.

[1] A memory configuration of M is an $(k+1)$-tuple $(q, c_1, \cdots, c_k)$, where q is the current internal state of M and c_i is the contents of the i-th counter of M for $1 \leq i \leq k$,

(b) In the same way as in the proof of Lemma 5.1, M' checks without using any counter whether $m_i = m'_i$ for all $1 \leq i \leq k-1$. If this is the case, then goto (c), otherwise M rejects the input.

(c) M' continues to simulate the action of M when reading the part of $0^{n'_{D_s(k)}}1 \cdots 10^{n'_1}a^q$.

(d) Each machine of M' enters an accepting state (with counter empty) if and only if M finally enters an accepting state.

It will be obvious that M' accepts $L'(N_s(k)+k-1)$ within n^s steps. □

From Theorem 5.1 and Lemma 5.4, we get the following corollary.

Corollary 5.2. For each $s > 1$ and each $k \geq 2$,

(1) $\mathcal{L}$[1CM(k)-Time(n^s)]$\subsetneq\mathcal{L}$[CS-1CM(k)[Time(n^s)]],

(2) $\mathcal{L}$[1DCM(k)-Time(n^s)]$\subsetneq\mathcal{L}$[CS-1DCM(k)[Time(n^s)]],

(3) $\mathcal{L}$[1CM(k)-Space(n^s)]$\subsetneq\mathcal{L}$[CS-1CM(k)[Space(n^s)]],

(4) $\mathcal{L}$[1DCM(k)-Space(n^s)]$\subsetneq\mathcal{L}$[CS-1DCM(k)[Space(n^s)]].

Theorem 5.2. For each $c, s \geq 1$ and each $k \geq 2$,

(1) $\mathcal{L}$[CS-1CM(k)[Time(cn)]]$\subseteq\mathcal{L}$[1CM($2k-1$)-Time(linear)],

(2) $\mathcal{L}$[CS-1CM(k)[Time(cn^{s+1})]]$\subseteq\mathcal{L}$[1CM($2k-1$)-Time(n^{s+1})],

(3) $\mathcal{L}$[CS-1CM(k)[Space(n^s)]]$\subseteq\mathcal{L}$[1CM($2k-1$)-Space($n^{(k-1)s+1}$)],

(4) $\mathcal{L}$[CS-CM(k)[Space(n^s)]]$\subseteq\mathcal{L}$[CM($2k$)-Space(n^s)].

The corresponding result also holds in the deterministic case.

Proof : Let $M = (CM_1, CM_2, \cdots, CM_k)$ be an arbitrary CS-1CM(k). We construct a 1CM($2k-1$) M' from M. M' acts as follows:

(1) M' stores in its finite control all the finite controls of CM_1, CM_2, $\cdots$, CM_k.

(2) For each cell of the input tape:

(a) M' stores in its finite control the internal state of each CM_i $(1 \leq i \leq k)$ when CM_i leaves the cell and the order $\langle d_1, d_2, \cdots, d_k \rangle$ in which CM_1, CM_2, $\cdots$, CM_k leave the cell subsequently (i.e., CM_{d_1} firstly leaves the cell, CM_{d_2} secondly leaves the cell, and so on),[2] and the contents of the counter of CM_i is stored by its i-th counter.

(b) Furthermore, for each i $(1 \leq i \leq k-1)$, the interval between the times at which CM_{d_i} and $CM_{d_{i+1}}$ leave the cell is stored by the $(k+i)$-th counter of M'. This can be done as follows: M' adds to the $(k+i)$-th counter the difference between the numbers of the steps for which $CM_{d_{i+1}}$ and CM_{d_i} stay at the cell.

If M operates with time bound $f(n)$, then it is easy to verify that M' can simulate M in at most $k \cdot f(n)$ time. From this fact and Lemma 5.2, (1) and (2) follow.

During the accepting computation of M, if the contents of its each counter does not exceed n^s, then it is easily seen that for each cell of the input tape, the interval between the times at which CM_{d_i} and $CM_{d_{i+1}}$ leave the cell is $O(n^{(k-1)s+1})$. From this observation it follows that one can make M' simulate M with space bound $n^{(k-1)s+1}$. Thus, (3) holds.

The proof of (4) is obvious, and omitted here. Note that for the deterministic case, we can give the proof of corresponding result in the same way. This completes the proof of the theorem. □

The result of Theorem 5.2 (1) is "optimal" in the following sense. That is, for each $k \geq 2$, $\mathcal{L}$[CS-1DCM(k)[Time($2n$)]]$-\mathcal{L}$[1CM($2k-2$)-Time(linear)]$\neq \emptyset$. In fact, for each $k \geq 2$, $L'(2k-1) \in \mathcal{L}$[CS-1DCM(k)[Time($2n$)]]$-\mathcal{L}$[1CM($2k-2$)-Space(n)] (by Lemma 5.1).

It was shown in [49] that multicounter machines with linear space bound and 3-counter machines with polynomial time bound are as powerful as multicounter machines with polynomial

[2] If CM_{i_1}, CM_{i_2}, $\cdots$, CM_{i_r} $(1 \leq i_1 < i_2 < \cdots < i_r \leq k)$ leave the cell simultaneously, we refer the order on them as $\langle i_1, i_2, \cdots, i_r \rangle$.

time or space bound in all the cases (deterministic and nondeterministic, one-way and two-way). From this fact and Theorems 5.1, 5.2, we have the following corollary.

Corollary 5.3.

(1) $\mathcal{L}$[CS-1CM(3)[Time(poly)]] ($\mathcal{L}$[CS-1DCM(3)[Time(poly)]])
$= \bigcup_{1\le k<\infty} \mathcal{L}$[CS-1CM($k$)[Space($n$)]] ($\bigcup_{1\le k<\infty} \mathcal{L}$[CS-1DCM($k$)[Space($n$)]])
$= \bigcup_{1\le k<\infty} \mathcal{L}$[CS-1CM($k$)[Time(poly)]] ($\bigcup_{1\le k<\infty} \mathcal{L}$[CS-1DCM($k$)[Time(poly)]])
$= \bigcup_{1\le k<\infty} \mathcal{L}$[CS-1CM($k$)[Space(poly)]] ($\bigcup_{1\le k<\infty} \mathcal{L}$[CS-1DCM($k$)[Space(poly)]]), and

(2) $\mathcal{L}$[CS-CM(3)[Time(poly)]] ($\mathcal{L}$[CS-DCM(3)[Time(poly)]])
$= \bigcup_{1\le k<\infty} \mathcal{L}$[CS-CM($k$)[Space($n$)]] ($\bigcup_{1\le k<\infty} \mathcal{L}$[CS-DCM($k$)[Space($n$)]])
$= \bigcup_{1\le k<\infty} \mathcal{L}$[CS-CM($k$)[Time(poly)]] ($\bigcup_{1\le k<\infty} \mathcal{L}$[CS-DCM($k$)[Time(poly)]])
$= \bigcup_{1\le k<\infty} \mathcal{L}$[CS-CM($k$)[Space(poly)]] ($\bigcup_{1\le k<\infty} \mathcal{L}$[CS-DCM($k$)[Space(poly)]]).

5.3 Hierarchies Based on The Number of Counter Machines

In this section, we investigate how the number of counter machines affects the accepting power of the cooperating system of counter machines with polynomial time (space) bound.

5.3.1 One-way Case

Theorem 5.3. For each $c, s, k \ge 1$,

(1) $\mathcal{L}$[CS-1CM(k)[Time(cn^s)]] ($\mathcal{L}$[CS-1DCM(k)[Time(cn^s)]])
$\subsetneq \mathcal{L}$[CS-1CM($k+1$)[Time(cn^s)]] ($\mathcal{L}$[CS-1DCM($k+1$)[Time(cn^s)]]), and

(2) $\mathcal{L}$[CS-1CM(k)[Space(cn^s)]] ($\mathcal{L}$[CS-1DCM(k)[Space(cn^s)]])
$\subsetneq \mathcal{L}$[CS-1CM($k+1$)[Space(cn^s)]] ($\mathcal{L}$[CS-1DCM($k+1$)[Space(cn^s)]])

Proof : For each $c, s, k \ge 1$, let $p_{c,s}(k)$ ($p'_{c,s}(k)$) denote the minimum number of counter machines required for cooperating systems of one-way multicounter machines with time (space)

bound cn^s to accept $L'(k)$, and $q_{c,s}(n)$ ($q'_{c,s}(k)$) denote the minimum number of counter machines required for cooperating systems of one-way deterministic multicounter machines with time (space) bound cn^s to accept $L'(k)$. Furthermore, let

$$CN_{c,s}(k) = \max\{m | p_{c,s}(m) = k\},$$

$$CN'_{c,s}(k) = \max\{m | p'_{c,s}(m) = k\},$$

$$CD_{c,s}(k) = \max\{m | q_{c,s}(m) = k\},$$

$$CD'_{c,s}(k) = \max\{m | q'_{c,s}(m) = k\}.$$

By Theorem 5.2 and Lemma 5.3, for each $c, s, k \geq 1$,

$$CN_{c,s}(k) \leq (2k-1)s,$$

$$CD_{c,s}(k) \leq (2k-1)s,$$

$$CN'_{c,s}(k) \leq (2k-1)((k-1)s+1),$$

$$CD'_{c,s}(k) \leq (2k-1)((k-1)s+1).$$

Then, we can easily prove that

(1) $L'(CN_{c,s}(k)+1) \in \mathcal{L}$[CS-1CM$(k+1)$[Time$(cn^s)$]],

(2) $L'(CD_{c,s}(k)+1) \in \mathcal{L}$[CS-1DCM$(k+1)$[Time$(cn^s)$]],

(3) $L'(CN'_{c,s}(k)+1) \in \mathcal{L}$[CS-1CM$(k+1)$[Space$(cn^s)$]],

(4) $L'(CD'_{c,s}(k)+1) \in \mathcal{L}$[CS-1DCM$(k+1)$[Space$(cn^s)$]].

For example, let M be a CS-1CM(k)[Time(cn^s)] accepting $L'(CN_{c,s}(k))$. We consider a CS-1CM$(k+1)$[Time(cn^s)] M' which acts as follows: Suppose that an input word

$$a^p 0^q 1 w 1 0^{q'} a^{p'},$$

where $w \in \{0^{i_1}10^{i_2}1\cdots10^{i_{CN_{c,s}(k)}}20^{i'_{CN_{c,s}(k)}}1\cdots10^{i'_2}10^{i'_1} \mid \forall j(1 \leq j \leq CN_{c,s}(k))[i_j, i'_j \geq 1]\}$ and $p, p' \geq 0$, $q, q' \geq 1$. (Input words in the form different from the above can be easily rejected by M'.) M' simulates the action of M on w by using its k counter machines, and checks by using the remaining counter machine whether $q = q'$. Each machine of M' enters an accepting state (with counter empty) only if it finds out that (1) M accepts w (i.e., $w \in L(CN_{c,s}(k))$) and (2) $q = q'$. Noting that for each $w \in L(CN_{c,s}(k))$ and each $w' = a^p0^q1w10^qa^{p'} \in L'(CN_{c,s}(k)+1)$, $|w|^s + p + p' + 2(q+1) \leq |w'|^s$. It will be obvious that M' accepts $L'(CN_{c,s}(k)+1)$ in time n^s.

Thus, the theorem follows from the above definitions and facts. □

Remark. The reader might have noted that the same technique is used in Chapter 4 (Theorem 4.6) and Chapter 5 (Theorem 5.3) for proving separation results. Its idea is rather straightforward: Given any cooperating system $A(k)$, where k is the number of processors in the system, we try to find a sequence of languages $L(1)$, $L(2)$, $\cdots$ such that there is a $L(f(k))$, $L(f(k)) \in \mathcal{L}[A(k+1)] - \mathcal{L}[A(k)]$, where $f(k)$ depends only on k, and is bounded. Note that it suffices to prove only the existence of $f(k)$ for our purpose. This is, in general, much simpler than to determine the value of $f(k)$. In particular, it is effective for the one-way (or three-way) case. Clearly, this technique can also be applied to other types of automata.

5.3.2 Two-way Case

In [21], Monien showed, by using a transformational method, that for languages over a one-letter alphabet, two-way $(k+1)$-head finite automata are more powerful than two-way k-head finite automata for each $k \geq 1$. In [44], Wang, Inoue and Takanami showed, by carefully extending the ideas of Monien [21], that for languages over a one-letter alphabet, two-way sensing $(k+1)$-head finite automata are more powerful than two-way sensing k-head finite automata for each $k \geq 3$. In this subsection, we give an analogous result for the cooperating system of (two-way) multicounter machines with space bound n, by using a similar method.

Let $\mathcal{L}_\Sigma$[CS-CM(k)[Space(n)]] ($\mathcal{L}_\Sigma$[CS-DCM(k)[Space(n)]]) denote the class of languages over the alphabet Σ accepted by CS-CM(k)[Space(n)]'s (CS-DCM(k)[Space(n)]'s), for each

$k \geq 1$. Let $\mathrm{SPACE}_\Sigma(f(n))$ ($\mathrm{DSPACE}_\Sigma(f(n))$) denote the class of languages over the alphabet Σ accepted by nondeterministic (deterministic) Turing machines with space bound $f(n)$.

In the following we only consider languages $L \subseteq \{0^{2^n} | n \geq 1\}$. Let $\tilde{\mathcal{L}}$[CS-CM(k)[Space(n)]] ($\tilde{\mathcal{L}}$[CS-DCM(k)[Space(n)]]) be the class of all languages L such that

$$L \in \mathcal{L}_{\{0\}}[\text{CS-CM}(k)[\text{Space}(n)]] \quad (L \in \mathcal{L}_{\{0\}}[\text{CS-DCM}(k)[\text{Space}(n)]])$$

and $L \subseteq \{0^{2^n} | n \geq 1\}$. Let $\widetilde{\mathrm{SPACE}}(\log n)$ ($\widetilde{\mathrm{DSPACE}}(\log n)$) be the class of all languages L such that

$$L \in \mathrm{SPACE}_{\{0\}}(\log n) \quad (L \in \mathrm{DSPACE}_{\{0\}}(\log n))$$

and $L \subseteq \{0^{2^n} | n \geq 1\}$.

For each $L \subseteq \{0^{2^n} | n \geq 1\}$ and $j \geq 1$, let

$$T_j(L) = \{0^{2^{j \cdot n}} | n \geq 1 \ \& \ 0^{2^n} \in L\}.$$

In all the proofs in this subsection there is no difference at all between the deterministic and the nondeterministic cases. Therefore we will always consider only the deterministic case.

Lemma 5.5. For each $k \geq 1$,

(1) $\tilde{\mathcal{L}}$[CS-CM(k)[Space(n)]]$\subsetneq\widetilde{\mathrm{SPACE}}(\log n)$, and

(2) $\tilde{\mathcal{L}}$[CS-DCM(k)[Space(n)]]$\subsetneq\widetilde{\mathrm{DSPACE}}(\log n)$.

Proof : It was shown in [21] (Lemma 1) that for all $k \geq 1$, $\tilde{\mathcal{L}}$[HA(k)]$\subsetneq\widetilde{\mathrm{SPACE}}(\log n)$ and $\tilde{\mathcal{L}}$[DHA(k)]$\subsetneq\widetilde{\mathrm{DSPACE}}(\log n)$, where $\tilde{\mathcal{L}}$[HA(k)] ($\tilde{\mathcal{L}}$[DHA(k)]) is the class of all languages L such that $L \subseteq \{0^{2^n} | n \geq 1\}$ and $L \in \mathcal{L}$[HA(k)] ($L \in \mathcal{L}$[DHA(k)]). On the other hand, one can easily show that for each $k \geq 1$, $\tilde{\mathcal{L}}$[CS-CM(k)[Space(n)]]$\subseteq$ $\tilde{\mathcal{L}}$[HA($2k+1$)] and $\tilde{\mathcal{L}}$[CS-DCM(k)[Space(n)]]$\subseteq$ $\tilde{\mathcal{L}}$[DHA($2k+1$)]. From those facts, the lemma follows. □

Lemma 5.6. For each $L \in \widetilde{\mathrm{SPACE}}(\log n)$ ($\widetilde{\mathrm{DSPACE}}(\log n)$), there exists an integer $j \geq 1$ such that $T_j(L) \in \tilde{\mathcal{L}}$[CS-CM(2)[Space($n$)]] ($\tilde{\mathcal{L}}$[CS-DCM(2)[Space($n$)]]).

Proof : The proof is very similar to that of Lemma 2 in [21]. Let L be any language in $\widetilde{\text{DSPACE}}(\log n)$, and M be a deterministic Turing machine accepting L within space bound $\log n$. Let M' be the following modification of M:

(1) M' writes bi(n) on its working tape, where n is the length of input tape and bi: the set of natural numbers, $\mathcal{N} \to \{0,1\}^*$, be the bijective mapping defined by: bi(n) $= \varphi \Leftrightarrow 1\varphi$ is the binary notation of n.

(2) During the rest of the computation M' never uses its input tape again. M' simulates M and during this simulation its working tape is divided into 3 tracks. On the first track M' stores bi(n), on the second track the position of the input head of M in binary notation and on the third track the inscription of the worktape of M.

Furthermore we can define M' in such a way that it has only two worktape symbols. There exists some $j \geq 1$ such that M' uses for every computation at most $j \cdot \log_2 n$ cells on its worktape.

We now define a deterministic CS-DCM(2) M'' accepting $T_j(L)$. M'' first tests whether the input tape is of the form $0^{2^{j \cdot n}}$, $n \geq 1$, using its two counters. If this is the case, then M'' simulates the action of M' on the input tape 0^{2^n}. The working tape of M' is divided by the head position into two parts as described in Fig. 5.1, and M'' stores this during the simulation on its input tape by using the input heads of its two counter machines (CM_1 and CM_2) as described in Fig. 5.2 (with the counters empty).

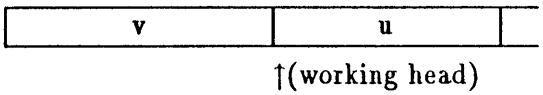

Fig. 5.1. The working tape of M'.

In order to simulate one step of M', CM_1 (CM_2) has to divide or multiply the number un(v^R) (un(u)) by two, and has to add $+1$ or -1 to the number un(v^R) (un(u)) (where "un" is

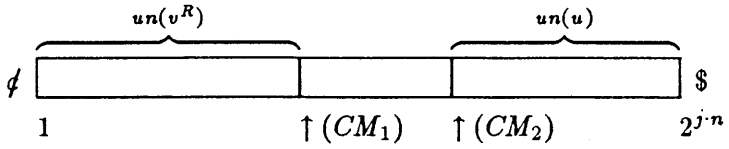

Fig. 5.2. The input tape of M'', where un: $\{0,1\}^* \to \mathcal{N}$ is the inverse mapping of bi.

the inverse mapping of "bi"). Moreover, CM_1 and CM_2 have to communicate with each other. All of this can be easily done with the help of their empty counters.

In order to initialize this simulation, M'' sets the position of the input head of CM_1 to $\text{un}(\text{bi}(2^n)^R) = \text{un}(\underbrace{00\cdots0}_{n}) = 2^n$ and then moves the input head of CM_2 to the right endmarker \$. This can be done using two counters.

It is clear that M'' accepts an input tape $0^{2^{j \cdot n}}$ if and only if M' accepts 0^{2^n}. □

Lemma 5.7. For each $L \subseteq \{0^{2^n} \mid n \geq 1\}$ and for $j, k \geq 1$:

$T_j(L) \in \tilde{\mathcal{L}}[\text{CS-CM}(k)[\text{Space}(n)]]$ ($\tilde{\mathcal{L}}[\text{CS-DCM}(k)[\text{Space}(n)]]$)

$\Rightarrow L \in \tilde{\mathcal{L}}[\text{CS-CM}(jk+1)[\text{Space}(n)]]$ ($\tilde{\mathcal{L}}[\text{CS-DCM}(jk+1)[\text{Space}(n)]]$).

Proof : It was shown in [44] (Lemma 3.3) that $T_j(L) \in \tilde{\mathcal{L}}[\text{SeHA}(k)]$ ($\tilde{\mathcal{L}}[\text{SeDHA}(k)]$) $\Rightarrow L \in \tilde{\mathcal{L}}[\text{SeHA}(jk)]$ ($\tilde{\mathcal{L}}[\text{SeDHA}(jk)]$), where $\tilde{\mathcal{L}}[\text{SeHA}(k)]$ ($\tilde{\mathcal{L}}[\text{SeDHA}(k)]$) is the class of all languages L such that $L \subseteq \{0^{2^n} \mid n \geq 1\}$ and $L \in \mathcal{L}[\text{SeHA}(k)]$ ($L \in \mathcal{L}[\text{SeDHA}(k)]$). On the other hand, one can easily show that for each $k \geq 1$,

$$\tilde{\mathcal{L}}[\text{CS-CM}(k)[\text{Space}(n)]]\ (\tilde{\mathcal{L}}[\text{CS-DCM}(k)[\text{Space}(n)]]) \subseteq \tilde{\mathcal{L}}[\text{SeHA}(2k)]\ (\tilde{\mathcal{L}}[\text{SeDHA}(2k)]),$$

and

$$\tilde{\mathcal{L}}[\text{SeHA}(k)]\ (\tilde{\mathcal{L}}[\text{SeDHA}(k)]) \subseteq \tilde{\mathcal{L}}[\text{CS-CM}(k+1)[\text{Space}(n)]]\ (\tilde{\mathcal{L}}[\text{CS-DCM}(k+1)[\text{Space}(n)]]).$$

From those facts, the lemma follows. □

Lemma 5.8. For each $L \subseteq \{0^{2^n} \mid n \geq 1\}$ and for $j > 3k$, $k \geq 1$:

$T_{j+1}(L) \in \tilde{\mathcal{L}}[\text{CS-CM}(k)[\text{Space}(n)]]$ ($\tilde{\mathcal{L}}[\text{CS-DCM}(k)[\text{Space}(n)]]$)

$\Rightarrow T_j(L) \in \tilde{\mathcal{L}}[\text{CS-CM}(k+1)[\text{Space}(n)]]$ ($\tilde{\mathcal{L}}[\text{CS-DCM}(k+1)[\text{Space}(n)]]$).

Proof : Let $M = (CM_1, CM_2, \cdots, CM_k)$ be a CS-DCM(k)[Space(n)] accepting $T_{j+1}(L)$.

We will construct a CS-DCM$(k+1)$[Space(n)] $M' = (CM'_1, CM'_2, \cdots, CM'_{k+1})$ accepting $T_j(L)$.

It can be tested easily (using 2 counters) whether the input is of the form $0^{2^{j \cdot n}}$ for some $n \geq 1$. If this is the case, then M' has to test whether $0^{2^{(j+1)n}}$ is accepted by M. In order to do so M' encodes the input head position, h_v, of each CM_v, and the counter contents, c_v, of each CM_v, where for each $1 \leq v \leq k$, $0 \leq h_v, c_v \leq 2^{(j+1)n} + 1$, by the input head position, h'_v, of CM'_v, the counter contents, c'_v, of CM'_v, and two additional numbers σ_{2v}, σ_{2k+v}, in such a form that always

$$0 \leq h'_v, c'_v \leq 2^{j \cdot n} + 1, \quad 0 \leq \sigma_{2v}, \sigma_{2k+v} < 2^n,$$

and

$$h_v = h'_v + \sigma_{2v} \cdot 2^{j \cdot n}, \quad c_v = c'_v + \sigma_{2k+v} \cdot 2^{j \cdot n}.$$

Note that $h_v = 2^{(j+1)n} + 1$ iff $h'_v = 2^{j \cdot n} + 1$ and $\sigma_{2v} = 2^n - 1$, and that $h_v = h_u$ iff either $(h'_v = h'_u$ and $\sigma_{2v} = \sigma_{2u})$ or $(h'_v - h'_u = \pm 2^{j \cdot n}$ and $\sigma_{2v} - \sigma_{2u} = \mp 1)$ holds.

M' uses the counter of CM'_{k+1} to store the $2k$ numbers $\sigma_2, \cdots, \sigma_{2k}, \sigma_{2k+1}, \cdots, \sigma_{3k}$ in the form

$$c'_{k+1} = \sigma_1 + \sigma_2 \cdot 2^n + \sigma_3 \cdot 2^{2n} + \cdots + \sigma_{3k} \cdot 2^{(3k-1)n} + 2^{(j-1)n}$$

where c'_{k+1} denotes the counter contents of CM'_{k+1} and $\sigma_{2i-1} = 1$ for $1 \leq i \leq k$. This is possible since $j > 3k$.

First M' has to encode the initial head positions and initial counter contents of M. That means it has to set

$$c'_{k+1} \leftarrow 2^{0 \cdot n} + 2^{2 \cdot n} + \cdots + 2^{(2k-2)n} + 2^{(j-1)n}.$$

This can be done easily (using two counter machines).

During the simulation of one step of M, CM'_1, CM'_2, $\cdots$, CM'_k can communicate with each other by means of CM'_{k+1} (since the input head of CM'_{k+1} is free in the computation), and every CM'_v $(1 \leq v \leq k)$ always stores in its finite memory which of the σ_{2v}'s, encoded by c'_{k+1}, are equal to $2^n - 1$ and the information whether $\sigma_{2v} = \sigma_{2u}$ or $\sigma_{2v} = \sigma_{2u} \pm 1$ holds for each

$v, u \in \{1, 2, \cdots, k\}$, and $v \neq u$. Furthermore, in order to simulate the action of CM_v, CM'_v has to distinguish two cases:

(1) $h'_v \neq 0$, $h'_v \neq 2^{j \cdot n} + 1$ (i.e., the input head of CM'_v does not scan either of the endmarkers) and $0 < c'_v < 2^{j \cdot n} + 1$. (Note that CM'_v can check with the help of CM'_{k+1} whether or not its counter contents is equal to $2^{j \cdot n} + 1$. In this case, the input head of CM_v does not scan either of the endmarkers, and $c_v > 0$. CM'_v simply changes its input head position and its counter contents in the same way as CM_v would do.

(2) $h'_v = 0$ or $h'_v = 2^{j \cdot n} + 1$ or $c'_v = 0$ or $c'_v = 2^{j \cdot n} + 1$. If $h'_v = 0$ (or $c'_v = 0$) and CM_v would move the input head to the left (or would decrease the counter), then CM'_v has to set $h'_v \leftarrow 2^{j \cdot n} - 1$ and $\sigma_{2v} \leftarrow \sigma_{2v} - 1$ (or $c'_v \leftarrow 2^{j \cdot n} - 1$ and $\sigma_{2k+v} \leftarrow \sigma_{2k+v} - 1$). If $h'_v = 2^{j \cdot n} + 1$ (or $c'_v = 2^{j \cdot n} + 1$) and CM_v would move the input head to the right (or would increase the counter), then CM'_v has to set $h'_v \leftarrow 2$ and $\sigma_{2v} \leftarrow \sigma_{2v} + 1$ (or $c'_v \leftarrow 2$ and $\sigma_{2k+v} \leftarrow \sigma_{2k+v} + 1$). Otherwise, CM'_v simply changes its input head position and its counter contents in the same way as CM_v would do.

In the simulation, performing the operation on σ_{2v} (or σ_{2k+v}) is the difficulty in this proof, and the other operations may be easily done (with the help of CM'_{k+1}). In the following we will show how M' can perform the operation on σ_{2v}'s (or σ_{2k+v}'s).

M' can perform the operation ± 1 on σ_{2v} (or σ_{2k+v}) and test whether the new σ_{2v} (or σ_{2k+v}) is equal to $2^n - 1$, using the algorithms described in 1 and 2 in the proof of Lemma 4 in [21]. Below, we refer to these algorithms as Algorithm 1 and Algorithm 2. For the sake of completeness, we recall them here, pointing out the necessary changes.

We will denote the counter contents of CM'_{k+1} by λ as follows:

$$\lambda = \sum_{\mu=1}^{j-1} \sigma_\mu \cdot 2^{(\mu-1)n} + 2^{(j-1)n}$$

with $\sigma_{2i-1} = 1$ for $1 \leq i \leq k$, and $\sigma_{3k+1} = \sigma_{3k+2} = \cdots = \sigma_{j-1} = 0$.

Furthermore we can assume that $c'_v < 2^{j \cdot n}$. (Note that M' can test whether $c'_v < 2^{j \cdot n}$ by moving the input head two cells to the right.) If $c'_v \geq 2^{j \cdot n}$ then we set $c'_v = 2^{j \cdot n} - 1$ and store

the difference in the finite control of CM'_v. We decompose c'_v in the form

$$c'_v = \psi_{v1} + \psi_{v2} \cdot 2^n$$

with $0 \leq \psi_{v1} < 2^n$, $0 \leq \psi_{v2} < 2^{(j-1)n}$.

Algorithm 1: Now suppose that $h'_v = 2^{j \cdot n} + 1$ and CM_v would move the input head to the right. (The cases $h'_v = 0$, $c'_v = 0$ and $c'_v = 2^{j \cdot n} + 1$ will lead to analogous considerations.) Note that the input head of CM'_v does not store anything and therefore it is free for intermediate computations.

CM'_v and CM'_{k+1} change $c'_v = \psi_{v1} + \psi_{v2} \cdot 2^n$, $0 \leq \psi_{v1} < 2^n$, $0 \leq \psi_{v2} < 2^{(j-1)n}$, and $\lambda = \sum_{\mu=1}^{j-1} \sigma_\mu \cdot 2^{(\mu-1)n} + 2^{(j-1)n}$, $\sigma_{2i-1} = 1$ for $1 \leq i \leq k$, $\sigma_{3k+1} = \sigma_{3k+2} = \cdots = \sigma_{j-1} = 0$, into

$$c'_v = \psi_{v2} + 2^{(j-1)n},$$

$$\lambda = R_n(\psi_{v1}) + \sigma_1 \cdot 2^n + \cdots + \sigma_{j-1} \cdot 2^{(j-1)n},$$

where for any $x < 2^n$, $R_n(x)$ is defined in the following way:

Let $\varphi_n(x) \in \{0,1\}^*$, $\mid \varphi_n(x) \mid = n$, be the binary notation of x lengthened by an appropriate number of leading zeros. Then $R_n(x) < 2^n$ is that number whose binary notation of length n (again allowing leading zeros) is the reversal of $\varphi_n(x)$. Note that $R_n(R_n(x)) = x$ for all $x < 2^n$.

CM'_v and CM'_{k+1} can do the above by executing the following:

$c'_v \leftarrow c'_v + 2^{j \cdot n}$,

While$_1$ $\lambda < 2^{j \cdot n}$ *Do*$_1$

Begin$_1$

$c'_v \leftarrow \left[\frac{c'_v}{2}\right]$, and $\alpha \leftarrow c'_v - 2\left[\frac{c'_v}{2}\right]$,

$\lambda \leftarrow \alpha + 2\lambda$,

End$_1$

$\lambda \leftarrow \lambda - 2^{j \cdot n}$.

It is clear that CM'_v and CM'_{k+1} can perform this computation, since their input heads are free now. The loop $(Begin_1, \cdots, End_1)$ is carried out exactly n times and this leads to the counter contents c'_v and λ that we wanted. Note that $2^{j \cdot n}$ is given by the position of the right endmarker, and during the computation, if $c'_v, \lambda \geq 2^{j \cdot n} + 1$, we can, in fact, store $\left[\frac{c'_v}{2^{j \cdot n}+1}\right]$, $\left[\frac{\lambda}{2^{j \cdot n}+1}\right]$ in the finite controls of CM'_v and CM'_{k+1}, and the residues $c'_v - (2^{j \cdot n}+1)\left[\frac{c'_v}{2^{j \cdot n}+1}\right]$, $\lambda - (2^{j \cdot n}+1)\left[\frac{\lambda}{2^{j \cdot n}+1}\right]$ in the counters of CM'_v and CM'_{k+1}, respectively.

Algorithm 2: CM'_v and CM'_{k+1} change c'_v and λ into

$$c'_v = \sigma_{j-1} + \psi_{v2} \cdot 2^n,$$

$$\lambda = R_n(\psi_{v1}) + \sigma_1 \cdot 2^n + \cdots + \sigma_{j-2} \cdot 2^{(j-2)n} + 2^{(j-1)n}.$$

Let $Bit_m(x)$ denote the m-th least significant bit of the binary notation of x with length $\geq m$ (allowing leading zeros). CM'_{k+1} first changes λ into

$$\lambda = R'_n(\psi_{v1}) + \sigma_1 \cdot 2^n + \cdots + \sigma_{j-2} \cdot 2^{(j-2)n} + 2^{(j-1)n},$$

where

$$R'_n(\psi_{v1}) = \begin{cases} R_n(\psi_{v1}) & \text{if } Bit_1(\psi_{v1}) = 1, \\ R_n(\psi_{v1}) + 1 & \text{otherwise.} \end{cases}$$

That is, only the lowest bit of λ is changed in such a way that λ becomes an odd number. (It is stored in the finite control whether $R_n(\psi_{v1})$ is odd or even.)

Afterwards CM'_v and CM'_{k+1} execute the following:

While$_2$ $\quad c'_v < 2^{j \cdot n}$ $\quad$ *Do*$_2$

Begin$_2$

$\quad$ *If*$_1$ $\quad 2\lambda \geq 2^{j \cdot n}$

$\quad$ *then*$_1$

$\quad\quad \lambda \leftarrow 2\lambda - 2^{j \cdot n}$,

$\quad\quad c'_v \leftarrow 1 + 2c'_v$,

$\quad$ *else*$_1$

$\lambda \leftarrow 2\lambda$,

$c'_v \leftarrow 2c'_v$,

Endif$_1$

End$_2$

In order to see what is done by the above execution, we consider the decomposition

$$\lambda = \lambda' + \alpha \cdot 2^{j \cdot n-1}, \quad \lambda' < 2^{j \cdot n-1}.$$

Then $\alpha = 1$ if and only if $2\lambda \geq 2^{j \cdot n}$. Note that $Bit_{j \cdot n}(\lambda) = \alpha$. As in *Algorithm 1* it can be seen that the loop (*Begin*$_2$, $\cdots$, *End*$_2$) is carried out exactly n times and the number σ_{j-1} is carried over from λ to c'_v bit by bit. Therefore, c'_v and λ are changed by this execution into

$$c'_v = \sigma_{j-1} + \psi_{v2} \cdot 2^n + 2^{j \cdot n},$$

$$\lambda = R'_n(\psi_{v1})2^n + \sigma_1 \cdot 2^{2n} + \cdots + \sigma_{j-2} \cdot 2^{(j-1)n}.$$

Then, we obtain the counter contents that we wanted by:

(a) subtracting $2^{j \cdot n}$ from c'_v,

(b) adding $2^{j \cdot n}$ to λ,

(c) dividing λ by 2 as long as the remainder is 0,

(d) changing $R'_n(\psi_{v1})$ into $R_n(\psi_{v1})$.

Instead of

$$c'_v = \psi_{v1} + \psi_{v2} \cdot 2^n, \quad \lambda = \sum_{\mu=1}^{j-1} \sigma_\mu \cdot 2^{(\mu-1)n} + 2^{(j-1)n},$$

we write

$$(c'_v; \lambda) = (\psi_{v1}; \sigma_1, \cdots, \sigma_{j-1}).$$

The application of *Algorithms 1, 2* induces the transition

$$(\psi_{v1}; \sigma_1, \cdots, \sigma_{j-1}) \rightarrow (\sigma_{j-1}; R_n(\psi_{v1}), \sigma_1, \cdots, \sigma_{j-2}).$$

Therefore, by $j-2v$ applications, we get

$$(\psi_{v1};\sigma_1,\cdots,\sigma_{j-1}) \rightarrow (\sigma_{j-1};R_n(\psi_{v1}),\sigma_1,\cdots,\sigma_{j-2}) \rightarrow \cdots$$
$$\rightarrow (\sigma_{2v};R_n(\sigma_{2v+1}),\cdots,R_n(\sigma_{j-1}),R_n(\psi_{v1}),\sigma_1,\cdots,\sigma_{2v-1}).$$

Now *Algorithm 1* is applied again. Since during the computation σ_{2v} is carried over from c'_v to λ bit by bit, CM'_v and CM'_{k+1} are able to add $+1$ and to test whether the new σ_{2v} is equal to 2^n-1 (this is true if and only if all bits which are carried over are equal to one). Afterwards we apply *Algorithm 2* and get

$$(\sigma_{2v-1};R_n(\sigma_{2v}+1),R_n(\sigma_{2v+1}),\cdots,R_n(\sigma_{j-1}),R_n(\psi_{v1}),\sigma_1,\cdots,\sigma_{2v-2}).$$

Since $R_n(R_n(x))=x$ for all $x<2^n$, further applications of *Algorithms 1, 2* lead to

$$(\psi_{v1};\sigma_1,\cdots,\sigma_{2v-1},\sigma_{2v}+1,\sigma_{2v+1},\cdots,\sigma_{j-1}).$$

This shows that by the applications of *Algorithms 1,2*, M' can perform the operation ±1 on σ_{2v} (or σ_{2k+v}) and test whether the new σ_{2v} (or σ_{2k+v}) is equal to 2^n-1.

Next we show how M' can decide whether the new $\sigma_{2v}=\sigma_{2u}$ for each $u\in\{1,2,\cdots,k\}$ and $v\neq u$ (when $k\geq 2$), by the application of *Algorithm 3* below. We will use CM'_v, CM'_u and CM'_{k+1}. (Note that the input heads of CM'_v and CM'_{k+1} are free.) As in the above, we decompose c'_v and c'_u in the form

$$c'_v=\psi_{v1}+\psi_{v2}\cdot 2^n$$

with $0\leq\psi_{v1}<2^n$, $0\leq\psi_{v2}<2^{(j-1)n}$, and

$$c'_u=\psi_{u1}+\psi_{u2}\cdot 2^n$$

with $0\leq\psi_{u1}<2^n$, $0\leq\psi_{u2}<2^{(j-1)n}$. The counter contents of CM'_{k+1} is denoted by λ as follows:

$$\lambda=\sum_{\mu=1}^{j-1}\sigma_\mu\cdot 2^{(\mu-1)n}+2^{(j-1)n}$$

with $\sigma_{2i-1}=1$ for $1\leq i\leq k$, and $\sigma_{3k+1}=\sigma_{3k+2}=\cdots=\sigma_{j-1}=0$.

Algorithm 3: We first apply *Algorithms 1, 2* to $(c'_v;\lambda) = (\psi_{v1};\sigma_1,\cdots,\sigma_{j-1})$ by $(j-2v)$ times to get

$$(\psi_{v1};\sigma_1,\cdots,\sigma_{j-1})$$
$$\Rightarrow (\sigma_{2v};R_n(\sigma_{2v+1}),\cdots,R_n(\sigma_{j-1}),R_n(\psi_{v1}),\sigma_1,\cdots,\sigma_{2v-1}).$$

After that, we apply *Algorithms 1, 2* to $(c'_u;\lambda) = (\psi_{u1};R_n(\sigma_{2v+1}),\cdots,R_n(\sigma_{j-1}),R_n(\psi_{v1}),\sigma_1,\cdots,\sigma_{2v-1})$ by $2(v-u)$ times to get

$$(\psi_{u1};R_n(\sigma_{2v+1}),\cdots,R_n(\sigma_{j-1}),R_n(\psi_{v1}),\sigma_1,\cdots,\sigma_{2v-1})$$
$$\Rightarrow (\sigma_{2u};R_n(\sigma_{2u+1}),\cdots,R_n(\sigma_{2v-1}),R_n(\psi_{u1}),$$
$$R_n(\sigma_{2v+1}),\cdots,R_n(\sigma_{j-1}),R_n(\psi_{v1}),\sigma_1,\cdots,\sigma_{2u-1}),$$

where we suppose, without loss of generality, that $1 \le u < v \le j$.

Now CM'_v, CM'_u and CM'_{k+1} compare σ_{2v} with σ_{2u} from $Bit_1(\sigma_{2v})$ to $Bit_{\lceil \frac{n}{2} \rceil}(\sigma_{2v})$ by executing the following computation. Note that in the following computation, $Bit_2(\lambda)$ (or its reverse) is used as an identifier for distinguishing the bits of σ_{2v} (σ_{2u}) from λ when the bits of σ_{2v} (σ_{2u}) are carried over to λ by using a rotation technique.

$\alpha \leftarrow Bit_2(\lambda)$,

$\beta = \gamma \leftarrow 0$,

While$_3$ $\lambda < 2^{j\cdot n}$ *and* $\beta = \gamma$ *Do*$_3$

Begin$_3$

$\beta \leftarrow Bit_1(c'_v)$,

$\gamma \leftarrow Bit_1(c'_u)$,

If$_2$ $\beta = \gamma$

then$_2$

$c'_v \leftarrow \left[\frac{c'_v}{2}\right]$,

$c'_u \leftarrow \left[\frac{c'_u}{2}\right]$,

If$_3$ $4\lambda < 2^{j\cdot n}$

then$_3$

$\lambda \leftarrow \bar{\alpha} + 2\lambda$,

$\lambda \leftarrow \beta + 2\lambda$,

$else_3$

$If_3 \quad 2\lambda < 2^{j \cdot n}$

$then_4$

$T \leftarrow 0$,

$else_4$

$T \leftarrow 1$,

$Endif_4$,

$\lambda \leftarrow \alpha + 2\lambda$,

$\lambda \leftarrow \beta + 2\lambda$,

$Endif_3$

$Endif_2$

End_3,

where $\bar{\alpha}$ denotes the reverse of α, that is, if $\alpha = 1$ then $\bar{\alpha} = 0$ else $\bar{\alpha} = 1$.

Note that in the above ($While_3 \cdots Do_3$) only one of control conditions ($\lambda < 2^{j \cdot n}$) and ($\beta = \gamma$) is changed. It is clear that after ($While_3 \cdots Do_3$) is carried out, if $\beta \neq \gamma$, then this means $\sigma_{2v} \neq \sigma_{2u}$ (so M' will restore the used counter machines, respectively, as jsut before ($While_3 \cdots Do_3$) is done, by the application of procedure *Replace* below), and otherwise (i.e., if $\beta = \gamma$), the loop ($Begin_3 \cdots End_3$) is carried out exactly $\lceil \frac{n}{2} \rceil$ times, and this leads to

$$\lambda = R_n(\sigma_{2u+1}) \cdot 2^{2\lceil \frac{n}{2} \rceil} + \cdots + \sigma_{2u-1} \cdot 2^{(j-2)n} \cdot 2^{2\lceil \frac{n}{2} \rceil} + 2^{(j-1)n} \cdot 2^{2\lceil \frac{n}{2} \rceil}.$$

Note that $\sigma_{2u-1} = 1$ and $2^{(j-1)n} \cdot 2^{2\lceil \frac{n}{2} \rceil} = (T+1)2^{j \cdot n}$. CM'_v, CM'_u and CM'_{k+1} then execute the following and compare σ_{2v} with σ_{2u} from $Bit_{\lceil \frac{n}{2} \rceil + 1}(\sigma_{2v})$ to $Bit_n(\sigma_{2v})$ if necessary.

$If_5 \quad \beta \neq \gamma$

$then_5$

$Replace(\lambda, \kappa_1, \kappa_2; \alpha)$,

$else_5$

$\lambda \leftarrow \lambda - (T+1)2^{j \cdot n}$,

$While_4 \quad \lambda < 2^{j \cdot n} \quad and \quad \beta = \gamma \quad Do_4$

$Begin_4$

$\beta \leftarrow Bit_1(c'_v)$,

$\gamma \leftarrow Bit_1(c'_u)$,

$If_6 \quad \beta = \gamma$

$then_6$

$c'_v \leftarrow \left[\frac{c'_v}{2}\right]$,

$c'_u \leftarrow \left[\frac{c'_u}{2}\right]$,

$\lambda \leftarrow \bar{\alpha} + 2\lambda$,

$\lambda \leftarrow \beta + 2\lambda$,

$Endif_6$

End_4,

$Replace(\lambda, c'_v, c'_u; \alpha)$,

$\lambda \leftarrow \lambda + (T+1)2^{j \cdot n}$,

$\pi \leftarrow Bit_1(\lambda)$,

$c'_v \leftarrow \pi + 2c'_v$,

$c'_u \leftarrow \pi + 2c'_u$,

$\lambda \leftarrow \left[\frac{\lambda}{2^2}\right]$,

$Replace(\lambda, \kappa_1, \kappa_2; \alpha)$,

$Endif_5$

Procedure $Replace(\lambda, c'_v, c'_u; \alpha)$:

While $Bit_2(\lambda) \neq \alpha$ *Do*

Begin

$\pi \leftarrow Bit_1(\lambda)$,

$c'_v \leftarrow \pi + 2c'_v$,

$c'_u \leftarrow \pi + 2c'_u$,

$\lambda \leftarrow \left[\frac{\lambda}{2^2}\right]$,

End

Endprocedure

It will be easily seen that the loop ($Begin_4 \cdots End_4$) is carried out at most $\lfloor \frac{n}{2} \rfloor$ times, and that during the above computation, the number σ_{2v} (σ_{2u}) is carried over to λ bit by bit as long as $\beta = \gamma$ holds. (See Fig. 5.3 and Fig. 5.4.)

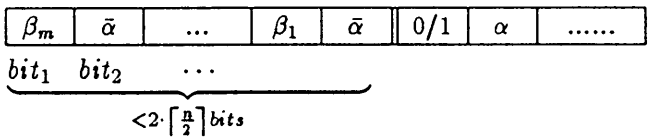

Fig. 5.3. The binary notation of λ after ($While_3 \cdots Do_3$) ends with $\beta \neq \gamma$, where for each $1 \leq i \leq m$, $\beta_i = Bit_i(\sigma_{2v}) = Bit_i(\sigma_{2u})$ (but $Bit_{m+1}(\sigma_{2v}) \neq Bit_{m+1}(\sigma_{2u})$).

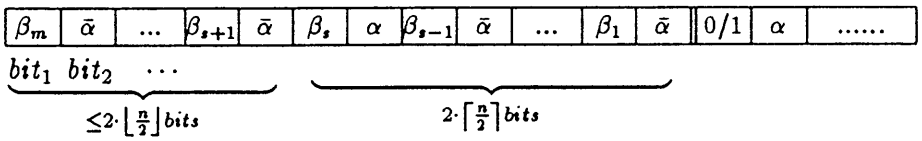

Fig. 5.4. The binary notation of λ after ($While_4 \cdots Do_4$) is carried out, where $s = \lceil \frac{n}{2} \rceil$ and if $m = n$ (i.e., $\beta = \gamma$) then $\beta_n \beta_{n-1} \cdots \beta_1$ is the binary representation of σ_{2v} ($= \sigma_{2u}$).

Now M' has finished the comparison between σ_{2v} and σ_{2u} (if $\beta = \gamma$, then $\sigma_{2v} = \sigma_{2u}$, and otherwise $\sigma_{2v} \neq \sigma_{2u}$) and restored the used counter contents as just before ($While_3 \cdots Do_3$). Then, further applications of *Algorithms 1, 2* lead to

$$(c'_u; \lambda) = (\psi_{u1}; R_n(\sigma_{2v+1}), \cdots, R_n(\sigma_{j-1}), R_n(\psi_{v1}), \sigma_1, \cdots, \sigma_{2v-1})$$

and

$$(c'_v; \lambda) = (\psi_{v1}; \sigma_1, \cdots, \sigma_{j-1}).$$

This shows that M' is able to simulate M step by step. □

Theorem 5.4. For each $k \geq 1$,

(1) $\mathcal{L}$[CS-CM(k)[Space(n)]]$\subsetneq\mathcal{L}$[CS-CM($k+1$)[Space(n)]],

(2) $\mathcal{L}$[CS-DCM(k)[Space(n)]]$\subsetneq\mathcal{L}$[CS-DCM($k+1$)[Space(n)]],

even if the alphabet is restricted to a one-letter.

Proof: Suppose there exists some $k \geq 1$ such that $\mathcal{L}_{\{0\}}$[CS-DCM(k)[Space(n)]]=$\mathcal{L}_{\{0\}}$[CS-DCM($k+1$)[Space(n)]]. This implies $\widetilde{\mathcal{L}}$[CS-DCM(k)[Space(n)]]= $\widetilde{\mathcal{L}}$[CS-DCM($k+1$)[Space(n)]], and therefore the following is true:

$\forall L \in \widetilde{\text{DSPACE}}(\log n)$

$\Rightarrow \exists j(>3k), T_j(L) \in \widetilde{\mathcal{L}}$[CS-DCM(2)[Space($n$)]] (Lemma 5.6)

$\Rightarrow T_j(L) \in \widetilde{\mathcal{L}}$[CS-DCM($k+1$)[Space($n$)]] =$\widetilde{\mathcal{L}}$[CS-DCM($k$)[Space($n$)]]

$\Rightarrow T_{j-1}(L) \in \widetilde{\mathcal{L}}$[CS-DCM($k+1$)[Space($n$)]] =$\widetilde{\mathcal{L}}$[CS-DCM($k$)[Space($n$)]] (Lemma 5.8)

$\Rightarrow \cdots$

$\Rightarrow T_{3k+1}(L) \in \widetilde{\mathcal{L}}$[CS-DCM($k+1$)[Space($n$)]] =$\widetilde{\mathcal{L}}$[CS-DCM($k$)[Space($n$)]] (Lemma 5.8)

$\Rightarrow L \in \widetilde{\mathcal{L}}$[CS-DCM($k(3k+1)+1$)[Space($n$)]] (Lemma 5.7)

Therefore $\mathcal{L}_{\{0\}}$[CS-DCM(k)[Space(n)]]=$\mathcal{L}_{\{0\}}$[CS-DCM($k+1$)[Space(n)]] implies $\widetilde{\text{DSPACE}}$ ($\log n$) $\subseteq \widetilde{\mathcal{L}}$[CS-DCM($k(3k+1)+1$)[Space($n$)]], which is a contradiction to Lemma 5.5. □

5.4 Hierarchies Based on The Bounded Time or Space

In this section, we give the hierarchies based on the bounded time (space) concerning cooperating systems of one-way counter machines.

Lemma 5.9. For each $k \geq 1$,

(1) $\mathcal{L}$[1CM(k)-Time(n^s)] ($\mathcal{L}$[1DCM(k)-Time(n^s)])

$\subsetneq\mathcal{L}$[1CM($k+1$)-Time(n^s)] ($\mathcal{L}$[1DCM($k+1$)-Time(n^s)]), and

(2) $\mathcal{L}$[1CM(k)-Space(n^s)] ($\mathcal{L}$[1DCM(k)-Space(n^s)])

$\subsetneq\mathcal{L}$[1CM($k+1$)-Space(n^s)] ($\mathcal{L}$[1DCM($k+1$)-Space(n^s)]).

Proof : We leave it to the reader, and the reader can refer to the proof of Theorem 5.3.

□

The following lemma is obtained from Theorem 2.4 in [49].

Lemma 5.10. For each $r, s, t \geq 1$,

(1) $\mathcal{L}[1\text{CM}(rt)\text{-Time}(n^s)]$ ($\mathcal{L}[1\text{DCM}(rt)\text{-Time}(n^s)]$)

$\subsetneq \mathcal{L}[1\text{CM}(r+t)\text{-Time}(n^{(r+1)s})]$ ($\mathcal{L}[1\text{DCM}(r+t)\text{-Time}(n^{(r+1)s})]$), and

(2) $\mathcal{L}[1\text{CM}(rt)\text{-Space}(n^s)]$ ($\mathcal{L}[1\text{DCM}(rt)\text{-Space}(n^s)]$)

$\subsetneq \mathcal{L}[1\text{CM}(r+t)\text{-Space}(n^{rs})]$ ($\mathcal{L}[1\text{DCM}(r+t)\text{-Space}(n^{rs})]$).

Theorem 5.5. For each $s \geq 1$ and each $k \geq 9$,

(1) $\mathcal{L}[\text{CS-1CM}(k)[\text{Time}(n^s)]]$ ($\mathcal{L}[\text{CS-1DCM}(k)[\text{Time}(n^s)]]$)

$\subsetneq \mathcal{L}[\text{CS-1CM}(k)[\text{Time}(n^{4s})]]$ ($\mathcal{L}[\text{CS-1DCM}(k)[\text{Time}(n^{4s})]]$), and

(2) $\mathcal{L}[\text{CS-1CM}(k)[\text{Space}(n^s)]]$ ($\mathcal{L}[\text{CS-1DCM}(k)[\text{Space}(n^s)]]$)

$\subsetneq \mathcal{L}[\text{CS-1CM}(k)[\text{Space}(n^{3(k-1)s+3})]]$ ($\mathcal{L}[\text{CS-1DCM}(k)[\text{Space}(n^{3(k-1)s+3})]]$).

Proof : We give the proof only for the nondeterministic case, since the same proof may be applied to the deterministic case. (Note that $2k-1 < 3(k-3)$ if $k \geq 9$.)

(1): $\mathcal{L}[\text{CS-1CM}(k)[\text{Time}(n^s)]]$

$\subseteq \mathcal{L}[1\text{CM}(2k-1)\text{-Time}(n^s)]$ (Theorem 5.2)

$\subsetneq \mathcal{L}[1\text{CM}(3(k-3))\text{-Time}(n^s)]$ (Lemma 5.9)

$\subseteq \mathcal{L}[1\text{CM}(k)\text{-Time}(n^{4s})]$ (Lemma 5.10)

$\subseteq \mathcal{L}[\text{CS-1CM}(k)[\text{Time}(n^{4s})]]$ (Theorem 5.1)

(2): $\mathcal{L}[\text{CS-1CM}(k)[\text{Space}(n^s)]]$

$\subseteq \mathcal{L}[1\text{CM}(2k-1)\text{-Space}(n^{(k-1)s+1})]$ (Theorem 5.2)

$\subsetneq \mathcal{L}[1\text{CM}(3(k-3))\text{-Space}(n^{(k-1)s+1})]$ (Lemma 5.9)

$\subseteq \mathcal{L}[1\text{CM}(k)\text{-Space}(n^{3(k-1)s+3})]$ (Lemma 5.10)

$\subseteq \mathcal{L}[\text{CS-}1\text{CM}(k)[\text{Space}(n^{3(k-1)s+3})]]$ (Theorem 5.1)

□

Remark. It was shown in [49] (Theorem 3.3 (2)) that for each $k \geq 3$, $\mathcal{L}[1\text{CM}(k)\text{-Time}(n^s)]$ $(\mathcal{L}[1\text{DCM}(k)\text{-Time}(n^s)])\subsetneq\mathcal{L}[1\text{CM}(k)\text{-Time}(n^{ks+3})]$ $(\mathcal{L}[1\text{DCM}(k)\text{-Time}(n^{ks+3})])$. In fact, we can improve this result as follows.

Corollary 5.4. For each $s \geq 1$ and each $k \geq 5$,

$\mathcal{L}[1\text{CM}(k)\text{-Time}(n^s)]$ $(\mathcal{L}[1\text{DCM}(k)\text{-Time}(n^s)])$

$\subsetneq \mathcal{L}[1\text{CM}(k)\text{-Time}(n^{3s})]$ $(\mathcal{L}[1\text{DCM}(k)\text{-Time}(n^{3s})])$.

Proof : For each $k' \geq 3$,

$\mathcal{L}[1\text{CM}(k'+2)\text{-Time}(n^s)]$ $(\mathcal{L}[1\text{DCM}(k'+2)\text{-Time}(n^s)])$

$\subsetneq \mathcal{L}[1\text{CM}(2k')\text{-Time}(n^s)]$ $(\mathcal{L}[1\text{DCM}(2k')\text{-Time}(n^s)])$ (Lemma 5.9)

$\subseteq \mathcal{L}[1\text{CM}(k'+2)\text{-Time}(n^{3s})]$ $(\mathcal{L}[1\text{DCM}(k'+2)\text{-Time}(n^{3s})])$ (Lemma 5.10)

□

A similar argument can show the following corollary.

Corollary 5.5. For each $s \geq 1$ and each $k \geq 5$,

$\mathcal{L}[1\text{CM}(k)\text{-Space}(n^s)]$ $(\mathcal{L}[1\text{DCM}(k)\text{-Space}(n^s)])$

$\subsetneq \mathcal{L}[1\text{CM}(k)\text{-Space}(n^{2s})]$ $(\mathcal{L}[1\text{DCM}(k)\text{-Space}(n^{2s})])$.

5.5 One-Way versus Two-Way and Determinism versus Nondeterminism

This section investigates the differences between the accepting powers of CS-CM(k)'s and CS-1CM(k)'s, and between the accepting powers of CS-1CM(k)'s and CS-1DCM(k)'s, where all the machines are polynomially time bounded.

Let $L_1 = \{wcw^R(20^{|w|})^{|w|} | w \in \{0,1\}^*\}$. In [47] it was shown that

$$L_1 \in \mathcal{L}[\text{DCM(1)-Time(linear)}] - \bigcup_{1 \leq k < \infty} \mathcal{L}[\text{1CM}(k)\text{-Time(poly)}].$$

From this fact and Lemma 5.2, Theorem 5.2, we get

Theorem 5.6. For any $c > 1$,

$$\mathcal{L}[\text{CS-DCM(1)[Time}(cn)]] - \bigcup_{1 \leq k < \infty} \mathcal{L}[\text{CS-1CM}(k)[\text{Time(poly)}]] \neq \emptyset.$$

Let $L_2 = \{w_1 1 w_2 | w_1, w_2 \in \{0,1\}^* \ \& \ |w_1| = |w_2|\}$. In [62] it was shown that

$$L_2 \notin \bigcup_{1 \leq k < \infty} \mathcal{L}[\text{1DCM}(k)\text{-Space}(n)].$$

On the other hand, it is easily seen that the language L_2 can be accepted by a CS-1CM(1)[Time(n)]. Thus, from this fact and Theorem 5.2, Corollary 5.3, we get

Theorem 5.7.

$$\mathcal{L}[\text{CS-1CM(1)[Time}(n)]] - \bigcup_{1 \leq k < \infty} \mathcal{L}[\text{CS-1DCM}(k)[\text{Time(poly)}]] \neq \emptyset.$$

Remark. It was shown in [50] that two-way nondeterministic (one-) counter machines are more powerful than deterministic ones. That is, $\mathcal{L}[\text{CS-DCM(1)}] \subsetneq \mathcal{L}[\text{CS-CM(1)}]$.

5.6 Closure Properties

This section investigates closure properties under several operations of the classes of languages accepted by cooperating systems of one-way counter machines with polynomial time bound.

We first examine closure properties for the deterministic case.

Lemma 5.11. For a word w in $1\{0,1\}^*$, let $n(w)$ be the integer represented by w as a binary number. Let

$$L_3 = \{w20^{n(w)} | w \in \{1\}\{0,1\}^*\}.$$

Then, $\overline{L_3} \notin \bigcup_{1 \leq k < \infty} \mathcal{L}[1\text{DCM}(k)\text{-Time(poly)}]$.

Proof : Suppose that there is a 1DCM(k) M, $k \geq 1$, which accepts the language $\overline{L_3}$. For each $n \geq 1$, let

$$V(n) = \{w2 | w \in \{1\}\{0,1\}^* \ \& \ |w| = n\}.$$

For each $x \in V(n)$, let $q(x)$, $c_1(x)$, $c_2(x)$, $\cdots$, $c_k(x)$ denote the internal state and the contents of each counter of M when it first reads the symbol "2" of the input x. For each $n \geq 1$, let

$$S(n) = \{\langle q(x), c_1(x), c_2(x), \cdots, c_k(x)\rangle | x \in V(n)\}.$$

Since M is polynomially time bounded, there is some constant $c > 0$ such that $|S(n)| = O(n^c)$. On the other hand, $|V(n)| = 2^{n-1}$. Thus as n gets large, there are two words $x = w_1 2$ and $y = w_2 2$ in $V(n)$ such that $w_1 \neq w_2$ but $\langle q(x), c_1(x), c_2(x), \cdots, c_k(x)\rangle = \langle q(y), c_1(y), c_2(y), \cdots, c_k(y)\rangle$.

Now consider the following two words:

$$x' = w_1 20^{n(w_1)}, \ y' = w_2 2^{n(w_1)}.$$

Clearly, $y' \in \overline{L_3}$, so y' will be accepted by M. Thus, x' will also be accepted by M. This a contradiction, because x' is not in $\in \overline{L_3}$. Hence $\overline{L_3}$ cannot be in $\bigcup_{1 \leq k < \infty} \mathcal{L}[1\text{DCM}(k)\text{-Time(poly)}]$.

□

It has been shown in [49] that L_3 can be accepted in linear time by a deterministic one-way 2-counter machine. (The machine first converts w to $n(w)$ stored in one counter; this takes time $n(w)$. Then it decreases $n(w)$ from the counter while simultaneously checking that the number of 0's is correct; this takes time $n(w)$. Thus L_3 is in $\mathcal{L}[1\text{DCM}(2)\text{-Time(linear)}]$.) From this fact, Theorem 5.2 and Lemmas 5.2, 5.11, we have

Theorem 5.8. For each $s \geq 1$, $k \geq 2$ and any $c > 1$, neither $\mathcal{L}$[CS-1DCM(k)[Time(cn^s)]] nor $\mathcal{L}$[CS-1DCM(k)[Time(poly)]] is closed under complementation.

Lemma 5.12. For each $k \geq 1$, let $A(k)$ and $A(\infty)$ be the languages defined in Lemma 2.8. Then, for any $k, c, s \geq 1$,

(1) $A(2ks) \notin \mathcal{L}$[CS-1DCM($k$)[Time($cn^s$)]] and

(2) $A(\infty) \notin \bigcup_{1 \leq k < \infty} \mathcal{L}$[CS-1DCM($k$)[Time(poly)]].

Proof : The proof is omitted here, and the reader may refer to the proof of Lemma 2.8 if necessary. □

Theorem 5.9. For each $c, s, k \geq 1$, $\mathcal{L}$[CS-1DCM(k)[Time(cn^s)]] is not closed under union and intersection.

Proof : For each $t \geq 1$ and $1 \leq i \leq t$, let

$$L(t,i) = \{0^{m_1}10^{m_2}1\cdots10^{m_i}20^{m'_i}1\cdots10^{m'_2}10^{m'_1} \mid \forall j(1 \leq j \leq t)[m_j, m'_j \geq 1] \ \&\ m_i = m'_i\},$$

$$A(t,i) = \{0^{m_1}10^{m_2}1\cdots10^{m_i}20^{m_i} \mid \forall j(1 \leq j \leq t)[m_j \geq 1]\}.$$

Then, it is easy to see that

$$a^*L(2ks,i)a^*, A(2ks,i) \in \mathcal{L}[\text{CS-1DCM}(1)[\text{Time}(n)]].$$

On the other hand, by Lemmas 5.3 and 5.12, for any $c \geq 1$,

$a^*L(2ks,1)a^* \cap \cdots \cap a^*L(2ks,2ks)a^*$

$= L'(2ks) \notin \mathcal{L}[1\text{CM}(2k-1)\text{-Time}(cn^s)]$, and

$A(2ks,1) \cup \cdots \cup A(2ks,2ks)$

$= A(2ks) \notin \mathcal{L}$[CS-1DCM($k$)[Time($cn^s$)]].

From this fact and Theorem 5.2, the theorem follows. □

Remark. By Corollary 5.3,

$$\mathcal{L}[\text{CS-1DCM}(3)[\text{Time}(\text{poly})]] = \bigcup_{1 \le k < \infty} \mathcal{L}[\text{CS-1DCM}(k)[\text{Time}(\text{poly})]].$$

From this, it will be obvious that for each $k \geq 3$, $\mathcal{L}$[CS-1DCM(k)[Time(poly)]] is closed under union and intersection.

Lemma 5.13. Let $B(\infty)$ be the language defined in Lemma 2.9. Then,

(1) $B(\infty) \in \mathcal{L}[\text{CS-1DCM}(2)[\text{Time}(n)]]$, and

(2) $B(\infty)^R \notin \bigcup_{1 \le k < \infty} \mathcal{L}[\text{CS-1DCM}(k)[\text{Time}(\text{poly})]]$.

Proof : The proof is omitted here, and the reader may refer to the proof of Lemma 2.9 if necessary. □

Theorem 5.10. For each $c, s \geq 1$ and each $k \geq 2$, neither $\mathcal{L}$[CS-1DCM(k)[Time(cn^s)]] nor $\mathcal{L}$[CS-1DCM(k)[Time(poly)]] is closed under the following operations:

(1) reversal "^{R}",

(2) concatenation,

(3) Kleene closure "*",

(4) nonerasing homomorphism.

Proof : (1): Nonclosure under reversal follows from Lemma 5.13.

(2), (3), (4): See the proof of (2), (3), (4) of Theorem 2.7. □

We next examine closure properties for the nondeterministic case.

Using the same language defined in Lemma 2.10, one can easily show the following theorem.

Theorem 5.11. For each $c, s, k \geq 1$, neither $\mathcal{L}$[CS-1CM(k)[Time(cn^s)]] nor $\mathcal{L}$[CS-1CM(k)[Time(poly)]] is closed under complementation.

Theorem 5.12. For each $c, s, k \geq 1$, $\mathcal{L}$[CS-1CM(k)[Time(cn^s)]] is closed under union, but not under intersection.

Proof : Closure under union is obvious, and the proof for nonclosure under intersection is the same as in the deterministic case (see the proof of Theorem 5.8). □

Remark. By Corollary 5.3,

$$\mathcal{L}[\text{CS-1CM}(3)[\text{Time}(\text{poly})]] = \bigcup_{1 \leq k < \infty} \mathcal{L}[\text{CS-1CM}(k)[\text{Time}(\text{poly})]].$$

From this, it is obvious that for each $k \geq 3$, $\mathcal{L}$[CS-1CM(k)[Time(poly)]] is closed under union and intersection.

By an argument similar to that in the proof of Theorem 2.9, we can get

Theorem 5.13. For each $c, s, k \geq 1$, both $\mathcal{L}$[CS-1CM(k)[Time(cn^s)]] and $\mathcal{L}$[CS-1CM(k)[Time (poly)]] are closed under reversal "^{R}", concatenation and Kleene closure "*".

The closure results obtained above are summarized in Table 5.1, where for each $c > 1$, $s \geq 1$, $k \geq 2$, $\mathcal{L}(cn^s)_k^D = \mathcal{L}$[CS-1DCM($k$)[Time($cn^s$)]], $\mathcal{L}(cn^s)_k^N = \mathcal{L}$[CS-1CM($k$)[Time($cn^s$)]], $\mathcal{L}(\text{poly})_k^D = \mathcal{L}$[CS-1DCM($k$)[Time(poly)]], $\mathcal{L}(\text{poly})_k^N = \mathcal{L}$[CS-1CM($k$)[Time(poly)]], and "$\surd$" means that the class is closed, "×" means that the class is not closed, "?" means that the closure property is not known.

Table 5.1: Closure properties of polynomial time bounded CS-1CM(k), $k \geq 2$

	$\mathcal{L}(cn^s)_k^D$	$\mathcal{L}(cn^s)_k^N$	$\mathcal{L}(\text{poly})_k^D$	$\mathcal{L}(\text{poly})_k^N$
complementation	×	×	×	×
union	×	$\surd$	$\surd(k \geq 3)$	$\surd$
intersection	×	×	$\surd(k \geq 3)$	$\surd$
concatenation	×	$\surd$	×	$\surd$
reversal	×	$\surd$	×	$\surd$
Kleene closure	×	$\surd$	×	$\surd$
nonerasing homomorphism	×	?	×	?

Remark. We can also show that similar results hold for cooperating systems of one-way counter machines with polynomial space bound.

5.7 Concluding Remarks

In this chapter, we have introduced the "cooperating system of counter machines", and analyzed some of its properties. We also investigated a relationship between the accepting powers of cooperating systems of counter machines and multicounter machines with polynomial time (or space) bound.

Some open problems in this chapter are:

(1) $\mathcal{L}$[CS-1CM(2)[Time(poly)]] ($\mathcal{L}$[CS-1DCM(2)[Time(poly)]]) $\subsetneq \mathcal{L}$[CS-1CM(3)[Time(poly)]] ($\mathcal{L}$[CS-1DCM(3)[Time(poly)]])?

(2) Is $\mathcal{L}$[CS-1CM(2)[Time(poly)]] ($\mathcal{L}$[CS-1DCM(2)[Time(poly)]]) closed under intersection? and

(3) Is $\mathcal{L}$[CS-1DCM(2)[Time(poly)]] closed under union?

(4) $\mathcal{L}$[CS-DCM(k)[Time(poly)]]$\subsetneq \mathcal{L}$[CS-CM(k)[Time(poly)]], for each $k \geq 2$?

BIBLIOGRAPHY

1. B. S. Baker and R. V. Book, "Reversal-bounded multipushdown machines", *J. Comput. System Sci.* **8: 3**, (1974) 315–332.

2. Minsky, M. L., "Recursive unsolvability of Post's problem of 'tag' and others topics in the theory of Turing machines", *Annals of Math.*, **74: 3**, (1961) 437–455.

3. McCulloch, W. S. and W. Pitts, "A logical calculus of the ideas immanent in nervous activity", *Bull. Math. Biophysicss* **5**, (1943) 115–133.

4. S. Eilenberg, "Automata, languages, and machines (Volume A)", Academic Press, New York and London, 1974.

5. S. Eilenberg, "Automata, languages, and machines (Volume B)", Academic Press, New York and London, 1974.

6. T. F. Piatkowski, "N-head finite state machines", Ph. D. Thesis, University of Michigan, 1963.

7. A. L. Rosenberg, "On multihead finite automata", *IBM J. Res. Develop.* **10**, (1966) 388–394.

8. A. C. Yao and R. L. Rivest, "$k+1$ heads are better than k", *J. Assoc. Comput. Mach.* **25**, (1978) 337–340.

9. R. W. Floyd, "Review 14, 352 of [37]", *Comput. Rev.* **9**, (1968) 280.

10. J. Hromkovic, "One-way multihead deterministic finite automata", *Acta Inform.* **19**, (1983) 377–384.

11. O. H. Ibarra, S. K. Sahni and C. E. Kim, "Finite automata with multiplication", *Theoret. Comput. Sci.* **2**, (1976) 271–294.

12. O. H. Ibarra and C. E. Kim, "A useful device for showing the solvability of some decision problems", *J. Comput. System Sci.* **13**, (1976) 153–160.

13. K. Inoue, I. Takanami, A. Nakamura and T. Ae, "One-way simple multihead finite automata", *Theoret. Comput. Sci.* **9**, (1979) 311–328.

14. P. Duris and J. Hromkovic, "One-way simple multihead finite automata are not closed under concatenation", *Theoret. Comput. Sci.* **27**, (1983) 121–125.

15. O. H. Ibarra, "Characterizations of some tape and time complexity classes of Turing machines in terms of multihead and auxiliary stack automata", *J. Comput. System Sci.* **5**, (1971) 88–117.

16. J. Hartmanis, "On non-determinacy in simple computing devices", *Acta Inform.* **1**, (1972) 336–344.

17. Z. Galil, "some open problems in the theory of computation as questions about two-way deterministic automata languages", *Math. Systems Theorey* **10**, (1977) 211–228.

18. B. Monien, "Transformational methods and their application to complexity problems", *Acta Inform.* **6**, (1976) 95–108.

19. B. Monien, "The LBA-problem and the deterministic tape complexity of two-way counter languages over a one-letter alphabet", *Acta Inform.* **8**, (1977) 371–382.

20. O. H. Ibarra, "On two-way multihead automata", *J. Comput. System Sci.* **7**, (1973) 28–37.

21. B. Monien, "Two-way multihead automata over a one-letter alphabet", *RAIRO Inform. Theor.* **14**, (1980) 67–82.

22. O. H. Ibarra, S. M. Kim and L. E. Rosier, "Some characterizations of multihead finite automata", *Inform. and Control* **67**, (1985) 114–125.

23. J. Engelfriet, "The power of two-way deterministic checking stack automata", *Inform. and Comput.* **80**, (1989) 114–120.

24. I. H. Sudborough, "Bounded-reversal multihead finite automata languages", *Inform. and Control* **25: 4**, (1974) 317–328.

25. W. J. Savitch and P. M. B. Vitanyi, "On the power of real-time two-way multihead finite automata with jumps", *Inform. Process. Lett.* **19**, (1984) 31–35.

26. M. Blum and C. Hewitt, "Automata on a two-dimensional tape", *IEEE Symp. Switching Automata Theory*, (1967) 155–160.

27. R. W. Ritchie and F. N. Springsteel, "Languages recognition by marking automata", *Inform. and Control* **20**, (1972) 313–330.

28. F. N. Springsteel, "Context-free languages and marking automata", Ph. D. Thesis, University of Wash-

ington, Seattle, wash., 1967.

29. P. Hsia and T. Y. Raymond, "Marker automata", *Inform. Sci.* **20**, (1972) 313–330.

30. M. Sipser, "Halting space-bounded computations", *Theoret. Comput. Sci.* **10**, (1980) 335–338.

31. Y. Wang, K. Inoue and I. Takanami, "Multihead finite automata with markers", *to appear* .

32. L. Budach, "On the solution of the labyrinth problem for finite automata", *EIK* **11/10-12**, (1975) 661–672.

33. L. Budach, "Automata and labyrinths", *Math. Nachrichten* **86**, (1978) 195–282.

34. A. N. Shah, "Pebble automata on arrays", *Computergraphics and Image Processing* **3**, (1974) 236–246.

35. F. Hoffmann, "One pebble does not suffice to search plane labyrinths", *Proc. 10th MFCS, Lecture Notes in Computer Science* **117**, (1981) 433–444.

36. A. Szepietowski, "A finite 5-pebble automaton can search every maze", *Inform. Process. Lett.* **10**, (1982) 199–204.

37. M. Blum and W. Sakoda, "On the capability of finite automata in 2 and 3 dimensional space", *Proc. 17th IEEE FOCS Conf.* (1977) 147–161.

38. H. A. Rollik, "Automata in planar graphs", *Acta Inform.* **13**, (1980) 287–298.

39. A. Hemmerling, "Normed two-plane traps for finite systems of cooperating compass automata", *EIK* **23/8-9**, (1987) 453–470.

40. M. Bull and A. Hemmerling, "Finite embedded trees and simply connected mazes cannot be searched by halting finite automata", *EIK* **23/8-9**, (1987) 453–470.

41. M. Bull and A. Hemmerling, "Traps for jumping multihead counter automata", *EIK* **28/6**, (1992) 343–361.

42. A. O. Buda, "Multiprocessor automata", *Inform. Process. Lett.* **25**, (1987) 257–261.

43. H. Kakugawa, H. Matsuno, K. Inoue and I. Takanami, "Some properties of one-way multiprocessor finite automata", *IEICE Trans. Inf. & Syst.* **j75: 11**, (1992) 963–972 (in Japanese).

44. Y. Wang, K. Inoue and I. Takanami, "Some hierarchy results on multihead automata over a one-letter alphabet", *IEICE Trans. Inf. & Syst.* **E76: 6**, (1993) 625–633.

45. J. W. Hong, "On similarity and duality of computation", *Proc. 21st IEEE Symposium on Foundations of Computer Science*, (1980) 348–359.

46. N. Pippenger, "On simultaneous resource bounds", *Proc. 20th IEEE Symposium on Foundations of Computer Science*, (1979) 307–311.

47. J. Hromkovic, "Fooling a two-way nondeterministic multihead automaton with reversal number restriction", *Acta Inform.* **22**, (1985) 589–594.

48. S. Ginsburg, "Algebraic and automata-theoretic properties of formal languages", *North-Holland, Amsterdam*, 1975.

49. S. A. Greibach, "Remarks on the complexity of nondeterministic counter languages", *Theoret. Comput. Sci.* **1**, (1976) 269–288.

50. M. Chrobak, "Variations on the technique of Duris and Galil", *J. Comput. System Sci.* **30**, (1985) 77–85.

51. P. C. Fischer, A. R. Meyer and A. L. Rosenberg, "Counter machines and counter languages", *Math. Systems Theorey* **2**, (1968) 265–283.

52. S. A. Greibach, "Erasable context-free languages", *Inform. and Control* **29**, (1975) 301–326.

53. M. A. Harrison and O. H. Ibarra, "Multi-tape and multi-head pushdown automata", *Inform. and Control* **13**, (1963) 433–470.

54. S. Ginsburg and H. G. Rice, "Two families of languages telated to ALGOL", *J. ACM* **9**, (1962) 350–371.

55. O. H. Ibarra, "Restricted one-counter machines with undecidable universe problems", *Math. Systems Theory* **13**, (1979) 181–186.

56. L. G. Valiant, "Deterministic one-counter automata", *J. Comput. System Sci.* **10**, (1975) 340–350.

57. P. Duris and J. Hromkovic, "Zerotesting bounded one-way multicounter machines", *Kybernetika* **23: 1**, (1987) 13–18.

58. K. Inoue, I. Takanami and H. Taniguchi, "Three-way two-dimensional simple multihead finite automata—hierarchical properties", *IEICE Trans.* **62-D/2**, (1979) 65–72 (in Japanese).

59. M. Sakamoto, I. Sakuramoto and K. Inoue, "A space lower-bound technique for two-dimensional alternat-

ing Turing machines", *IEICE Trans.* **J74-D-I/4**, (1991) 311–314 (in Japanese).

60. K. Inoue, I. Takanami and H. Taniguchi, "Three-way two-dimensional simple multihead finite automata—closure properties", *IEICE Trans.* **62-D/4**, (1979) 273–280 (in Japanese).

61. K. Inoue and I. Takanami, "Three-way two-dimensional multicounter automata", *Inform. Sci.* **19**, (1979) 1–20.

62. A. Szepietowski, "Some remarks on two-dimensional finite automata", to appear in *Inform. Sci.*.

63. A. Szepitowski, "On three-way two-dimensional multicounter automata", *Inform. Sci.* **55**, (1979) 35–47.

64. K. Inoue and I. Takanami, "On-line n-bounded multicounter automata", *Inform. Sci.* **17**, (1979) 239–251.

論文目録

論文題目	公表の方法及び時期	共著者	本論文に関連する章
1. コオペレーティング1方向有限オートマタシステムのある性質	電子情報通信学会論文誌D-I Vol. J75-D-I, pp.391-399, 1992年7月。	井上克司 高浪五男	Chapter 2
2. Some Hierarchy Results on Multihead Automata over a One-Letter Alphabet	IEICE TRANS. INF. & SYST., Vol. E76-D, No. 6, pp.625-633 (1993, 6).	K. Inoue I. Takanami	Chapter 2 Chapter 4 Chapter 5
3. A Note on One-Way Multicounter Machines and Cooperating Systems of One-Way Finite Automata	IEICE TRANS. INF. & SYST., Vol. E76-D, No. 10, pp.1302-1306 (1993, 10).	K. Inoue I. Takanami	Chapter 3 Chapter 5
4. Multihead Finite Automata with Markers	to appear in IEICE TRANS. INF. & SYST.	K. Inoue I. Takanami	
5. コオペレーティング1方向カウンタ機械システム	電子情報通信学会論文誌D-I（掲載予定）	井上克司 高浪五男	Chapter 5