Vol. 36 No. 12 Transactions of Information Processing Society of Japan Dec. 1995

Regular Paper

A Note on Alternating Multi-Counter Automata with Small Space

TSUNEHIRO YOSHINAGA† and KATSUSHI INOUE††

This paper introduces space-bounded alternating multi-counter automata, and shows some fundamental properties of these automata. We first investigate relationships (1) between strong and weak space-bounds, (2) between one-way and two-way, and (3) among determinism, nondeterminism, and full alternation. Let *weak*-2ACA$(k, L(n))$ (*weak*-1ACA$(k, L(n))$) denote the class of sets accepted by weakly $L(n)$ space-bounded two-way (one-way) alternating k-counter automata. We show, for example, that *weak*-2ACA$(1, \log n)$ $-$ *weak*-1ACA $(k, L(n)) \neq \phi$ for any $k \geq 1$ and any $L(n)$ such that $\lim_{n\to\infty}[\log L(n)/\log n] = 0$. We then investigate a relationship between the accepting powers of multi-counter and multi-head finite automata. Let 2AFA(k) denote the class of sets accepted by two-way alternating k-head finite automata. We show, for example, that *weak*-2ACA(k, n) = 2AFA$(k+1)$ for each $k \geq 1$. We finally investigate hierarchical properties based on the number of counters. Let *weak*-1UCA $(k, L(n), \text{real})$ denote the class of sets accepted by weakly $L(n)$ space-bounded one-way alternating k-counter automata with only universal states which operate in realtime. We show, for example, that *weak*-1UCA$(k+1, \log n, \text{real})$ $-$ *weak*-1ACA$(k, \log n) \neq \phi$ for each $k \geq 1$.

1. Introduction

Alternating Turing machines were introduced in Ref. 1) as a mechanism to model parallel computation, and in related papers 2)~12), investigations of alternation have continued. Recently, several properties of alternating Turing machines with small space bounds were given in Refs. 5), 6), 10)~12). It is well known that without time or space limitations, counter automata have the same power as Turing machines; however, when time restrictions are applied, a different situation emerges. For example, hierarchical properties in the accepting powers of one-way alternating multi-counter automata operating in realtime are investigated in Refs. 7), 8), 18). On the other hand, properties of alternating multi-counter automata with small space (especially, with sublinear space) are little investigated as far as we know.

In this paper, we are interested in obtaining some fundamental properties of alternating multi-counter automata with small space. We mainly investigate the effects (i) of the amount of space and (ii) of the number of counters used in computations on the accepting powers of alternating multi-counter automata with sublinear space, and investigate the differences (in the accepting powers of these automata) (i) between weakly and strongly space-bounded computations, (ii) between one-way and two-way computations and (iii) among deterministic, nondeterministic and alternating computations. Section 2 gives the definitions and notations necessary for this paper. Section 3 investigates a relationship between the accepting powers of alternating multi-counter automata with strong and weak space-bounds. Let *strong*-2ACA$(k, L(n))$ (*weak*-2DCA$(k, L(n))$) denote the class of sets accepted by strongly (weakly) $L(n)$ space-bounded two-way alternating (deterministic) k-counter automata, and let *strong*-2ATM$(L(n))$ (*weak*-2DTM$(L(n))$) denote the class of sets accepted by strongly (weakly) $L(n)$ space-bounded two-way alternating (deterministic) Turing machines. From the results in Refs. 10), 14), it follows that *weak*-2DTM$(\log\log n)$ $-$ *strong*-2ATM$(L(n)) \neq \phi$ for any $L(n)$ such that $\lim_{n\to\infty}[L(n)/\log n] = 0$. We show, for example, that *weak*-2DCA$(4, \log n)$ $-$ *strong*-2ACA$(k, L(n)) \neq \phi$ for any $k \geq 1$, and any $L(n)$ such that $\lim_{n\to\infty}[\log L(n)/\log n] = 0$. Section 4 investigates a relationship between one-way and two-way. Let *weak*-1ACA$(k, L(n))$ (*weak*-2ACA $(k, L(n))$) denote the class of sets accepted by weakly $L(n)$ space-bounded one-way (two-way) alternating k-counter automata, and *weak*-1ATM$(L(n))$ (*weak*-2ATM$(L(n))$) denote the

† Department of Information and Electronics Engineering, Tokuyama College of Technology
†† Department of Computer Science and Systems Engineering, Faculty of Engineering, Yamaguchi University

Table 1 Strong versus weak space-bounds.

(1)	*weak*-2DTM($\log\log n$) − *strong*-2ATM($L(n)$) $\neq \phi$	(Refs. 10), 14))
	weak-2DCA($4, \log n$) − *strong*-2ACA($k, L(n)$) $\neq \phi$	(Theorem 3.2)
(2)	*weak*-1NTM($\log\log n$) − *strong*-2ATM($L(n)$) $\neq \phi$	(Refs. 10), 13))
	weak-1NCA($3, \log n$) − *strong*-2ACA($k, L(n)$) $\neq \phi$	(Theorem 3.2)
(3)	*weak*-1UTM($\log\log n$) − *strong*-1ATM($L(n)$) $\neq \phi$	(Refs. 5), 11))
	weak-1UCA($1, \log n$, real) − *strong*-1ACA($k, L(n)$) $\neq \phi$	(Theorem 3.2)

class of sets accepted by weakly $L(n)$ space-bounded one-way (two-way) alternating Turing machines. It is shown in Ref. 6) that *weak*-2ATM($\log\log n$) − *weak*-1ATM($L(n)$) $\neq \phi$ for any $L(n)$ such that $\lim_{n\to\infty}[L(n)/\log n] = 0$, and in Ref. 2) that *weak*-1ATM($L(n)$) = *weak*-2ATM($L(n)$) for any $L(n) \geq \log n$. We show, for example, that *weak*-2ACA($1, \log n$) − *weak*-1ACA($k, L(n)$) $\neq \phi$ for any $k \geq 1$, and any $L(n)$ such that $\lim_{n\to\infty}[\log L(n)/\log n] = 0$, and that *weak*-2ACA($k, L(n)$) $\subseteq$ *weak*-1ACA($k+1, L(n)$) for each $k \geq 1$ and each $L(n) \geq n$. Section 5 investigates a relationship among determinism, nondeterminism, and full alternation. Let *weak*-1ACA($k, L(n)$, real) (*weak*-1UCA($k, L(n)$, real)) denote the class of sets accepted by weakly $L(n)$ space-bounded one-way alternating k-counter automata (alternating k-counter automata with only universal states) operating in realtime. It is shown in Ref. 6) that *weak*-1ATM($\log\log n$) − *weak*-2DTM($L(n)$) $\neq \phi$ for any $L(n)$ such that $\lim_{n\to\infty}[L(n)/\log n] = 0$. We show, for example, that *weak*-1ACA($1, \log n$, real) − *weak*-2DCA($k, L(n)$) $\neq \phi$ for any $k \geq 1$, and any $L(n)$ such that $\lim_{n\to\infty}[\log L(n)/\log n] = 0$. Section 6 investigates a relationship between the accepting powers of alternating multi-counter and multi-head automata. Let 2AFA(k) denote the class of sets accepted by two-way alternating k-head finite automata. In this section, we show that *weak*-2ACA(k, n) = 2AFA($k+1$) for each $k \geq 1$. Section 7 investigates hierarchical properties based on the number of counters. In Ref. 8), it is shown that for each $k \geq 1$, there is a set accepted by a one-way alternating $(k+1)$-counter automata with only universal states operating in realtime, but not accepted by any one-way alternating k-counter automata operating in realtime. We show that for each $k \geq 1$, *weak*-1UCA($k+1, \log n$, real) − *weak*-1ACA($k, \log n$) $\neq \phi$. Section 8 concludes this paper by giving several open problems.

Tables 1 through **5** give a summary of our results. We summerize, in Tables 1, 2 and 3, the results of Turing machines shown in Refs. 2), 5), 6), 10), 11), 13), 14) and our corresponding results of multi-counter automata. In all the entries in the tables, k denotes any positive integer, c (> 0) denotes any constant, and $L(n)$ denotes the function such that $\lim_{n\to\infty}[L(n)/\log n] = 0$ for Turing machines and $\lim_{n\to\infty}[\log L(n)/\log n] = 0$ for multi-counter automata except for (4) of Table 2, in which $L(n) \geq \log n$ for Turing machines and $L(n) \geq n$ for multi-counter automata.

2. Preliminaries

A multi-counter automaton is a multi-pushdown automaton whose pushdown stores operate as counters, i.e., each storage tape is a pushdown tape of the form Z^i (Z fixed). (See Refs. 17), 18) for formal definitions of multi-counter automata.)

A *two-way alternating multi-counter automaton* (2amca) M is the generalization of a two-way nondeterministic multi-counter automaton in the same sense as in Refs. 1)∼3). That is, the state set of M is divided into two disjoint sets, the set of *universal* states and the set of *existential* states. Of course, M has a specified set of *accepting* states.

We assume that 2amca's have the left endmarker "¢" and the right endmarker "$" on the input tape, read the input tape right or left, and can enter an accepting state only when falling off the right endmarker $. We also assume that in one step 2amca's can increment or decrement the contents (i.e., the length) of each counter by at most one. For each $k \geq 1$, we denote a two-way alternating k-counter automaton by 2aca(k).

An *instantaneous description* (ID) of 2aca(k) M is an element of

$$\Sigma^* \times (N \cup \{0\}) \times S_M,$$

where Σ ($\not\in, \$ \notin \Sigma$) is the input alphabet of M,

Table 2 One-way versus two-way.

(1)	*weak*-2UTM($\log\log n$) − *weak*-1UTM($L(n)$) $\neq \phi$	(Ref. 5))
	strong-2DCA($1, \log n$) − *weak*-1UCA($k, L(n)$) $\neq \phi$	(Theorem 4.1)
(2)	*weak*-2ATM($\log\log n$) − *weak*-1ATM($L(n)$) $\neq \phi$	(Ref. 6))
	strong-2ACA($1, \log n$) − *weak*-1ACA($k, L(n)$) $\neq \phi$	(Theorem 4.1)
(3)	*weak*-2DTM($\log\log n$) − *weak*-1NTM($L(n)$) $\neq \phi$	(Ref. 6))
	strong-2DCA($1, \log n$) − *weak*-1NCA($k, L(n)$) $\neq \phi$	(Theorem 4.1)
	strong-2DCA($1, \log n$) − *strong*-1ACA($k, L(n)$) $\neq \phi$	(Theorem 4.1)
(4)	m-2ATM($L(n)$) = m-1ATM($L(n)$)	(Ref. 2))
	m-2ACA($k, L(n)$) $\subseteq$ m-1ACA($k+1, L(n)$) for each m $\in$ {*weak*, *strong*}	(Theorem 4.2)

Table 3 Relationship among determinism, nondeterminism and alternation.

weak-1ATM($\log\log n$) − *weak*-2YTM($L(n)$) $\neq \phi$	(Ref. 6))
weak-1ACA($1, \log n$, real) − *weak*-2YCA($k, L(n)$) $\neq \phi$	(Theorem 5.1)
strong-2ACA($1, \log n$) − *weak*-2YCA($k, L(n)$) $\neq \phi$	(Theorem 5.1)
for each Y$\in$\{U,N,D\}	

Table 4 Relationship between multi-head and multi-counter.

weak-2ACA(k, n) = 2AFA($k+1$)	(Theorem 6.1)

Table 5 Hierarchy results based on the number of counters.

weak-1UCA($k+1, \log n$, real) − *weak*-1ACA($k, \log n$) $\neq \phi$	(Theorem 7.1)
weak-2ACA(k, cn) $\subsetneq$ *weak*-2ACA($k+1, cn$)	(Corollary 7.2)
weak-1ACA(k, cn) $\subsetneq$ *weak*-1ACA($k+2, cn$)	(Corollary 7.3)

N denotes the set of all positive integers, and $S_M = Q \times (\{Z\}^*)^k$, where Q is the set of states of the finite control of M, and Z is the storage symbol of M. The first and second components, w and i, of an ID $I = (w, i, (q, (\alpha_1, \ldots, \alpha_k)))$ represent the input string and the input head position, respectively.☆ The third component $(q, (\alpha_1, \ldots, \alpha_k))$ of I represents the state of the finite control and the contents of the k counters. An element of S_M is called a *storage state* of M. If q is the state associated with an ID I, then I is said to be a *universal* (*existential*, *accepting*) ID if q is a universal (existential, accepting) state. The *initial* ID of M on $w \in \Sigma^*$ is $I_M(w) = (w, 0, (q_0, (\lambda, \ldots, \lambda)))$, where q_0 is the initial state of M and λ denotes the empty string.

We write $I \vdash_M I'$ and say I' is a *successor* of I if an ID I' follows from an ID I in one step, according to the transition function of M.

A *computation path* of M on input w is a sequence $I_0 \vdash_M I_1 \vdash_M \ldots \vdash_M I_n$ $(n \geq 0)$, where $I_0 = I_M(w)$. A *computation tree* of M is a finite, nonempty labeled tree with the following properties:

1. each node π of the tree is labeled with an ID, $\ell(\pi)$,
2. if π is an internal node (a non-leaf) of the tree, $\ell(\pi)$ is universal and $\{I | \ell(\pi) \vdash_M I\} = \{I_1, I_2, \ldots, I_r\}$, then π has exactly r children $\rho_1, \rho_2, \ldots, \rho_r$ such that $\ell(\rho_i) = I_i$, and
3. if π is an internal node of the tree and

☆ We note that $0 \leq i \leq |w| + 2$, where for any string v, $|v|$ denotes the length of v. "0", "1", "$|w|+1$" and "$|w|+2$" represent the positions of the left endmarker ¢, the leftmost symbol of w, the right endmarker \$, and the immediate right to \$, respectively.

$\ell(\pi)$ is existential, then π has exactly one child ρ such that $\ell(\pi) \vdash_M \ell(\rho)$.

A *computation tree of M on input w* is a computation tree of M whose root is labeled with $I_M(w)$. An *accepting* computation tree of M on w is a computation tree of M on w whose leaves are all labeled with accepting ID's. We say that M *accepts* w if there is an accepting computation tree of M on w. We denote the set of input words accepted by M by $T(M)$.

A *one-way* alternating multi-counter automaton (1amca) is a 2amca which reads the input tape from left to right only.

For each $k \geq 1$, let 1aca(k) denote a one-way alternating k-counter automaton. We denote by 2uca(k) (1uca(k)) a 2aca(k) (1aca(k)) with only universal states, i.e., with no existential state. A two-way (one-way) *nondeterministic* k-counter automaton, denoted by 2nca(k) (1nca(k)), is a 2aca(k) (1aca(k)) which has no universal state. Further, let 2dca(k) (1dca(k)) denote a two-way (one-way) *deterministic* k-counter automaton which is a 2uca(k) (1uca(k)) whose ID's each have at most one successor.

Let $L : N \to R$ be a function, where R denotes the set of all non-negative real numbers. For each x$\in${1,2}, each y$\in${a,u,n,d} and each $k \geq 1$, an xyca(k) M is *weakly* (*strongly*) $L(n)$ *space-bounded* if for any $n \geq 1$ and any input w of length n accepted by M, there is an accepting computation tree t of M on w such that for each node π of t, the length of each counter in $\ell(\pi)$ is bounded by $L(n)$ (if for any $n \geq 1$ and any input w of length n (accepted or not), and each node π of any computation tree of M on w, the length of each counter in $\ell(\pi)$ is bounded by $L(n)$). A weakly (strongly) $L(n)$ space-bounded xyca(k) is denoted by *weak*-xyca($k, L(n)$) (*strong*-xyca($k, L(n)$)).

Let $T : N \to N$ be a function. For each m$\in${*weak*, *strong*}, each x$\in${1,2}, each y$\in${a,u, n,d}, each $k \geq 1$, and any function $L : N \to R$, we say that an m-xyca($k, L(n)$) M *operates in time* $T(n)$ if for each input w accepted by M, there is an accepting computation tree t of M on w such that the length of each computation path of t is at most $T(|w|)$. An m-1yca($k, L(n)$) M *operates in realtime* if $T(n) = n+1$. For operating time, we are only interested in realtime in this paper. For each m$\in${*weak*, *strong*} and each x$\in${1,2}, we define

$$\text{m-xACA}(k, L(n)) = \{L \mid L = T(M) \text{ for some m-xaca}(k, L(n))M\},$$

$$\text{m-xUCA}(k, L(n)) = \{L \mid L = T(M) \text{ for some m-xuca}(k, L(n))M\},$$

$$\text{m-xNCA}(k, L(n)) = \{L \mid L = T(M) \text{ for some m-xnca}(k, L(n))M\},$$

$$\text{m-xDCA}(k, L(n)) = \{L \mid L = T(M) \text{ for some m-xdca}(k, L(n))M\},$$

$$\textit{weak}\text{-1ACA}(k, L(n), \text{real}) = \{L \mid L = T(M) \text{ for some } \textit{weak}\text{-1aca}(k, L(n)) \; M \text{ operating in realtime}\}, \text{ and}$$

$$\textit{weak}\text{-1UCA}(k, L(n), \text{real}) = \{L \mid L = T(M) \text{ for some } \textit{weak}\text{-1uca}(k, L(n)) \; M \text{ operating in realtime}\}.$$

An alternating Turing machine (aTm) we consider in this paper has a read-only input tape with the left endmarker ¢ and the right endmarker \$, and a storage tape. The reader is referred to Refs. 5), 6) for the formal definition of aTm's. We denote an aTm with only universal states by uTm, a nondeterministic Turing machine by nTm, and a deterministic Turing machine by dTm. For any $L : N \to R$ and any y$\in${a,u,n,d}, we denote a weakly (strongly) $L(n)$ space-bounded one-way yTm by *weak*-1yTm ($L(n)$) (*strong*-1yTm($L(n)$)), and we denote a weakly (strongly) $L(n)$ space-bounded two-way yTm by *weak*-2yTm($L(n)$) (*strong*-2yTm ($L(n)$)). (See Refs. 6), 11), 12), 15), 16) for the definition of *weakly* (*strongly*) $L(n)$ space-bounded aTm's.) For each m$\in${*weak*, *strong*} and each x$\in${1,2}, we define

$$\text{m-xATM}(L(n)) = \{L \mid L = T(M) \text{ for some m-xaTm}(L(n))M\},$$

$$\text{m-xUTM}(L(n)) = \{L \mid L = T(M) \text{ for some m-xuTm}(L(n))M\},$$

$$\text{m-xNTM}(L(n)) = \{L \mid L = T(M) \text{ for some m-xnTm}(L(n))M\}, \text{ and}$$

$$\text{m-xDTM}(L(n)) = \{L \mid L = T(M) \text{ for some m-xdTm}(L(n))M\}.$$

The following lemmas can be easily proved.

Lemma 2.1. For each m$\in${*weak*, *strong*}, each x$\in${1,2}, each Y$\in${A,U,N,D}, each $k \geq 1$, and any function $L : N \to R$,

$$\text{m-xYCA}(k, L(n)) \subseteq \text{m-xYTM}(\log L(n)).^{\star}$$

However, we do not know whether or not $\bigcup_{1 \leq k < \infty} \text{m-xYCA}(k, L(n)) \subsetneq \text{m-xYTM}(\log L(n))$.

It is shown in Ref. 12) that for any func-

* From now on, logarithms are base 2.

tion $L : N \to R$ such that $\lim_{n\to\infty}[L(n)/\log\log n] = 0$,

weak-2ATM($L(n)$) is the class of regular sets.

From this fact and Lemma 2.1, we can show that for any $k \geq 1$, and any function $L : N \to R$ such that $\lim_{n\to\infty}[\log L(n)/\log\log n] = 0$,

weak-2ACA($k, L(n)$) is the class of regular sets.

However, we do not know whether or not for any $k \geq 1$, and any function $L : N \to R$ such that $\lim_{n\to\infty}[L(n)/\log n] = 0$,

weak-2ACA($k, L(n)$) is the class of regular sets.

Lemma 2.2. For any m$\in\{$*weak*, *strong*$\}$, any x$\in\{$1,2$\}$, any Y$\in\{$A,U,N,D$\}$, any $k \geq 1$, any function $L : N \to R$, and any constant c,

m-xYCA($k, cL(n)$) = m-xYCA($k, L(n)$).

3. Strong versus Weak Space-Bounds

In this section, we investigate a relationship between the accepting powers of strongly and weakly space-bounded 2amca's and 1amca's. This investigation is based on the results of Turing machines. Throughout this section, let

$L_1 = \{a^m b^n | m \neq n\}$, and

$L_2 = \{B(1)\sharp B(2)\sharp \ldots \sharp B(n) | n \geq 2\}$,

where for each positive integer $i \geq 1$, $B(i)$ denotes the string in $\{0,1\}^+$ that represents the integer i in binary notation (with no leading zeros).

Lemma 3.1.

(1) $L_1 \in$ *weak*-1NTM($\log\log n$),

(2) $L_1 \in$ *weak*-2DTM($\log\log n$), and

(3) $L_1 \notin$ *strong*-2ATM($L(n)$) for any function $L : N \to R$ such that $\lim_{n\to\infty}[L(n)/\log n] = 0$.

Proof. (1), (2), and (3) are shown in Refs. 13), 14), and 10), respectively. □

Lemma 3.2.

(1) $L_2 \in$ *weak*-1UTM($\log\log n$), and

(2) $L_2 \notin$ *strong*-1ATM($L(n)$) for any function $L : N \to R$ such that $\lim_{n\to\infty}[L(n)/\log n] = 0$.

The proof of (1): By using the technique in Ref. 5), we can easily construct a *weak*-1uTm($\log\log n$) which accepts L_2.

The proof of (2): It is shown in Ref. 11) that for any function $L : N \to R$ such that $\lim_{n\to\infty}[L(n)/\log n] = 0$, any *strong*-1aTm($L(n)$) M can only accept a reguler set. On the other hand, L_2 is not reguler. From these facts, (2) follows. □

From Lemmas 3.1 and 3.2, we have the following:

Theorem 3.1. For any function $L : N \to R$ such that $\lim_{n\to\infty}[L(n)/\log n] = 0$,

(1) *weak*-2DTM($\log\log n$)

– *strong*-2ATM($L(n)$) $\neq \phi$,

(2) *weak*-1NTM($\log\log n$)

– *strong*-2ATM($L(n)$) $\neq \phi$, and

(3) *weak*-1UTM($\log\log n$)

– *strong*-1ATM($L(n)$) $\neq \phi$.

Corollary 3.1. Let $L : N \to R$ be a function such that $L(n) \geq \log\log n$ and $\lim_{n\to\infty}[L(n)/\log n] = 0$. Then,

(1) *strong*-2YTM($L(n)$) $\subsetneq$ *weak*-2YTM($L(n)$) for each Y$\in\{$A,U,N,D$\}$, and

(2) *strong*-1YTM($L(n)$) $\subsetneq$ *weak*-1YTM($L(n)$) for each Y$\in\{$A,U,N$\}$.

It is unknown whether or not *strong*-1DTM($L(n)$) $\subsetneq$ *weak*-1DTM($L(n)$) for any function $L : N \to R$ such that $L(n) \geq \log\log n$ and $\lim_{n\to\infty}[L(n)/\log n] = 0$.

In correspondence to Theorem 3.1 and Corollary 3.1, we investigate a relationship between strong and weak space-bounds of 1amca's and 2amca's. To do so, we need the following lemma, which is shown in Ref. 15).

Lemma 3.3. For any integers m and n, if $m \neq n$, then there exists an integer $r \leq c\times\log(m+n)$ such that $m \neq n \pmod r$, where c is a constant.

We first give two key lemmas.

Lemma 3.4.

(1) $L_1 \in$ *weak*-2DCA($4, \log n$),

(2) $L_1 \in$ *weak*-1NCA($3, \log n$), and

(3) $L_1 \notin$ *strong*-2ACA($k, L(n)$) for any $k \geq 1$, and any function $L : N \to R$ such that $\lim_{n\to\infty}[\log L(n)/\log n] = 0$.

The proof of (1): We consider a 2dca(4) M which acts as follows. Suppose that an input string $w = \phi a^m b^n \$$ is presented to M. (Input strings in a form different from the above can easily be rejected by M.) Based on Lemma 3.3, M tries one by one each natural number to look for r ($\leq c \times \log(m+n)$) such that $m \neq n \pmod r$, by using the following algorithm. Let C_1, C_2, C_3 and C_4 be the counters of M. We denote by "$x \bmod y$" the remainder of x devided by y for any positive integers x and y. In order to check that $m \neq n \pmod r$, M stores $Z^{m \bmod r}$ in C_1, and $Z^{n \bmod r}$ in C_2. C_4 is used in order to store Z^r, and C_3 is used in order to restore the contents of other counters. M first sets r to two as the initial value, and starts the algorithm by storing $Z^r (= Z^2)$ in C_4.

(i) M stores Z^r in C_1 and C_3 by using C_4, and restores Z^r in C_4 by using C_3. Then,

by using C_1 and C_3, M calculates m mod r while reading a^m of w, and stores Z^{r_m} in C_1, where $r_m = m \bmod r$.

(ii) In the same way as in (i) above, M stores Z^{r_n} in C_2 after reading b^n of w, where $r_n = n \bmod r$.

(iii) M checks whether $r_m = r_n$. If $r_m \neq r_n$, then this algorithm completes successfully. Otherwise, M adds Z further in C_4 (i.e., r is incremented by one), moves the input head to the left endmarker $\not\in$, and iterates (i)–(iii) until

(a) M finds out that $r_m \neq r_n$, in which case M accepts w, or

(b) M finds out that $r > m$, in which case M rejects w.

From Lemma 3.3, it will be obvious that M is a weakly $c \log n$ space-bounded 2dca(4) accepting L_1. From this and Lemma 2.2, (1) follows.

The proof of (2): We consider a 1nca(3) M which acts as follows. Let C_1, C_2 and C_3 be the counters of M. For a presented input string $w = \not a^m b^n \$$, M existentially guesses some r, stores Z^r in C_1 and C_2, checks whether $m \neq n \pmod r$ by using the algorithm in the proof of (1) (using C_3 to restore the contents of other counters), and accepts w if $m \neq n \pmod r$. Again from Lemma 3.3, it will be obvious that M is a weakly $c \log n$ space-bounded 1nca(3) accepting L_1. From this and Lemma 2.2, (2) follows.

The proof of (3): (3) follows from Lemma 2.1 and Lemma 3.1 (3). □

Lemma 3.5.

(1) $L_2 \in$ *weak*-1UCA$(1, \log n, \text{real})$, and

(2) $L_2 \in$ *strong*-2DCA$(1, \log n)$,

and for any $k \geq 1$, and any function $L : N \to R$ such that $\lim_{n\to\infty}[\log L(n)/\log n] = 0$,

(3) $L_2 \notin$ *weak*-1NCA$(k, L(n))$, and

(4) $L_2 \notin$ *strong*-1ACA$(k, L(n))$.

The proof of (1): L_2 is accepted by a realtime *weak*-1uca$(1, \log n)$ M which acts as follows. M essentially uses the algorithm in Ref. 5). Suppose that an input string $w = y_1 \sharp y_2 \sharp \ldots \sharp y_n$, where each y_i $(1 \leq i \leq n)$ is in $\{1\}\{0,1\}^*$, is presented to M. (Input strings in different form from the above can easily be rejected by M.) By using universal branches and only one counter, M can check in a way described below whether $y_i = B(i)$ for each i $(1 \leq i \leq n)$. M compares y_i with y_{i+1}, and verifies that y_{i+1} represents in binary notation (with no leading zeros) an integer which is one more than the integer represented by y_i in binary notation (with no leading zeros). In doing so, M will compare the j-th symbol of y_i and y_{i+1}, for all appropriate j. Observe that if y_{i+1} is one more than y_i, then (i) $y_{i+1} = x10^m$ and $y_i = x01^m$, where x is a string (starting with 1) over $\{0,1\}$ and m is some non-negative integer, or (ii) $y_{i+1} = 10^m$ and $y_i = 1^m$, where m is some positive integer.

Let C be the counter of M. For each j $(1 \leq j \leq |y_i|)$, M stores the symbol $y_i(j)$* in its finite control and Z^j in C just after it has read the symbol $y_i(j)$, and makes a universal branch.

(a) In one branch, it compares $y_i(j)$ with the symbol $y_{i+1}(j)$ using Z^j stored in C, and checks whether both the symbols satisfy (i) or (ii) above. (It determines whether they should be the same or not by scanning the remaining symbols of y_i. If they are all 1, then $y_i(j)$ and $y_{i+1}(j)$ should not be the same; otherwise, $y_i(j)$ and $y_{i+1}(j)$ should be the same.)

(b) In another branch, it reads the next symbol $y_i(j+1)$, stores it in the finite control, and adds Z to C in order to store Z^{j+1} in C.

In this way, M can check that y_{i+1} is one more than y_i $(1 \leq i \leq n)$. It will be obvious that if the input string $w = y_1 \sharp y_2 \sharp \ldots \sharp y_n$ is such a string that $y_i = B(i)$ for each i $(1 \leq i \leq n)$ (i.e., $w \in L_2$), then the length of C is bounded by $\log n$. Furthermore, it is clear that M operates in realtime. Thus, M is desired machine accepting L_2.

The proof of (2): L_2 is accepted by a *strong*-2dca$(1, \log n)$ M which acts as follows. Suppose that an input string described in the proof of (1) is presented to M. As in the proof of (1), for all the successive two strings y_i and y_{i+1} in the input, M compares y_i and y_{i+1} symbol by symbol, and verifies that y_{i+1} represents in binary notation an integer which is one more than that represented by y_i in binary notation. By using a technique similar to that in the poof of (1), M checks by scanning back and forth instead of performing universal branch that the above holds. It is clear that for any input w, M accepts or rejects w in a way described above without exceeding more than $\log |w|$ space. Therefore, M is strongly $\log n$ space-bounded.

The proof of (3): It is shown in Ref. 11) that L_2 is not in *weak*-1NTM$(L(n))$ for any function

* For each string w, $w(i)$ denotes the i-th symbol (from the left) of w.

$L : N \to R$ such that $\lim_{n\to\infty}[L(n)/\log n] = 0$.
(3) follows from this fact and Lemma 2.1.
The proof of (4): (4) follows from Lemma 3.2 (2) and Lemma 2.1. □

Theorem 3.2. For any $k \geq 1$, and any function $L : N \to R$ such that $\lim_{n\to\infty}[\log L(n)/\log n] = 0$,
(1) *weak*-2DCA$(4, \log n)$
– *strong*-2ACA$(k, L(n)) \neq \phi$,
(2) *weak*-1NCA$(3, \log n)$
– *strong*-2ACA$(k, L(n)) \neq \phi$, and
(3) *weak*-1UCA$(1, \log n, \text{real})$
– *strong*-1ACA$(k, L(n)) \neq \phi$.

Proof. (1) follows from Lemma 3.4 (1) and (3). (2) follows from Lemma 3.4 (2) and (3). And, (3) follows from Lemma 3.5 (1) and (4). □

Corollary 3.2. Let $L : N \to R$ be any function such that $L(n) \geq \log n$ and $\lim_{n\to\infty}[\log L(n)/\log n] = 0$. Then,
(1) *strong*-2YCA$(k, L(n))$
$\subsetneq$ *weak*-2YCA$(k, L(n))$ for each Y$\in${A,U,D} and each $k \geq 4$,
(2) *strong*-1YCA$(k, L(n))$
$\subsetneq$ *weak*-1YCA$(k, L(n))$ for each Y$\in${A,U} and each $k \geq 1$, and
(3) *strong*-XNCA$(k, L(n))$
$\subsetneq$ *weak*-XNCA$(k, L(n))$ for each X$\in${1,2} and each $k \geq 3$.

It is unknown whether or not *strong*-1DCA$(k, L(n)) \subsetneq$ *weak*-1DCA$(k, L(n))$ for each $k \geq 1$, and any function $L : N \to R$ such that $L(n) \geq \log n$ and $\lim_{n\to\infty}[\log L(n)/\log n] = 0$.

4. One-Way versus Two-Way

This section investigates a relationship between one-way and two-way alternating multi-counter automata with small space. It is shown in Refs. 5), 6) that for any function $L : N \to R$ such that $\lim_{n\to\infty}[L(n)/\log n] = 0$,

weak-2UTM$(\log\log n)$
$-$*weak*-1UTM$(L(n)) \neq \phi$
weak-2ATM$(\log\log n)$
$-$*weak*-1ATM$(L(n)) \neq \phi$, and
weak-2DTM$(\log\log n)$
$-$*weak*-1NTM$(L(n)) \neq \phi$.

In correspondence to this result, we can show several results for multi-counter automata.

Lemma 4.1. Let $L_3 = \{B(1)\sharp B(2)\sharp \ldots \sharp B(n)\ 2wcw_1cw_2c \ldots cw_r | n \geq 2\ \&\ r \geq 1\ \&\ w \in \{0,1\}^{\lceil \log n\rceil}\ \&\ \forall i(1 \leq i \leq r)[w_i \in \{0,1\}^+]\ \&\ \exists j(1 \leq j \leq r)[w = w_j]\}$. Then,
(1) $L_3 \in$ *strong*-2ACA$(1, \log n)$, and
(2) $L_3 \notin$ *weak*-1ACA$(k, L(n))$ for any $k \geq 1$, and any function $L : N \to R$ such that $\lim_{n\to\infty}[\log L(n)/\log n] = 0$.

The proof of (1): One can construct a *strong*-2aca$(1, \log n)$ M accepting L_3 which acts as follows. Suppose that an input string

$$x = \not\epsilon y_1\sharp y_2\sharp \ldots \sharp y_n 2wcw_1cw_2c \ldots cw_r\$$$

(where $n \geq 2$, $r \geq 1$, and y_i's, w and w_j's are all in $\{0,1\}^+$) is presented to M. (Input strings in different form from the above can easily be rejected by M.)

In the first phase, M checks that $y_i = B(i)$ for each i $(1 \leq i \leq n)$. This can be done deterministically as in the proof of Lemma 3.5 (2).

Note that just after M successfully completes the above check, it has already stored $Z^{\lceil \log n\rceil}$ in its counter, where $\lceil \log n\rceil$ corresponds to the length of y_n $(= B(n))$.

If M successfully completes the first phase, then it checks in one universal branch (by using $Z^{\lceil \log n\rceil}$ stored in its counter) that $|w| = \lceil \log n\rceil$. In another universal branch, M existentially guesses some j $(1 \leq j \leq r)$, proceeds the input head to the first symbol of w_j, and checks that $w = w_j$. The check of "$w = w_j$" can easily be done by first checking that $|w_j| = \lceil \log n\rceil$ and then universally checking that $w_j(i) = w(i)$ for each $1 \leq i \leq |w| = |w_j| = \lceil \log n\rceil$. (For this check, $\log n$ space is sufficient.)

The proof of (2): It is shown in Ref. 6) that L_3 is not in *weak*-1ATM$(L(n))$ for any function $L : N \to R$ such that $\lim_{n\to\infty}[L(n)/\log n] = 0$. From this result and Lemma 2.1, (2) follows. □

Lemma 4.2. Let $L_4 = \{B(1)\sharp B(2)\sharp \ldots \sharp B(n)\ 2wcw' | n \geq 2\ \&\ (w, w' \in \{0,1\}^{\lceil \log n\rceil})\ \&\ w \neq w'\}$. Then,
(1) $L_4 \in$ *strong*-2DCA$(1, \log n)$, and
(2) $L_4 \notin$ *weak*-1UCA$(k, L(n))$ for any $k \geq 1$, and any function $L : N \to R$ such that $\lim_{n\to\infty}[\log L(n)/\log n] = 0$.

The proof of (1): L_4 is accepted by a *strong*-2dca$(1, \log n)$ M which acts as follows. Suppose that an input string $x = \not\epsilon y_1\sharp y_2\sharp \ldots \sharp y_n 2wcw'\$$ (where $n \geq 2$, and y_i's, w and w' are all in $\{0,1\}^+$) is presented to M. As in the proof of Lemma 3.5 (2), M can check that $y_i = B(i)$ for each i $(1 \leq i \leq n)$, and store $Z^{\lceil \log n\rceil}$ in one counter. M then checks by using $Z^{\lceil \log n\rceil}$ in its counter that $|w| = |w'| = \lceil \log n\rceil$. After that, to check that $w \neq w'$, M compares the corresponding symbols in w and w' by moving its input head back and forth.

The proof of (2): It is shown in Ref. 5) that $L_4 \notin$ *weak*-1UTM$(L(n))$ for any function $L : N \to R$ such that $\lim_{n\to\infty}[L(n)/\log n] = 0$.

From this result and Lemma 2.1, (2) follows. □

Theorem 4.1. For any $k \geq 1$, and any function $L : N \to R$ such that $\lim_{n\to\infty} [\log L(n)/\log n] = 0$,
(1) *strong*-2ACA$(1, \log n)$
 − *weak*-1ACA$(k, L(n)) \neq \phi$,
(2) *strong*-2DCA$(1, \log n)$
 − *weak*-1UCA$(k, L(n)) \neq \phi$,
(3) *strong*-2DCA$(1, \log n)$
 − *weak*-1NCA$(k, L(n)) \neq \phi$, and
(4) *strong*-2DCA$(1, \log n)$
 − *strong*-1ACA$(k, L(n)) \neq \phi$.

Proof. (1) follows from Lemma 4.1. (2) follows from Lemma 4.2. (3) follows from Lemma 3.5 (2) and (3). (4) follows from Lemma 3.5 (2) and (4). □

From this theorem, we have the following corollary.

Corollary 4.1. Let $L : N \to R$ be a function such that $L(n) \geq \log n$ and $\lim_{n\to\infty}[\log L(n)/\log n] = 0$. For each m∈{*strong*, *weak*}, each Y∈{A,U,N,D}, and each $k \geq 1$,

m-1YCA$(k, L(n)) \subsetneq$ m-2YCA$(k, L(n))$.

It is essentially shown in Ref. 2) that for any function $L : N \to R$ such that $L(n) \geq \log n$,

strong-2ATM$(L(n))$
 $=$ *strong*-1ATM$(L(n))$, and
weak-2ATM$(L(n))$ $=$ *weak*-1ATM$(L(n))$.

By using the same idea as in the proof of this result, we can prove the following theorem.

Theorem 4.2. For each $k \geq 1$, and each function $L : N \to R$ such that $L(n) \geq n$,
(1) *strong*-2ACA$(k, L(n)) \subseteq$ *strong*-1ACA$(k+1, L(n))$, and
(2) *weak*-2ACA$(k, L(n)) \subseteq$ *weak*-1ACA$(k+1, L(n))$.

Proof. Let M be any *strong*-2aca$(k, L(n))$ (resp., *weak*-2aca$(k, L(n))$) for any $k \geq 1$, and any function $L : N \to R$ such that $L(n) \geq n$. We can construct a *strong*-1aca$(k+1, L(n))$ (resp., *weak*-1aca$(k+1, L(n))$) M' which simulates M step by step as follows. In order to simulate the reading of an input symbol, M' maintains a count of the input head position of M on its extra counter. When M reads an input symbol, M' guesses the symbol using an existential state and then enters a universal state to choose one of two further actions:

(i) one action is to continue the simulation of M. During the simulation, if M enters an accepting state, then M' enters an accepting state.
(ii) the other action is to check that the symbol guessed is actually in the input position indicated by the count. This action is the only time the input head of M' moves and it can do so one-way. □

Unfortunately it is not known whether or not

strong-2ACA$(k, L(n))$
 $=$ *strong*-1ACA$(k, L(n))$,
weak-2ACA$(k, L(n))$
 $=$ *weak*-1ACA$(k, L(n))$,
strong-2ACA$(k, L(n))$
 $\subsetneq$ *strong*-1ACA$(k+1, L(n))$, and
weak-2ACA$(k, L(n))$
 $\subsetneq$ *weak*-1ACA$(k+1, L(n))$

for each $k \geq 1$, and each function $L : N \to R$ such that $L(n) \geq n$.

5. A Relationship among Determinism, Nondeterminism, and Full Alternation

This section investigates a relationship among the accepting powers of 1amca's and 2amca's with only universal states, with only existential states and with full alternation.

It is shown in Ref. 6) that for each Y∈{U,N,D}, and any function $L : N \to R$ such that $\lim_{n\to\infty}[L(n)/\log n] = 0$,

weak-1ATM$(\log\log n)$
 $-$*weak*-2YTM$(L(n)) \neq \phi$.

The following theorem corresponds to this result.

Theorem 5.1. For each Y∈{U,N,D}, each $k \geq 1$, and any function $L : N \to R$ such that $\lim_{n\to\infty}[\log L(n)/\log n] = 0$,
(1) *weak*-1ACA$(1, \log n, \text{real})$
 − *weak*-2YCA$(k, L(n)) \neq \phi$, and
(2) *strong*-2ACA$(1, \log n)$
 − *weak*-2YCA$(k, L(n)) \neq \phi$.

Proof. Let $L_5 = \{B(1)\sharp B(2)\sharp \ldots \sharp B(n)\, cw_1cw_2c \ldots cw_r2w | n \geq 2 \;\&\; r \geq 1 \;\&\; w \in \{0,1\}^{\lceil \log n \rceil} \;\&\; \forall i(1 \leq i \leq r)[w_i \in \{0,1\}^{\lceil \log n \rceil}] \;\&\; \exists j(1 \leq j \leq r)[w = w_j]\}$. To prove this theorem, we show that

(i) $L_5 \in$ *weak*-1ACA$(1, \log n, \text{real})$,
(ii) $L_5 \in$ *strong*-2ACA$(1, \log n)$, and
(iii) $L_5 \notin$ *weak*-2YCA$(k, L(n))$ for each Y∈{U,N,D}, each $k \geq 1$, and any function $L : N \to R$ such that $\lim_{n\to\infty}[\log L(n)/\log n] = 0$.

(i): A *weak*-1aca$(1, \log n)$ M which operates in realtime can accept L_5 as follows. Suppose that an input $\phi y_1\sharp y_2\sharp \ldots \sharp y_n cw_1cw_2c \ldots cw_r2w\$$

(where $n \geq 2$, $r \geq 1$, and y_i's, w and w_j's are all in $\{0,1\}^+$) is presented to M. M first makes a universal branch. In one branch, M existentially guesses some j $(1 \leq j \leq r)$, and universally checks that $w_j = w$. This check is easily done with the length of the counter bounded by $|w_j|$. In another branch, M checks by using the same idea as in the proof of Lemma 3.5 (1) whether $y_i = B(i)$ for each i $(1 \leq i \leq n)$. If M successfully completes this check, then it can store $Z^{\lceil \log n \rceil}$ in its counter. After this successful check, M universally checks by using $Z^{\lceil \log n \rceil}$ in its counter that $|w_1| = |w_2| = \ldots = |w_r| = |w| = \lceil \log n \rceil$. It will be obvious that M accepts L_5.

(ii): The proof is similar to that of (1) of Lemma 4.1 in this paper, and so it is omitted here.

(iii): It is shown in Ref. 6) that L_5 is not in *weak*-2YTM($L(n)$) for each Y$\in$\{U,N, D\}, and any function $L : N \to R$ such that $\lim_{n\to\infty}[L(n)/\log n] = 0$. It follows from this and Lemma 2.1 that L_5 is not in *weak*-2YCA($k, L(n)$) for each Y$\in$\{U,N,D\}, each $k \geq 1$, and any function $L : N \to R$ such that $\lim_{n\to\infty}[\log L(n)/\log n] = 0$. □

Corollary 5.1. For each m$\in\{$*weak*, *strong*$\}$, Y$\in$\{U,N,D\}, each $k \geq 1$, and any function $L : N \to R$ such that $L(n) \geq \log n$ and $\lim_{n\to\infty}[\log L(n)/\log n] = 0$,

(1) *weak*-1YCA($k, L(n)$)
$\subsetneq$ *weak*-1ACA($k, L(n)$), and

(2) m-2YCA($k, L(n)$) $\subsetneq$ m-2ACA($k, L(n)$).

From Lemma 3.5 (1) and (3), we have the following results.

Theorem 5.2. For each $k \geq 1$, and any function $L : N \to R$ such that $\lim_{n\to\infty}[\log L(n)/\log n] = 0$,

weak-1UCA($1, \log n$, real)
$-$*weak*-1NCA($k, L(n)$) $\neq \phi$.

Corollary 5.2. For each $k \geq 1$, and any function $L : N \to R$ such that $L(n) \geq \log n$ and $\lim_{n\to\infty}[\log L(n)/\log n] = 0$,

weak-1UCA($k, L(n)$)
$-$*weak*-1NCA($k, L(n)$) $\neq \phi$.

6. A Relationship between Multi-Counter and Multi-Head

In this section, we investigate a relationship between two-way alternating multi-counter and multi-head finite automata. For formal definitions of multi-head finite automata, the reader is referred to Refs. 19), 20). For each $k \geq 1$, a two-way alternating k-head finite automaton (2afa(k)) is defined as usual.[3] For each $k \geq 1$, a two-way alternating simple k-head finite automaton (2asfa(k)) is a 2afa(k) with the restriction that one head (called the "reading head") can sense input symbols and move in two directions while the other $(k-1)$ heads (called the "counting heads") can only detect the left endmarker "¢" and the right endmarker "$" and move in two directions. Let 2AFA(k) (resp., 2ASFA(k)) denote the class of sets accepted by 2afa(k)'s (resp., 2asfa(k)'s). The following is shown in Ref. 4).

Lemma 6.1. For each $k \geq 1$,
2AFA(k)=2ASFA (k).

It is shown in Ref. 3) that the following lemma holds.

Lemma 6.2. For each $k \geq 1$,
2AFA(k) $\subsetneq$ 2AFA($k+1$).

We now show the main result in this section.

Theorem 6.1. For each $k \geq 1$,
weak-2ACA(k, n) = 2AFA($k+1$).

Proof. From Lemma 6.1, it is sufficient to show that

weak-2ACA(k, n) $\subseteq$ 2ASFA($k+1$)
$\subseteq$ *weak*-2ACA(k, n).

We first show that *weak*-2ACA(k, n) $\subseteq$ 2ASFA($k+1$) for each $k \geq 1$. For any *weak*-2aca(k, n) M_1, we can construct a 2asfa($k+1$) M_2 simulating M_1 as follows. Let $C_1, C_2, \ldots, C_k$ be the counters of M_1 and H be the input head of M_1. Let $H_1, H_2, \ldots, H_k$ be the counting heads of M_2 and H_{k+1} be the reading head of M_2. In order for M_2 to simulate M_1, M_2 has only to use H_{k+1} to simulate H, and use the distance of H_i from the left endmarker ¢ to store the contents of C_i for each i $(1 \leq i \leq k)$. During the simulation, if one of the counting heads of M_2 reaches the right endmarker $, then M_2 is forced to enter a rejecting state. Noting that M_1 is weakly n space-bounded, it will be obvious that M_2 accepts $T(M_1)$.

We then show that 2ASFA($k+1$) $\subseteq$ *weak*-2ACA(k, n) for each $k \geq 1$. For any 2asfa($k+1$) M_3, we can construct a *weak*-2aca(k, n) M_4 simulating M_3 as follows. Let $H_1, H_2, \ldots, H_k$ be the counting heads of M_3 and H_{k+1} be the reading head of M_3. Let $C_1, C_2, \ldots, C_k$ be the counters of M_4 and H be the input head of M_4. M_4 uses H to simulate H_{k+1}, and uses C_i to simulate H_i for each i $(1 \leq i \leq k)$. Of cource, when H_i is at the left endmarker ¢, C_i is empty. Whenever M_3 moves H_i $(1 \leq i \leq k)$

to the right, M_4 enters an existential state to choose one of the following two actions.

(i) One action is to increment C_i by one and then simulate the action of M_3 when H_i is not at the right endmarker \$.

(ii) Another action is to enter a universal state to choose one of the following two actions.

(a) One action is to unchange the contents of C_i and then simulate the action of M_3 when H_i is at \$.

(b) Another action is to check that C_i now stores the length of the input (i.e., to check that H_i will reach \$ by moving one cell to right).

It will be obvious that M_4 is a weakly n space-bounded machine accepting $T(M_3)$. This completes the proof of the theorem. □

7. Hierarchy Results Based on the Number of Counters

It is shown in Ref. 8) that for each $k \geq 1$, there is a set accepted by a 1uca(k+1) operating in realtime, but not accepted by any 1aca(k) operating in realtime. The main purpose of this section is to show that for each $k \geq 1$,

$$weak\text{-1UCA}(k+1, \log n, \text{real}) - weak\text{-1ACA}(k, \log n) \neq \phi.$$

To prove the result, we first give some necessary definitions. Let M be a $weak$-1aca($k, \log n$), $k \geq 1$, and Σ be the input alphabet of M. For each storage state s of M and for each $w \in \Sigma^+$, let an *s-computation tree* of M on w is a computation tree of M whose root is labeled with the ID $(w, 1, s)$. (That is, an s-computation tree of M on w is a computation tree which represents a computation of M on $w\$$ starting with the input head on the leftmost position of w and with the storage state s.) An *s-accepting computation tree* of M on w is an s-computation tree of M on w whose leaves are all labeled with accepting ID's.

For each $n \geq 2$ and for integers $a_1, a_2, \ldots, a_k$ such that $0 \leq a_i \leq \lceil \log n \rceil$ $(1 \leq i \leq k)$, let $f_n(a_k, a_{k-1}, \ldots, a_1)$ denote the integer represented by $(\lceil \log n \rceil + 1)$-ary number $a_k a_{k-1} \ldots a_2 a_1$. That is,

$$f_n(a_k, a_{k-1}, \ldots, a_1) = a_k \times (\lceil \log n \rceil + 1)^{k-1} + a_{k-1} \times (\lceil \log n \rceil + 1)^{k-2} + \ldots + a_1 \times (\lceil \log n \rceil + 1)^0.$$

The following lemma leads to our main results.

Lemma 7.1. For each $k \geq 1$, let $L(k) = \{B(1)\sharp B(2)\sharp \ldots \sharp B(n)\sharp a^{s_1}\sharp a^{s_2}\sharp \ldots \sharp a^{s_k}\sharp h(\log n, m_1)\sharp h(\log n, m_2)\sharp \ldots \sharp h(\log n, m_r) | n \geq 2 \ \& \ r \geq 1 \ \& \ \forall i(1 \leq i \leq k)[0 \leq s_i \leq \lceil \log n \rceil] \ \& \ \forall j(1 \leq j \leq r)[m_j \geq 1 \ \& \ m_j \neq f_n(\lceil \log n \rceil - s_k, \lceil \log n \rceil - s_{k-1}, \ldots, \lceil \log n \rceil - s_1)]\}$, where $h(\log n, m) = (0\natural^{\lceil \log n \rceil})^m$. Then, for each $k \geq 1$,

(1) $L(k) \in weak\text{-1UCA}(k, \log n, \text{real})$,

and for each $k \geq 2$,

(2) $L(k) \notin weak\text{-1ACA}(k-1, \log n)$.

The proof of (1): $L(k)$ is accepted by a $weak$-1uca($k, \log n$) M (operating in realtime) which acts as follows. Let $C_1, C_2, \ldots, C_k$ be the counters of M and H be the input head of M. Suppose that an input string

$$w = y_1\sharp y_2\sharp \ldots \sharp y_n\sharp a^{s_1}\sharp a^{s_2}\sharp \ldots \sharp a^{s_k}\sharp 0\natural^{n_{11}}0\natural^{n_{12}}\ldots 0\natural^{n_{1m_1}}\sharp 0\natural^{n_{21}}0\natural^{n_{22}}\ldots 0\natural^{n_{2m_2}}\sharp \ldots \sharp 0\natural^{n_{r1}}0\natural^{n_{r2}}\ldots 0\natural^{n_{rm_r}}\$$$

(where $n \geq 2$, $r \geq 1$, $y_i \in \{0,1\}^+$, $n_{ij} \geq 1$, $m_i \geq 1$) is presented to M. (Input strings in a form different from the above can easily be rejected by M.) M universally branches to check the follwing three points:

(i) whether $y_i = B(i)$ for each i $(1 \leq i \leq n)$,

(ii) whether $n_{ij} = \lceil \log n \rceil$ for every segment $\natural^{n_{ij}}$, and

(iii) whether $0 \leq s_i \leq \lceil \log n \rceil$ for each i $(1 \leq i \leq k)$, and $m_j \neq f_n(\lceil \log n \rceil - s_k, \lceil \log n \rceil - s_{k-1}, \ldots, \lceil \log n \rceil - s_1)$ for each j $(1 \leq j \leq r)$.

(i) above can be checked as in the proof of Lemma 3.5 (1), by using one counter and universally branches. (ii) above can be checked as follows. Assuming that (i) above is successfully checked, M can store $Z^{\lceil \log n \rceil}$ in one counter. By using $Z^{\lceil \log n \rceil}$ stored in one counter, M can universally check whether $n_{ij} = \lceil \log n \rceil$ for all i, j. Assuming that (i) and (ii) above is successfully checked, (iii) above can be checked by using the following algorithm.

(a) While reading the initial segment $y_1\sharp y_2\sharp \ldots \sharp y_n$, M stores $Z^{\lceil \log n \rceil}$ in each of k counters $C_1, C_2, \ldots, C_k$. After that, for each i $(1 \leq i \leq k)$, while reading the segment a^{s_i}, M erases Z^{s_i} in C_i. (During this action, M can check whether $0 \leq s_i \leq \lceil \log n \rceil$ for each i, $1 \leq i \leq k$.) Thus, after reading the segment $y_1\sharp y_2\sharp \ldots \sharp y_n\sharp a^{s_1}\sharp a^{s_2}\sharp \ldots \sharp a^{s_k}\sharp$, $\alpha_i = Z^{\lceil \log n \rceil - s_i}$ for each $1 \leq i \leq k$, where α_i denotes the contents of C_i, and so $f_n(|\alpha_k|, |\alpha_{k-1}|, \ldots, |\alpha_1|) = f_n(\lceil \log n \rceil - s_k, \lceil \log n \rceil - s_{k-1}, \ldots, \lceil \log n \rceil - s_1)$.

(b) Assuming that (i) and (ii) above are successfully checked (i.e., $y_l = B(l)$ for all l,

and $n_{ij} = \lceil \log n \rceil$ for all i, j), after reading the segment $y_1\sharp y_2\sharp \ldots \sharp y_n\sharp a^{s_1}\sharp a^{s_2}\sharp \ldots \sharp a^{s_k}\sharp$, M universally branches to check whether $m_j \neq f_n(|\alpha_k|, |\alpha_{k-1}|, \ldots, |\alpha_1|)$ for each j $(1 \leq j \leq r)$. To check whether $m_j \neq f_n(|\alpha_k|, |\alpha_{k-1}|, \ldots, |\alpha_1|)$, M decrements $f_n(|\alpha_k|, |\alpha_{k-1}|, \ldots, |\alpha_1|)$ by one each time H meets the symbol "0" in the substring $0\natural^{n_{j1}}0\natural^{n_{j2}}\ldots 0\natural^{n_{jm_j}}$ $(\triangleq u_j)$. In order to do so, M decrements α_1 by one each time H meets the symbol 0. In this case, for example, if $|\alpha_1| = 0$ when H meets the q-th 0 from the left in u_j, then M decrements α_m (where m is the smallest integer such that $|\alpha_m| \neq 0$) by one instead of decrementing α_1 by one, and M sets $\alpha_1 = \alpha_2 = \ldots = \alpha_{m-1} = Z^{\lceil \log n \rceil}$ by using the (assumed) length $\lceil \log n \rceil$ of $\natural^{n_{jl}}$'s in u_j (note that we assume that $n_{jl} = \lceil \log n \rceil$ for each $1 \leq l \leq m_j$). M enters an accepting state only if H meets the last "0" in u_j with $|\alpha_i| \neq 0$ for some $1 \leq i \leq k$ (i.e., $f_n(|\alpha_k|, |\alpha_{k-1}|, \ldots, |\alpha_1|) \neq 0$) or H meets 0 in u_j after $|\alpha_1| = |\alpha_2| = \ldots = |\alpha_k| = 0$). It will be obvious that M accepts $L(k)$.

The proof of (2): Suppose that there exists a *weak*-1aca$(k-1, \log n)$ M which accepts $L(k)$. For each $n \geq 2$, let

$V(n) = \{B(1)\sharp \ldots \sharp B(n)\sharp a^{s_1}\sharp \ldots \sharp a^{s_k}\sharp h(\log n, m_1)\sharp \ldots \sharp h(\log n, m_{g(n)}) | \forall i(1 \leq i \leq k)[0 \leq s_i \leq \lceil \log n \rceil] \ \& \forall j(1 \leq j \leq g(n))[0 \leq m_j \leq g(n)] \ \& \ \forall l(1 \leq l \leq g(n)))[m_l \neq f_n(\lceil \log n \rceil - s_k, \lceil \log n \rceil - s_{k-1}, \ldots, \lceil \log n \rceil - s_1)]\} \subseteq L(k)$, where

$g(n) = (\lceil \log n \rceil + 1)^k - 1$, and let

$W(n) = \{h(\log n, m_1)\sharp h(\log n, m_2)\sharp \ldots \sharp h(\log n, m_{g(n)}) | \forall i(1 \leq i \leq g(n))[1 \leq m_i \leq g(n)]\}$.

Note that for each

$x = B(1)\sharp \ldots \sharp B(n)\sharp a^{s_1}\sharp \ldots \sharp a^{s_k}\sharp h(\log n, m_1)\sharp \ldots \sharp h(\log n, m_{g(n)})$

in $V(n)$,

(i) $|x| \leq cn\lceil \log n \rceil$ $(\triangleq r(n))$ for some constant c, and

(ii) there exists an accepting computation tree t of M on x such that for each node π of t, the length of each counter in $\ell(\pi)$ is bounded by $\log r(n)$.

For each storage state s of M and for each y in $W(n)$, let

$M_y(s)$

$=1$ if there exists an s-accepting computation tree t of M on y such that for each node π of t, the length of each counter in $\ell(\pi)$ is bounded by $\log r(n)$,

$=0$ otherwise.

For any two strings y, z in $W(n)$, we say that y and z are M-equivalent if $M_y(s) = M_z(s)$ for each storage state $s = (q, (\alpha_1, \ldots, \alpha_{k-1}))$ of M with $0 \leq |\alpha_i| \leq \log r(n)$ $(1 \leq i \leq k-1)$. Clearly, M-equivalence is equivalence relation on strings in $W(n)$, and there are at most

$$e(n) = 2^{p(\log r(n))^{k-1}}$$

M-equivalence classes, where p is a constant depending only on M. We denote these M-equivalence classes by $C_1, C_2, \ldots, C_{e(n)}$. For each

$y = h(\log n, m_1)\sharp h(\log n, m_2)\sharp \ldots \sharp h(\log n, m_{g(n)})$

in $W(n)$, let $b(y) = \{m | \exists i(1 \leq i \leq g(n))[m = m_i]\}$. Furthermore, for each $n \geq 2$, let $R(n) = \{b(y) | y \in W(n)\}$. Then,

$$|R(n)| = \binom{g(n)}{1} + \binom{g(n)}{2} + \ldots + \binom{g(n)}{g(n)} = 2^{g(n)} - 1.$$

We can easily see that $\log e(n) = O(\lceil \log n \rceil^{k-1})$ and $\log |R(n)| = O(\lceil \log n \rceil^k)$.☆ Thus, we have $|R(n)| > e(n)$ for large n. For such n, there must be some $\mathcal{Q}$, $\mathcal{Q}'$ $(\mathcal{Q} \neq \mathcal{Q}')$ in $R(n)$ and some C_i $(1 \leq i \leq e(n))$ such that the following statement holds:

"There are two strings y, $z \in W(n)$ such that (a) $b(y) = \mathcal{Q} \neq \mathcal{Q}' = b(z)$, and (b) y, $z \in C_i$ (i.e., y and z are M-equivalent)."

Because of (a), we can, without loss of generality, assume that there is some positive integer m such that $1 \leq m \leq g(n)$ and $m \in b(y) - b(z)$. Clearly, there are some $s_1, s_2, \ldots, s_k$ such that $m = f_n(\lceil \log n \rceil - s_k, \ldots, \lceil \log n \rceil - s_1)$, and for such s_l $(1 \leq l \leq k)$, it follows that

$y' = B(1)\sharp \ldots \sharp B(n)\sharp a^{s_1}\sharp \ldots \sharp a^{s_k}\sharp y \notin L(k)$,

and

$z' = B(1)\sharp \ldots \sharp B(n)\sharp a^{s_1}\sharp \ldots \sharp a^{s_k}\sharp z \in L(k)$.

But because of (b), y' is accepted by M iff z' is accepted by M, which is a contradiction. This completes the proof of (2). □

We are now ready to have our main result.

Theorem 7.1. For each $k \geq 1$,

weak-1UCA$(k+1, \log n, \text{real})$ $-$*weak*-1ACA$(k, \log n) \neq \phi$.

Corollary 7.1. For each $k \geq 1$ and each Y∈{A,U},

☆ For any set S, $|S|$ denotes the number of elements of S.

weak-1YCA$(k, \log n)$
$\subsetneq$*weak*-1YCA$(k+1, \log n)$.

Unfortunately, it is unknown whether or not *weak*-1YCA$(k+1, \log n, \text{real}) - $ *weak*-1ACA$(k, \log n) \neq \phi$ for each $k \geq 1$ and each Y$\in${N,D}.

For linear space-bounds, we can show two results as follows.

Corollary 7.2. For each $k \geq 1$, and any constant $c > 0$,

weak-2ACA(k, cn)
$\subsetneq$*weak*-2ACA$(k+1, cn)$.

Proof. The corollary follows from Lemma 6.2, Theorem 6.1 and Lemma 2.2. □

Corollary 7.3. For each $k \geq 1$, and any constant $c > 0$,

weak-1ACA(k, cn)
$\subsetneq$*weak*-1ACA$(k+2, cn)$.

Proof. From Corollary 7.2, Theorem 4.2 and Lemma 2.2, it follows that *weak*-1ACA(k, cn) = *weak*-1ACA$(k, n) \subseteq$ *weak*-2ACA$(k, n) \subsetneq$ *weak*-2ACA$(k+1, n) \subseteq$ *weak*-1ACA$(k+2, n)$ = *weak*-1ACA$(k+2, cn)$. □

8. Conclusion

In this paper, we presented several hierarchical results in the accepting powers of 1amca's and 2amca's with small space. We conclude this paper by listing up some interesting open problems.

(1) Let $L : N \to R$ be a function such that $L(n) \geq \log n$ and $\lim_{n\to\infty}[\log L(n)/\log n] = 0$.

- *strong*-1DCA$(k, L(n)) \subsetneq$ *weak*-1DCA$(k, L(n))$ for each $k \geq 1$?
- *strong*-xNCA$(k, L(n)) \subsetneq$ *weak*-xNCA$(k, L(n))$ for each x$\in${1,2} and each $k = 1, 2$?
- *strong*-2YCA$(k, L(n)) \subsetneq$ *weak*-2YCA$(k, L(n))$ for each Y$\in${A,U,D} and each k $(1 \leq k \leq 3)$?
- Is *strong*-2NCA$(k, L(n))$ (resp., *weak*-2NCA$(k, L(n))$) incomparable with *strong*-2UCA$(k, L(n))$ (resp., *weak*-2UCA$(k, L(n))$) for each $k \geq 1$?
- Is *weak*-1NCA$(k, L(n))$ incomparable with *weak*-1UCA$(k, L(n))$ for each $k \geq 1$?

(2) Let $L : N \to R$ be a function such that $\lim_{n\to\infty}[\log L(n)/\log n] = 0$. For each Y$\in${U,N,D} and each $k \geq 1$,

strong-2YCA$(1, \log n)$ $-$ *weak*-1ACA$(k, L(n)) \neq \phi$?

(3) Let $L : N \to R$ be a function such that $L(n) \geq n$. For each m$\in\{$*strong*, *weak*$\}$ and each $k \geq 1$,

- m-2ACA$(k, L(n))$ = m-1ACA$(k, L(n))$?
- m-2ACA$(k, L(n)) \subsetneq$ m-1ACA$(k+1, L(n))$?

(4) Let $L : N \to R$ be a function such that $\lim_{n\to\infty}[L(n)/n] = 0$.

Is *strong*-1ACA$(k, L(n))$ the class of regular sets for each $k \geq 1$?

(5) Let $L : N \to R$ be a function such that $\lim_{n\to\infty}[L(n)/\log n] = 0$.

Is *weak*-2ACA$(k, L(n))$ the class of regular sets for each $k \geq 1$?

(6) For each Y$\in${N,D} and each $k \geq 1$,

weak-1YCA$(k+1, \log n, \text{real}) - $ *weak*-1ACA$(k, \log n) \neq \phi$?

(7) For each Y$\in${A,U,N,D}, each $k \geq 1$, and any function $L : N \to R$ such that $\lim_{n\to\infty}[L(n)/n] = 0$,

strong-1YCA$(k+1, \log n) -$ *weak*-1ACA$(k, L(n)) \neq \phi$?, and

(8) For each m$\in\{$*strong*, *weak*$\}$, each $k \geq 1$, and any constant $c > 0$,

m-1ACA$(k, cn) \subsetneq$ m-1ACA$(k+1, cn)$?

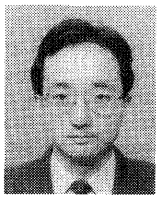
Acknowledgments The authors would like to thank the anonymous referees for greatly improving the presentation.

References

1) Chandra, A.K., Kozen, D.C. and Stockmeyer, L.J.: Alternation, *J. ACM*, Vol.28, No.1, pp.114–133 (1981).
2) Ladner, R.E., Liptom, R.J. and Stockmeyer, L.J.: Alternating Pushdown Automata, *Proc. 19th IEEE Symp. on Found. Comput. Sci.*, (Ann Arbor, MI.), pp.92–106 (1978).
3) King, K.N.: Alternating Multihead Finite Automata, *Theoret. Comput. Sci.*, Vol.61, pp.149–174 (1985).
4) Matsuno, H., Inoue, K., Taniguchi, H. and Takanami, I.: Alternating Simple Multihead Finite Automata, *Theoret. Comput. Sci.*, Vol.36, pp.291–308 (1985).
5) Inoue, K., Takanami, I. and Vollmar, R.: Alternating On-line Turing Machines with Only Universal States and Small Space Bounds, *Theoret. Comput. Sci.*, Vol.41, pp.331–339 (1985).
6) Ito, A., Inoue, K. and Takanami, I.: A Note on Alternating Turing Machines using Small Space, *Trans. IECE Japan*, Vol.E70, No.10, pp.990–996 (1987).
7) Inoue, K., Ito, A. and Takanami, I.: A Note on Real-time One-way Alternating Multicounter Machines, *Theoret. Comput. Sci.*, Vol.88, pp.287–296 (1991).
8) Yoshinaga, T., Inoue, K. and Takanami, I.: Hierarchical Properties of Realtime One-way

Alternating Multi-stack-counter Automata, *Trans. IEICE Japan*, Vol.E77-A, No.4, pp.621–629 (1994).
9) Hromkovič, J.: Alternating Multicounter Machines with Constant Number of Reversals, *Inform. Process. Lett.*, Vol.21, pp.7–9 (1985).
10) Liśkiewicz, M. and Reischuk, R.: Separating the Lower Levels of the Sublogarithmic Space Hierarchy, *STACS*, pp.17–27 (1993).
11) Chang, J.H., Ibarra, O.H., Ravikumar, B. and Berman, L.: Some Observations Concerning Alternating Turing Machines using Small Space, *Inform. Process. Lett.*, Vol.25, pp.1–9 (1987).
12) Iwama, K.: ASPACE($o(\log\log n)$) Is Regular, *SIAM J. on Computing*, Vol.22, pp.136–146 (1988).
13) Freivalds, R.: On Time Complexity of Deterministic and Nondeterministic Turing Machines, *Latvian Math.*, Vol.23, pp.158–165 (1979).
14) Alt, H. and Melhorn, K.: Lower Bounds for the Space Complexity of Context-free Recognition, *Proc. 3rd Coll. on Automata, Languages and Programing*, pp.339–354, Springer, Berlin (1976).
15) Szepietowski, A.: Some Notes on Strong and Weak $\log\log n$ Space Complexity, *Inform. Process. Lett.*, Vol.33, pp.109–112 (1989).
16) Geffert, V.: Nondeterministic Computations in Sublogarithmic Space and Space Costructibility, *SIAM J. Comput.*, Vol.20, No.3, pp.484–498 (1991).
17) Fisher, P.C., Meyer, A.R. and Rosenberg, A.L.: Counter Machines and Counter Languages, *Math. Systems Theory*, Vol.2, pp.265–283 (1968).
18) Book, R. and Ginsburg, S.: Multi-stack-counter Languages, *Math. Systems Theory*, Vol.6, pp.37–48 (1972).
19) Ibarra, O.H.: On Two-way Multihead Automata, *J. Comput. System Sci.*, Vol.7, pp.28–36 (1973).
20) Sudborough, I.H.: One-way Multihead Writing Finite Automata, *Information and Control*, Vol.30, pp.1–20 (1976).

(Received April 3, 1995)
(Accepted September 6, 1995)



Tsunehiro Yoshinaga received the B.S. degree in mathematics from Yamaguchi University, Yamaguchi, in 1982. From 1982 to 1989, he was an Assistant Professor of Tokuyama College of Technology. He is presently a Lecturer of Department of Information and Electronics Engineering, Tokuyama College of Thechnology. His current main research interests are automata theory and database theory. He is a member of IPSJ.

Katsushi Inoue received the B.E. and M.E. degrees in electrical engineering from Hiroshima University, in 1969 and 1971 respectively, and Ph.D. degree in electrical engineering from Nagoya University in 1977. From 1971 to 1973, he joined Musashino Electrical Communication Laboratory, NTT, Musashino. From 1973 to 1977, he was an Assistant Professor of Hiroshima University and from 1977 to 1987, he was an Associate Professor of Yamaguchi University. He is presently a Professor in the Faculty of Engineering, Yamaguchi University. His main interests are automata theory, computer geometry and parallel processing.