ALTERNATION FOR TWO-WAY (INKDOT) MULTI-COUNTER AUTOMATA WITH SUBLINEAR SPACE

Tsunehiro YOSHINAGA[1], Jianliang XU[2], and Katsushi INOUE[3]

[1]Department of Computer Science and Electronics Engineering, Tokuyama College of Technology
[2]NEC Software, Ltd.
[3]Department of Computer Science and Systems Engineering

This paper investigates an alternation hierarchy of alternating multi-counter automata (amca's) and some fundamental properties of two-way 1-inkdot amca's which have sublinear space. We first show that for each $k \geq 1$, an alternation hierarchy of alternating k-counter automata (aca(k)'s) with sublinear space is infinite. We then investigate a relationship between the accepting powers of amca's with and without 1 inkdot. We show, for example, that for each $k \geq 1$ and each $l \geq 1$ ($l \geq 1$ ($l \neq 3$)), sublinear space-bounded two-way aca(k)'s making at most $l-1$ alternations in any computation path on any input, with the initial state universal (existential) which have 1 inkdot are more powerful than those which have no inkdots.

Key Words : *alternating multi-counter automata, 1-inkdot, alternation hierarchy, sublinear space, computational complexity*

1. INTRODUCTION

Alternating Turing machines were introduced in [1] as a mechanism to model parallel computation, and related investigations have been continued in [2], [6]–[10], [13]–[19]. Recently, several properties of alternating Turing machines with small space bounds were provided in [2], [4]–[6], [15]–[19]. For example, von Braunmühl et al. [2] showed that there is an infinite alternation hierarchy of Turing machines with sublogarithmic space. Ranjan et al. [3] introduced a slightly modified Turing machine model, called a 1-inkdot Turing machine. The 1-inkdot Turing machine is a Turing machine with the additional power of marking 1 tape-cell in the input (with an inkdot) in any computation path. Inoue et al. [4]–[6] investigated some accepting powers of 1-inkdot alternating Turing machines and extended this model to that which has multiple inkdots.

As is well known, two-counter automata without time or space limitations have the same power as Turing machines; however, when time or space restrictions are applied, a different situation occurs. For example, hierarchical properties in the accepting powers of one-way alternating multi-counter automata operating in realtime, and alternating multi-counter automata which have small space are investigated in [7]–[9], and [10], respectively.

While many researches have been advanced and exciting results have been obtained for alternating Turing machines with sublogarithmic space, as mentioned above, properties of alternating multi-counter automata with sublinear space are little researched as far as we know. On the other hand, we think that alternating multi-counter automata can be a theoretical model of parallel computation simpler than alternating Turing machines. From the foregoing reasons, it is valuable to make an exhaustive study of the automata.

In this paper, we investigate an alternation hierarchy of multi-counter automata and some properties of 1-inkdot alternating multi-counter automata which have sublinear space.

Section 2 gives the definitions and notations necessary for this paper. Let *strong*-$2\Sigma_l\mathrm{CA}(k, s(n))$ (*weak*-$2\Sigma_l\mathrm{CA}(k, s(n))$) and *strong*-$2\Pi_l\mathrm{CA}(k, s(n))$ (*weak*-$2\Pi_l\mathrm{CA}(k, s(n))$) denote the classes of sets accepted by strongly (weakly) $s(n)$ space-bounded two-way alternating k-counter automata making at most $l-1$ alternations in any computation path on any input, with the initial states existential and universal, respectively, and let *strong*-$2\Sigma_l\mathrm{CA}^*(k, s(n))$ (*weak*-$2\Sigma_l\mathrm{CA}^*(k, s(n))$) and *strong*-$2\Pi_l\mathrm{CA}^*(k, s(n))$ (*weak*-$2\Pi_l\mathrm{CA}^*(k, s(n))$) denote the classes of sets accepted by strongly (weakly) $s(n)$ space-bounded two-way 1-inkdot alternating k-counter automata making at most $l-1$ alternations in any computation path on any input, with the initial states existential and universal, respectively. We also denote by *strong*-$2\Sigma_l\mathrm{TM}(s(n))$ (*weak*-$2\Sigma_l\mathrm{TM}(s(n))$) and *strong*-$2\Pi_l\mathrm{TM}(s(n))$ (*weak*-$2\Pi_l\mathrm{TM}(s(n))$) the classes of sets accepted by strongly (weakly) $s(n)$ space-bounded two-way alternating Turing machines making at most $l-1$ alternations in any computation path on any input, with the initial states existential and universal, respectively, and denote by *strong*-$2\Sigma_l\mathrm{TM}^*(s(n))$ and *strong*-$2\Pi_l\mathrm{TM}^*(s(n))$ the classes of sets accepted by strongly $s(n)$ space-bounded two-way 1-inkdot alternating Turing machines making at most $l-1$ alternations in any computation path on any input, with the initial states existential and universal, respectively.

Section 3 investigates an infinite alternation hierarchy of alternating multi-counter automata with sublinear space. It is shown in [2], for example, that for each $l \geq 2$, *strong*-$2\Sigma_l\mathrm{TM}(\log\log n)$ $-$ *weak*-$2\Pi_l\mathrm{TM}(o(\log n)) \neq \phi$ and *strong*-$2\Pi_l\mathrm{TM}(\log\log n)$ $-$ *weak*-$2\Sigma_l\mathrm{TM}(o(\log n)) \neq \phi$. In correspondence to this result, we show, for example, that for each $l \geq 2$ and any function $s(n)$ such that $\log s(n) = o(\log n)$, *strong*-$2\Pi_l\mathrm{CA}(1, \log n)$ $- \bigcup_{1\leq k<\infty}$*weak*-$2\Sigma_l\mathrm{CA}(k, s(n)) \neq \phi$ and *strong*-$2\Sigma_l\mathrm{CA}(1, \log n)$ $- \bigcup_{1\leq k<\infty}$*weak*-$2\Pi_l\mathrm{CA}(k, s(n)) \neq \phi$.

Section 4 investigates a relationship between the accepting powers of alternating multi-counter automata with and without 1 inkdot. It is shown in [4], [6], for example, that *strong*-$2\Sigma_1\mathrm{TM}^*(\log\log n)$ $-$ *weak*-$2\Sigma_1\mathrm{TM}(o(\log n)) \neq \phi$ and *strong*-$2\Pi_1\mathrm{TM}^*(\log\log n)$ $-$ *weak*-$2\Pi_1\mathrm{TM}(o(\log n)) \neq \phi$. In correspondence to this result, we show, for example, that for any $s(n)$ such that $\log s(n) = o(\log n)$, *strong*-$2\Sigma_1\mathrm{CA}^*(1, \log n)$ $- \bigcup_{1\leq k<\infty}$*weak*-$2\Pi_2\mathrm{CA}(k, s(n)) \neq \phi$ and *strong*-$2\Pi_1\mathrm{CA}^*(1, \log n) - \bigcup_{1\leq k<\infty}$*weak*-$2\Sigma_2\mathrm{CA}(k, s(n)) \neq \phi$.

Section 5 concludes this paper by giving some open problems.

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 [11], [12] 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 [1], [13], [14]. That is, the state set of M is divided into two disjoint sets, the set of *universal* states and the set of *existential* states. Intuitively, in a universal state M splits into some submachines which act in parallel, and in an existential state M nondeterministically chooses one of possible subsequent actions. Of course, M has a specified set of *accepting* states. We assume that M has the left endmarker "$\not\text{¢}$" and the right endmarker "\$" on the input tape, reads the input tape in two directions (that is, right or left), and can enter an accepting state only when falling off the right endmarker \$. We also assume that in one step M can increment or decrement the contents (that is, 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 ($\not\text{¢}, \$ \notin \Sigma$) is the input alphabet of M, 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 $I = (w, i, (q, (\alpha_1, \alpha_2, \ldots, \alpha_k)))$ represent the input string and the input head position, respectively. Here, 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 $\not\text{¢}$, the leftmost symbol of w, the right endmarker \$ and the immediate right to \$, respectively. The third component $(q, (\alpha_1, \alpha_2, \ldots, \alpha_k))$ ($\in S_M$) 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 $I_M(w) = (w, 0, (q_0, (\underbrace{\lambda, \ldots, \lambda}_{k})))$, 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 ν of the tree is labeled with an ID, $\ell(\nu)$,
2. if ν is an internal node (that is, a non-leaf) of the tree, $\ell(\nu)$ is universal and $\{I | \ell(\nu) \vdash_M I\} = \{I_1, I_2, \ldots, I_r\}$, then ν has exactly r children $\rho_1, \rho_2, \ldots, \rho_r$ such that $\ell(\rho_i) = I_i$, and
3. if ν is an internal node of the tree and $\ell(\nu)$ is existential, then ν has exactly one child ρ such that $\ell(\nu) \vdash_M \ell(\rho)$.

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 $T(M)$.

M makes an *alternation* if it changes an existential state into a universal state or vice versa.

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.

Let $s : N \to R$ be a function, where R denotes the set of all nonnegative real numbers. For each $\mathrm{x} \in \{1, 2\}$ and each $k \geq 1$, xaca(k) M is *weakly* (*strongly*) $s(n)$ *space-bounded* if for any $n \geq 1$ and any input w of length n accepted by M, there is an accepting computation tree τ of M on w such that for each node ν of τ, the length of each counter in $\ell(\nu)$ is bounded by $s(n)$ (if for any $n \geq 1$ and any input w of length n (accepted or not), and each node ν of any computation tree of M on w, the length of each counter in $\ell(\nu)$ is bounded by $s(n)$). A weakly (strongly) $s(n)$ space-bounded xaca(k) is denoted by *weak*-xaca($k, s(n)$) (*strong*-xaca($k, s(n)$)).

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

For each $m \in \{weak, strong\}$, each $\mathrm{x} \in \{1, 2\}$, each $k \geq 1$ and each $l \geq 1$, and any function $s : N \to R$, we denote by m-x$\sigma_l$ca($k, s(n)$) (m-x$\pi_l$ca($k, s(n)$)) a m-xaca($k, s(n)$) making at most $l - 1$ alternations in any computation path on any input, with the initial state existential (universal).

For each $m \in \{weak, strong\}$, each $k \geq 1$ and each $l \geq 1$, and any function $s : N \to R$, an m-2$\sigma_l$ca($k, s(n)$) (m-2$\pi_l$ca($k, s(n)$)) with 1 inkdot, denoted by m-2$\sigma_l$ca*($k, s(n)$) (m-2$\pi_l$ca*($k, s(n)$)), can mark 1 tape-cell on the input (with an inkdot) in any computation path. This tape-cell is marked once and for all (no erasing) and no more than one dot of ink is available. The action of the machine depends on the current state, the currently scanned input, the current contents of counters, and the presence of the inkdot on the currently scanned tape-cell.

For each $m \in \{weak, strong\}$, each $\mathrm{x} \in \{1, 2\}$, each $k \geq 1$ and each $l \geq 1$, and any function $s : N \to R$, we define

$$m\text{-xACA}(k, s(n)) = \{L | L = T(M) \text{ for some } m\text{-xaca}(k, s(n))\ M\},$$
$$m\text{-x}\Sigma_l\text{CA}(k, s(n)) = \{L | L = T(M) \text{ for some } m\text{-x}\sigma_l\text{ca}(k, s(n))\ M\},$$
$$m\text{-x}\Pi_l\text{CA}(k, s(n)) = \{L | L = T(M) \text{ for some } m\text{-x}\pi_l\text{ca}(k, s(n))\ M\},$$
$$weak\text{-}1\Sigma_l\text{CA}(k, s(n), \text{real}) = \{L | L = T(M) \text{ for some } weak\text{-}1\sigma_l\text{ca}(k, s(n))\ M \text{ operating in realtime}\},$$
$$weak\text{-}1\Pi_l\text{CA}(k, s(n), \text{real})$$

$$= \{L | L = T(M) \text{ for some } weak\text{-}1\pi_l\mathrm{ca}(k, s(n))\ M \text{ operating in realtime}\},$$

$$m\text{-}2\Sigma_l\mathrm{CA}^*(k, s(n)) = \{L | L = T(M) \text{ for some } m\text{-}2\sigma_l\mathrm{ca}^*(k, s(n))\ M\}, \text{ and}$$

$$m\text{-}2\Pi_l\mathrm{CA}^*(k, s(n)) = \{L | L = T(M) \text{ for some } m\text{-}2\pi_l\mathrm{ca}^*(k, s(n))\ M\}.$$

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 separate storage tape. (The reader is referred to [15], [16] for the formal definition of aTm's.) For any function $s : N \to R$, we denote a weakly (strongly) $s(n)$ space-bounded one-way aTm and two-way aTm by $weak$-1aTm($s(n)$) ($strong$-1aTm($s(n)$)) and $weak$-2aTm($s(n)$) ($strong$-2aTm($s(n)$)), respectively. (See [2]–[6], [15]–[19] for the definition of *weakly* (*strongly*) $s(n)$ space-bounded aTm's.) For each $m \in \{weak, strong\}$, each $\mathrm{x} \in \{1, 2\}$ and each $l \geq 1$, and any function $s : N \to R$, we denote by m-xσ_lTm ($s(n)$) (m-xπ_lTm($s(n)$)) a m-xaTm($s(n)$) making at most $l - 1$ alternations in any computation path on any input, with the initial state existential (universal), and denote by m-$2\sigma_l$Tm*($s(n)$) (m-$2\pi_l$Tm*($s(n)$)) a m-$2\sigma_l$Tm($s(n)$) (m-$2\pi_l$Tm ($s(n)$)) with 1 inkdot. (The reader is referred to [3]–[6] for formal definitions of m-$2\sigma_l$Tm*($s(n)$) and m-$2\pi_l$Tm*($s(n)$).)

For each $m \in \{weak, strong\}$, each $\mathrm{x} \in \{1, 2\}$ and each $l \geq 1$, and any function $s : N \to R$, we define

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

$$m\text{-x}\Sigma_l\mathrm{TM}(s(n)) = \{L | L = T(M) \text{ for some } m\text{-x}\sigma_l\mathrm{Tm}(s(n))\ M\},$$

$$m\text{-x}\Pi_l\mathrm{TM}(s(n)) = \{L | L = T(M) \text{ for some } m\text{-x}\pi_l\mathrm{Tm}(s(n))\ M\},$$

$$m\text{-}2\Sigma_l\mathrm{TM}^*(s(n)) = \{L | L = T(M) \text{ for some } m\text{-}2\sigma_l\mathrm{Tm}^*(s(n))\ M\}, \text{ and}$$

$$m\text{-}2\Pi_l\mathrm{TM}^*(s(n)) = \{L | L = T(M) \text{ for some } m\text{-}2\pi_l\mathrm{Tm}^*(s(n))\ M\}.$$

The following lemma can be easily proved. From now on, logarithms are base 2.

Lemma 2.1. For each $m \in \{weak, strong\}$, each $\mathrm{x} \in \{1, 2\}$, each $\mathrm{Y} \in \{\Sigma, \Pi\}$ and each $l \geq 1$, and any function $s : N \to R$,

(1) $\bigcup_{1 \leq k < \infty} m\text{-xACA}(k, s(n)) \subseteq m\text{-xATM}(\log s(n))$,

(2) $\bigcup_{1 \leq k < \infty} m\text{-x}\mathrm{Y}_l\mathrm{CA}(k, s(n)) \subseteq m\text{-x}\mathrm{Y}_l\mathrm{TM}(\log s(n))$, and

(3) $\bigcup_{1 \leq k < \infty} m\text{-}2\mathrm{Y}_l\mathrm{CA}^*(k, s(n)) \subseteq m\text{-}2\mathrm{Y}_l\mathrm{TM}^*(\log s(n))$.

It is shown in [17] that

- $strong$-1ATM($o(\log n)$) is the class of regular sets and
- $weak$-2ATM($o(\log\log n)$) is the class of regular sets.

From this result and Lemma 2.1, we can show that for any functions $s_1 : N \to R$ such that $\log s_1(n) = o(\log n)$ and $s_2 : N \to R$ such that $\log s_2(n) = o(\log\log n)$,

- $\bigcup_{1 \leq k < \infty} strong$-1ACA($k, s_1(n)$) is the class of regular sets and
- $\bigcup_{1 \leq k < \infty} weak$-2ACA($k, s_2(n)$) is the class of regular sets.

At the end of this section, we further give two notations: for each string w, w^R and $w(i)$ denote the reversal and the i-th symbol (from the left) of w, respectively.

3. AN INFINITE ALTERNATION HIERARCHY WITHOUT INKDOTS

It is shown in [2] that the alternation hierarchy for aTm's with space-bounds between $\log\log n$ and $\log n$ is infinite. This section investigates an infinite alternation hierarchy for 1amca's and 2amca's with sublinear space (without inkdots). Throughout this section, we need the languages in [2] described below.

For each $i \in N$, a special symbol $\overset{i}{\#}$ is introduced. Let

$$\begin{aligned}
D_1 &= \{0,1\}^* \text{ and} \\
D_{i+1} &= (D_i\{\overset{i}{\#}\})^* \cdot D_i \text{ for each } i \geq 1, \\
\exists D_1(u) &= \{0,1\}^* - \{u\} \text{ and} \\
\forall D_1(u) &= \{u\}, \text{ and for each } i \geq 1, \\
\exists D_{i+1}(u) &= \{W_1\overset{i}{\#}W_2\overset{i}{\#}\ldots\overset{i}{\#}W_m \in D_{i+1} | \\
&\quad \exists j(1 \leq j \leq m)[W_j \in \forall D_i(u)] \\
&\quad \& \ m \geq 1\} \text{ and} \\
\forall D_{i+1}(u) &= \{W_1\overset{i}{\#}W_2\overset{i}{\#}\ldots\overset{i}{\#}W_m \in D_{i+1} | \\
&\quad \forall j(1 \leq j \leq m)[W_j \in \exists D_i(u)] \\
&\quad \& \ m \geq 1\}.
\end{aligned}$$

C, and checks whether both the symbols satisfy (i) or (ii) above. (It determines whether they should be the same or not, by checking the first occurrence of 1 in y_{i+1}. If the symbol 1 has already occurred, then $y_{i+1}(j)$ and $y_i(j)$ should be the same; otherwise, $y_i(j)$ and $y_{i+1}(j)$ should not be the same.)

- 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, $M_2^{\exists}$ can check if ${y_{i+1}}^R$ is one more than ${y_i}^R$ ($1 \le i \le n$) using only universal branches and only one counter, and operating in one-way. It will be obvious that if $y_n\sharp y_{n-1}\sharp\dots\sharp y_1$ is such a string that $y_i = B(i)^R$ for each i ($1 \le i \le n$), then the length of y_n ($= B(n)^R$) is equal to $\lceil \log n \rceil$, and thus the length of C is bounded by $\log n$. Furthermore, it is clear that $M_2^{\exists}$ operates in realtime.

(II) The language

$$\begin{aligned} L_2^{\forall} = \{ & w_1 {}^{1}_{\#} w_2 {}^{1}_{\#} \dots {}^{1}_{\#} w_m \sharp u \sharp B(n)^R \sharp B(n-1)^R \sharp \\ & \dots \sharp B(1)^R \sharp^t \mid n \ge 2 \ \&\ m \ge 1 \ \& \\ & t \ge 1 \ \&\ u \in \{0,1\}^{\lceil \log n \rceil} \ \& \\ & \forall i (1 \le i \le m)[w_i \in D_1 \ \&\ w_i \ne u] \} \end{aligned}$$

is accepted by a *weak*-$1\pi_2\mathrm{ca}(1, \log n)$ $M_2^{\forall}$ operating in realtime as follows. Suppose that an input string

$$x = w_1 {}^{1}_{\#} w_2 {}^{1}_{\#} \dots {}^{1}_{\#} w_m \sharp u \sharp y_n \sharp y_{n-1} \sharp \dots \sharp y_1 \sharp^t$$

(where $n \ge 2$, $m \ge 1$ and $t \ge 1$, and w_i's, y_j's and u are all in $\{0,1\}^*$) is presented to $M_2^{\forall}$. (Input strings in the form different from the above can easily be rejected by $M_2^{\forall}$.)

$M_2^{\forall}$ moves on x while making a universal branch at the first symbol of each w_i ($1 \le i \le m$).

(A) In one branch, $M_2^{\forall}$ continues the action above until it reads the first $\sharp$, and then makes a universal branch to check the following two points:

(a) whether $|u| = |y_n|$, and

(b) whether $y_i = B(i)^R$ for each i ($1 \le i \le n$).

(a) and (b) can be checked in a way as described in (I).

(B) In another branch, $M_2^{\forall}$ immediately enters an existential state, guesses some j ($1 \le j \le |w_i|$), and compares $w_i(j)$ with $u(j)$ in order to check if $w_i(j) \ne u(j)$.

2. Assume that assertion (1) of this lemma holds for $L_i^{\exists}$ and $L_i^{\forall}$ ($i = 3, 4, \dots, l-1$). We shall prove assertion (1) of the lemma holds for $L_l^{\exists}$ and $L_l^{\forall}$, too.

(I) An input string x in $L_l^{\exists}$ has the form $x = WS$ with W in $\exists D_l(u)$, $W = W_1 {}^{l-1}_{\#} W_2 {}^{l-1}_{\#} \dots {}^{l-1}_{\#} W_m$ with W_i in $\forall D_{l-1}(u)$ for some i ($1 \le i \le m$), and $S = \sharp u \sharp B(n)^R \sharp \dots \sharp B(1)^R \sharp^t$, where u is in $\{0,1\}^{\lceil \log n \rceil}$, and $t \ge 1$, $m \ge 1$ and $n \ge 2$. By the assumption above, there is a realtime *weak*-$1\pi_{l-1}\mathrm{ca}(1, \log n)$ $M_{l-1}^{\forall}$ which accepts W_iS iff W_i is in $\forall D_{l-1}(u)$. $L_l^{\exists}$ is accepted by a realtime *weak*-$1\sigma_l\mathrm{ca}(1, \log n)$ $M_l^{\exists}$ acts as follows. Suppose that an input string

$$x = W_1 {}^{l-1}_{\#} W_2 {}^{l-1}_{\#} \dots {}^{l-1}_{\#} W_m \sharp u \sharp y_n \sharp y_{n-1} \sharp \dots \sharp y_1 \sharp^t$$

(where $n \ge 2$, $m \ge 1$ and $t \ge 1$, and W_i's are all in $\{0, 1, {}^{1}_{\#}, {}^{2}_{\#}, \dots, {}^{l-2}_{\#}\}^+$ and y_j's are all in $\{0,1\}^+$) is presented to $M_l^{\exists}$. (Input strings in the form different from the above can easily be rejected by $M_l^{\exists}$.) $M_l^{\exists}$ first guesses some i ($1 \le i \le m$) and runs on x to W_i. $M_l^{\exists}$ then enters a universal state, and acts just like $M_{l-1}^{\forall}$ above, but ignores any symbols between the next ${}^{l-1}_{\#}$ (just after W_i) and the first $\sharp$.

(II) An input string x in $L_l^{\forall}$ has the form $x = WS$ with W in $\forall D_l(u)$, $W = W_1 {}^{l-1}_{\#} W_2 {}^{l-1}_{\#} \dots {}^{l-1}_{\#} W_m$ with W_i in $\exists D_{l-1}(u)$ for all i ($1 \le i \le m$), and $S = \sharp u \sharp B(n)^R \sharp \dots \sharp B(1)^R \sharp^t$, where u is in $\{0,1\}^{\lceil \log n \rceil}$, and $t \ge 1$, $m \ge 1$ and $n \ge 2$. By the assumption above, there is a realtime *weak*-$1\sigma_{l-1}\mathrm{ca}(1, \log n)$ $M_{l-1}^{\exists}$ which accepts W_iS iff W_i is in $\exists D_{l-1}(u)$. There is a realtime *weak*-$1\pi_l\mathrm{ca}(1, \log n)$ $M_l^{\forall}$ which accepts $L_l^{\forall}$ as follows. Suppose that an input string x described in (I) above is presented to $M_l^{\forall}$. $M_l^{\forall}$ moves on x while making a universal branch at the first symbol of each W_i ($1 \le i \le m$).

(A) In one branch, $M_l^{\forall}$ continues the action above until it reaches the first $\sharp$. After that, $M_l^{\forall}$ runs to the right endmarker \$, and enters an accepting state.

(B) In another branch, $M_l^{\forall}$ enters an existential state, and acts just like $M_{l-1}^{\exists}$ above, but ignores all the segments between the next ${}^{l-1}_{\#}$ and the first $\sharp$.

Now, let us define the following witness languages: for each $l \geq 2$,

$$\begin{aligned} L_l^{\exists} &= \{w\sharp u\sharp B(n)^R\sharp B(n-1)^R\sharp\ldots\sharp B(1)^R\sharp^t \mid \\ &\quad w \in \exists D_l(u)\ \&\ u \in \{0,1\}^{\lceil \log n\rceil} \\ &\quad \&\ n \geq 2\ \&\ t \geq 1\}, \text{ and} \\ L_l^{\forall} &= \{w\sharp u\sharp B(n)^R\sharp B(n-1)^R\sharp\ldots\sharp B(1)^R\sharp^t \mid \\ &\quad w \in \forall D_l(u)\ \&\ u \in \{0,1\}^{\lceil \log n\rceil} \\ &\quad \&\ n \geq 2\ \&\ t \geq 1\}, \end{aligned}$$

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). The following lemma is shown in [2].

Lemma 3.1. For each $l \geq 2$,
(1) $L_l^{\exists} \in$ *weak*-$1\Sigma_l$TM$(\log\log n)$ and $L_l^{\forall} \in$ *weak*-$1\Pi_l$TM$(\log\log n)$,
(2) $L_l^{\exists} \in$ *strong*-$2\Sigma_l$TM$(\log\log n)$ and $L_l^{\forall} \in$ *strong*-$2\Pi_l$TM$(\log\log n)$, and
(3) $L_l^{\exists} \notin$ *weak*-$2\Pi_l$TM$(o(\log n))$ and $L_l^{\forall} \notin$ *weak*-$2\Sigma_l$TM$(o(\log n))$.

In correspondence to this result, we can show the following lemma.

Lemma 3.2. For each $l \geq 2$,
(1) $L_l^{\exists} \in$ *weak*-$1\Sigma_l$CA$(1,\log n,\text{real})$ and $L_l^{\forall} \in$ *weak*-$1\Pi_l$CA$(1,\log n,\text{real})$,
(2) $L_l^{\exists} \in$ *strong*-$2\Sigma_l$CA$(1,\log n)$ and $L_l^{\forall} \in$ *strong*-$2\Pi_l$CA$(1,\log n)$,
and for any function $s : N \to R$ such that $\log s(n) = o(\log n)$,
(3) $L_l^{\exists} \notin \bigcup_{1\leq k<\infty}$*weak*-$2\Pi_lCA(k,s(n))$ and $L_l^{\forall} \notin \bigcup_{1\leq k<\infty}$*weak*-$2\Sigma_lCA(k,s(n))$.

The proof of (1). We prove (1) of this lemma by using induction for l (≥ 2).
1. (I) The following *weak*-$1\sigma_2$ca$(1,\log n)$ $M_2^{\exists}$ operating in realtime accepts the language

$$\begin{aligned} L_2^{\exists} &= \{w_1{}^{\overset{1}{\#}}w_2{}^{\overset{1}{\#}}\ldots{}^{\overset{1}{\#}}w_m\sharp u\sharp B(n)^R\sharp B(n-1)^R\sharp \\ &\quad \ldots\sharp B(1)^R\sharp^t \mid n \geq 2\ \&\ m \geq 1\ \& \\ &\quad t \geq 1\ \&\ u \in \{0,1\}^{\lceil \log n\rceil}\ \& \\ &\quad \forall i(1 \leq i \leq m)[w_i \in D_1]\ \& \\ &\quad \exists j(1 \leq j \leq m)[w_i = u]\}. \end{aligned}$$

Suppose that an input string

$$x = w_1{}^{\overset{1}{\#}}w_2{}^{\overset{1}{\#}}\ldots{}^{\overset{1}{\#}}w_m\sharp u\sharp y_n\sharp y_{n-1}\sharp\ldots\sharp y_1\sharp^t$$

(where $n \geq 2$, $m \geq 1$ and $t \geq 1$, and w_i's, y_j's and u are all in $\{0,1\}^*$) is presented to $M_2^{\exists}$. (Input strings in the form different from the above can easily be rejected by $M_2^{\exists}$.)

$M_2^{\exists}$ first existentially guesses some j $(1 \leq j \leq m)$, and runs to w_j. $M_2^{\exists}$ then makes a universal branch.

(A) In one branch, in order to check whether $w_j = u$, $M_2^{\exists}$ universally checks if $w_j(i) = u(i)$ for each i $(1 \leq i \leq |w_j|)$. That is, to verify $w_j(i) = u(i)$, $M_2^{\exists}$ stores i in its counter when it picks up the symbol $w_j(i)$, compares the symbol $w_j(i)$ with the symbol $u(i)$ by using Z^i in the counter, and enters an accepting states only if $w_j(i) = u(i)$.

(B) In another branch, $M_2^{\exists}$ branches to check the following two points:

(a) whether $|u| = |y_n|$, and

(b) whether $y_i = B(i)^R$ for each i $(1 \leq i \leq n)$.

(a) above can easily be checked by using only one counter, and $M_2^{\exists}$ enters an accepting state only if (a) is successfully checked. (b) above can be checked as follows. $M_2^{\exists}$ essentially uses the algorithm in [10], [15]. By using universal branches and only one counter, $M_2^{\exists}$ can check in a way described below whether $y_i = B(i)^R$ for each i $(1 \leq i \leq n)$. $M_2^{\exists}$ compares y_{i+1} with y_i, and verifies that ${y_{i+1}}^R$ represents in binary notation (with no leading zeros) an integer which is one more than the integer represented by ${y_i}^R$ in binary notation (with no leading zeros). In doing so, $M_2^{\exists}$ will compare the j-th symbols of y_i and y_{i+1}, for all appropriate j. Observe that if ${y_{i+1}}^R$ is one more than ${y_i}^R$, then (i) $y_{i+1} = 0^m1x$ and $y_i = 1^m0x$, where x is a string (finishing with 1) over {0,1} and m is some non-negative integer, or (ii) $y_{i+1} = 0^m1$ and $y_i = 1^m$, where m is some positive integer. Let C be the counter of $M_2^{\exists}$. For each j $(1 \leq j \leq |y_{i+1}|)$, M stores the symbol $y_{i+1}(j)$ in its finite control and Z^j in C just after it has read the symbol $y_{i+1}(j)$, and makes a universal branch.

- In one branch, it compares $y_{i+1}(j)$ with the symbol $y_i(j)$ by using Z^j stored in

Clearly, the lengths of the counters of $M_l^{\exists}$ and $M_l^{\forall}$ are bounded by $\lceil \log n \rceil$, because those used in the computation are basically equal to the lengths of the counters of $M_2^{\exists}$ and $M_2^{\forall}$ when they enter accepting states. $M_2^{\exists}$ and $M_2^{\forall}$ on accepted inputs use no more than $\lceil \log n \rceil$ space, which is shown as in 1 above.

The proof of (2). It is shown in [10] that the language

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

is accepted by a strongly $\log n$ space-bounded two-way deterministic 1-counter automaton. For each $l \geq 2$, $L_l^{\exists}$ (resp., $L_l^{\forall}$) can be accepted by *strong*-$2\sigma_l\mathrm{ca}(1, \log n)$ (resp., *strong*-$2\pi_l\mathrm{ca}(1, \log n)$) M as follows. M begins by examining whether the suffix of a given input is of the form $B(n)^R\sharp B(n-1)^R\sharp \ldots \sharp B(1)^R (= (B(1)\sharp B(2)\sharp \ldots \sharp B(n))^R)$ in the way as in [10]. If this examination is successful and M stores $Z^{\lceil \log n \rceil}$ in its counter, then M can check by using the same technique as in the proof of (1) of this lemma whether the given string is a desired one.

The proof of (3). It is shown in [2] that $L_l^{\exists}$ and $L_l^{\forall}$ are not in *weak*-$2\Pi_l\mathrm{TM}(o(\log n))$ and *weak*-$2\Sigma_l\mathrm{TM}(o(\log n))$, respectively. (3) follows from this result and Lemma 2.1. □

From this lemma, we have the following theorem and corollaries.

Theorem 3.1. For each $l \geq 2$, and any function $s : N \to R$ such that $\log s(n) = o(\log n)$,
(1) *weak*-$1\Sigma_l\mathrm{CA}(1, \log n, \mathrm{real})$
$\cap$ *strong*-$2\Sigma_l\mathrm{CA}(1, \log n)$
$- \bigcup_{1 \leq k < \infty}$ *weak*-$2\Pi_l\mathrm{CA}(k, s(n)) \neq \phi$, and
(2) *weak*-$1\overline{\Pi}_l\mathrm{CA}(1, \log n, \mathrm{real})$
$\cap$ *strong*-$2\Pi_l\mathrm{CA}(1, \log n)$
$- \bigcup_{1 \leq k < \infty}$ *weak*-$2\Sigma_l\mathrm{CA}(k, s(n)) \neq \phi$.

Corollary 3.1. For each $l \geq 2$, each $k \geq 1$ and each $m, m' \in \{weak, strong\}$, and any function $s : N \to R$ such that $s(n) \geq \log n$ and $\log s(n) = o(\log n)$,
(1) m-$2\Sigma_l\mathrm{CA}(k, s(n))$ is incomparable with m'-$2\Pi_l\mathrm{CA}(k, s(n))$,
(2) *weak*-$1\Sigma_l\mathrm{CA}(k, s(n), \mathrm{real})$ is incomparable with m-$2\Pi_l\mathrm{CA}(k, s(n))$, and
(3) *weak*-$1\Pi_l\mathrm{CA}(k, s(n), \mathrm{real})$ is incomparable with m-$2\Sigma_l\mathrm{CA}(k, s(n))$.

Corollary 3.2. For each $l \geq 1$, each $k \geq 1$, each $m \in \{weak, strong\}$ and each $\mathrm{Y}, \mathrm{Y}' \in \{\Sigma, \Pi\}$, and any function $s : N \to R$ such that $s(n) \geq \log n$ and $\log s(n) = o(\log n)$,
(1) m-$2\mathrm{Y}_l\mathrm{CA}(k, s(n))$
$\subsetneqq m$-$2\mathrm{Y}'_{l+1}\mathrm{CA}(k, s(n))$,
(2) *weak*-$1\mathrm{Y}_l\mathrm{CA}(k, s(n))$
$\subsetneqq$ *weak*-$1\mathrm{Y}'_{l+1}\mathrm{CA}(k, s(n))$, and
(3) *weak*-$1\mathrm{Y}_l\mathrm{CA}(k, s(n), \mathrm{real})$
$\subsetneqq$ *weak*-$1\mathrm{Y}'_{l+1}\mathrm{CA}(k, s(n), \mathrm{real})$.

We then show a relationship between one-way and two-way operations, and strongly and weakly space-bounds.

Theorem 3.2. For each $\mathrm{Y} \in \{\Sigma, \Pi\}$, and any function $s : N \to R$ such that $\log s(n) = o(\log n)$,
strong-$2\mathrm{Y}_2\mathrm{CA}(1, \log n)$
$- \bigcup_{1 \leq k < \infty}$ *weak*-$1\mathrm{ACA}(k, s(n)) \neq \phi$.

Proof. Let

$$\begin{aligned} L_1 = \{&B(1)\sharp B(2)\sharp \ldots \sharp B(n) 2 w c w_1 c w_2 c \ldots c w_r | \\ &n \geq 2 \ \& \ r \geq 1 \ \& \ w \in \{0,1\}^{\lceil \log n \rceil} \\ &\& \ \forall i (1 \leq i \leq r)[w_i \in \{0,1\}^+] \\ &\& \ \exists j (1 \leq j \leq r)[w = w_j]\}, \text{ and} \\ L_2 = \{&B(1)\sharp B(2)\sharp \ldots \sharp B(n) 2 w c w_1 c w_2 c \ldots c w_r | \\ &n \geq 2 \ \& \ r \geq 1 \ \& \ w \in \{0,1\}^{\lceil \log n \rceil} \\ &\& \ \forall i (1 \leq i \leq r)[w_i \in \{0,1\}^{\lceil \log n \rceil} \\ &\& \ w \neq w_i]\}. \end{aligned}$$

Then, (i) '$L_1 \in$ *strong*-$2\Sigma_2\mathrm{CA}(1, \log n)$' and (ii) '$L_1 \notin \bigcup_{1 \leq k < \infty}$ *weak*-$1\mathrm{ACA}(k, s(n))$' are essentially proved in [10]. On the other hand, (iii) '$L_2 \in$ *strong*-$2\Pi_2\mathrm{CA}(1, \log n)$' and (iv) '$L_2 \notin$ *weak*-$1\mathrm{ATM}(o(\log n))$' can be proved in the same way as in the proofs of Lemma 4.1 in [10] and Lemma 1 in [16], respectively. From (iv) and Lemma 2.1, '$L_2 \notin \bigcup_{1 \leq k < \infty}$ *weak*-$1\mathrm{ACA}(k, s(n))$' follows. So, the full proofs are omitted here. □

Corollary 3.3. For each $l \geq 2$ and each $k \geq 1$, and any function $s : N \to R$ such that $s(n) \geq \log n$ and $\log s(n) = o(\log n)$,
(1) *weak*-$1\Sigma_l\mathrm{CA}(k, s(n))$
$\subsetneqq$ *weak*-$2\Sigma_l\mathrm{CA}(k, s(n))$, and
(2) *weak*-$1\Pi_l\mathrm{CA}(k, s(n))$
$\subsetneqq$ *weak*-$2\Pi_l\mathrm{CA}(k, s(n))$.

Let *weak*-$2\mathrm{DCA}(k, s(n))$ (*strong*-$2\mathrm{DCA}(k, s(n))$) denote the class of sets accepted by weakly (strongly) $s(n)$ space-bounded two-way determin-

istic k-counter automata. It is shown in [10] that for any function $s : N \to R$ such that $\log s(n) = o(\log n)$,

$$\begin{aligned}&\textit{weak}\text{-2DCA}(4, \log n)\\ &\quad - \bigcup_{1\le k<\infty} \textit{strong}\text{-2ACA}(k, s(n)) \neq \phi, \text{ and}\\ &\textit{weak}\text{-1}\Sigma_1\text{CA}(3, \log n)\\ &\quad - \bigcup_{1\le k<\infty} \textit{strong}\text{-2ACA}(k, s(n)) \neq \phi.\end{aligned}$$

From this result, the following corollary is shown.

Corollary 3.4. Let $s : N \to R$ be a function such that $s(n) \geq \log n$ and $\log s(n) = o(\log n)$. Then,
(1) $\textit{strong}\text{-2Y}_l\text{CA}(k, s(n)) \subsetneq \textit{weak}\text{-2Y}_l\text{CA}(k, s(n))$
for each $\text{Y} \in \{\Sigma, \Pi\}$, each $l \geq 2$ and each $k \geq 3$,
(2) $\textit{strong}\text{-2}\Sigma_1\text{CA}(k, s(n)) \subsetneq \textit{weak}\text{-2}\Sigma_1\text{CA}(k, s(n))$
for each $k \geq 3$, and
(3) $\textit{strong}\text{-2}\Pi_1\text{CA}(k, s(n)) \subsetneq \textit{weak}\text{-2}\Pi_1\text{CA}(k, s(n))$
for each $k \geq 4$.

4. THE POWER OF ONE INKDOT

This section investigates a relationship between the accepting powers of space-bounded 2amca's with and without 1 inkdot. This investigation is based on the results of 2aTm's in [4], [6]. Inoue et al. [4], [6] showed that

$$\begin{aligned}&\textit{strong}\text{-2}\Sigma_1\text{TM}^*(\log\log n)\\ &\quad - \textit{weak}\text{-2}\Sigma_1\text{TM}(o(\log n)) \neq \phi,\\ &\textit{strong}\text{-2}\Pi_1\text{TM}^*(\log\log n)\\ &\quad - \textit{weak}\text{-2}\Pi_1\text{TM}(o(\log n)) \neq \phi, \text{ and}\\ &\textit{strong}\text{-2}\Pi_3\text{TM}^*(\log\log n)\\ &\quad - \textit{weak}\text{-2ATM}(o(\log n)) \neq \phi.\end{aligned}$$

In correspondence to this result, we can show several results for 2amca's. In order to do so, we first give the following two lemmas.

Lemma 4.1. Let $L_2^{\exists}$ and $L_2^{\forall}$ be the languages described in the proof of (1) of Lemma 3.2. Then,
(1) $L_2^{\exists} \in \textit{strong}\text{-2}\Sigma_1\text{CA}^*(1, \log n)$, and
(2) $L_2^{\forall} \in \textit{strong}\text{-2}\Pi_1\text{CA}^*(1, \log n)$,
and for any function $s : N \to R$ such that $\log s(n) = o(\log n)$,
(3) $L_2^{\forall} \notin \bigcup_{1\le k<\infty} \textit{weak}\text{-2}\Sigma_2\text{CA}(k, s(n))$, and
(4) $L_2^{\exists} \notin \bigcup_{1\le k<\infty} \textit{weak}\text{-2}\Pi_2\text{CA}(k, s(n))$.

The proof of (1) (resp., (2)). We can construct a $\textit{strong}\text{-2}\sigma_1\text{ca}^*(1, \log n)$ (resp., $\textit{strong}\text{-2}\pi_1\text{ca}^*(1, \log n)$) M which acts as follows. Suppose that an input string

$$\not\!\!\$ w_1 \overset{1}{\#} w_2 \overset{1}{\#} \dots \overset{1}{\#} w_m \sharp u \sharp y_n \sharp y_{n-1} \sharp \dots \sharp y_1 \sharp^t \$$$

(where $n \geq 2$, $m \geq 1$ and $t \geq 1$, and w_i's, y_j's and u are all in $\{0,1\}^*$) is presented to M. (Input strings in the form different from the above can easily be rejected by M.) For each i $(1 \leq i \leq n)$, M can first check whether $y_i = B(i)^R$ and store $Z^{\lceil \log n \rceil}$ in its counter when $y_i = B(i)^R$, as in the same way in the proof of (2) of Lemma 3.2. (Of course, M never enters an accepting state if $y_i \neq B(i)^R$ for some $1 \leq i \leq n$.) If M successfully completes the action above, then it checks by using $Z^{\lceil \log n \rceil}$ stored in the counter if $|u| = \lceil \log n \rceil$. After that, M existentially guesses some j $(1 \leq j \leq m)$, and marks the symbol $\overset{1}{\#}$ just before w_j by the inkdot in order to check whether $u = w_j$ (resp., M universally branches and marks the symbol $\overset{1}{\#}$ just before w_j by the inkdot in order to check whether $u \neq w_j$ for each j $(1 \leq j \leq m)$). Finally, M checks by using $Z^{\lceil \log n \rceil}$ in its counter whether $|w_j| = \lceil \log n \rceil$, and then deterministically checks by using the inkdot as a pilot whether $u = w_j$ (resp., M deterministically checks by using the inkdot as a pilot whether $u \neq w_j$). That is, for example, M stores Z^i $(1 \leq i \leq |u| = \lceil \log n \rceil)$ in its counter when M picks up the symbol $w_j(i)$ and by using Z^i in its counter compares $w_j(i)$ with $u(i)$ while moving its input head back and forth. (For the check, it is clear that $\log n$ space is sufficient.) M enters an accepting state only if these checks above are all successful. It will be obvious that M accepts the language $L_2^{\exists}$ (resp., $L_2^{\forall}$).
The proofs of (3) and (4). These proofs have been already shown in Lemma 3.2 (3). □

Lemma 4.2. Let

$$\begin{aligned}L_3 \;=\; &\{B(1)\sharp B(2)\sharp \dots \sharp B(n) c w_1 c w_2 c \dots\\ &c w_r c c u_1 c u_2 c \dots c u_{r'} \mid n \geq 2 \;\&\; (r, r' \geq 1)\\ &\&\; \forall i(1 \leq i \leq r)\forall j(1 \leq j \leq r')\\ &[w_i, u_j \in \{0,1\}^{\lceil \log n \rceil}] \;\&\; \forall i(1 \leq i \leq r)\\ &[\exists j(1 \leq j \leq r')[w_i = u_j]]\} \text{ and}\\ L_4 \;=\; &\{B(1)\sharp B(2)\sharp \dots \sharp B(n) c w_1 c w_2 c \dots\\ &c w_r c c u_1 c u_2 c \dots c u_{r'} \mid n \geq 2 \;\&\; (r, r' \geq 1)\\ &\&\; \forall i(1 \leq i \leq r)\forall j(1 \leq j \leq r')\end{aligned}$$

$[w_i, u_j \in \{0,1\}^{\lceil \log n \rceil}]$ & $\exists i(1 \leq i \leq r)$
$[\forall j(1 \leq j \leq r')[w_i \neq u_j]]\}$.

Then,
(1) $L_3 \in$ *strong*-$2\Pi_3\mathrm{CA}^*(1, \log n)$, and
(2) $L_4 \in$ *strong*-$2\Sigma_3\mathrm{CA}^*(1, \log n)$,
and for any function $s : N \to R$ such that $\log s(n) = o(\log n)$,
(3) $L_3 \notin \bigcup_{1 \leq k < \infty}$ *weak*-$2\mathrm{ACA}(k, s(n))$,
(4) $L_3 \notin \bigcup_{1 \leq k < \infty}$ *weak*-$2\Sigma_1\mathrm{CA}^*(k, s(n))$, and
(5) $L_4 \notin \bigcup_{1 \leq k < \infty}$ *weak*-$2\Pi_1\mathrm{CA}^*(k, s(n))$.

The proof of (1) (resp., (2)). One can construct a *strong*-$2\pi_3\mathrm{ca}^*(1, \log n)$ (resp., *weak*-$2\sigma_3\mathrm{ca}^*(1, \log n)$) M which accepts L_3 (resp., L_4) as follows. Suppose that an input string

$$\not{\epsilon} y_1 \sharp y_2 \sharp \ldots \sharp y_n c w_1 c w_2 c \ldots c w_r c c u_1 c u_2 c \ldots c u_{r'} \$$$

(where $n \geq 2$ and $(r,\ r' \geq 1)$, and y_i's and w_j's are all in $\{0,1\}^+$) is presented to M. (Input strings in the form different from the above can easily be rejected by M.) M first stores $Z^{\lceil \log n \rceil}$ in its counter when $y_i = B(i)$ for each $1 \leq i \leq n$ in the way as in [10]. M then checks by using $Z^{\lceil \log n \rceil}$ stored in the counter if $|w_1| = \ldots = |w_r| = |u_1| = \ldots = |u_{r'}| = \lceil \log n \rceil$. After that, M universally checks whether for all i $(1 \leq i \leq r)$, $w_i = u_j$ for some j $(1 \leq j \leq r')$ (resp., M existentially checks whether for some i $(1 \leq i \leq r)$, $w_i \neq u_j$ for all j $(1 \leq j \leq r')$). That is, for example, in order to check if $w_i = u_j$ for some j $(1 \leq j \leq r')$, M first branches, marks the symbol c just before w_i by the inkdot for each i $(1 \leq i \leq r)$, and then moves to the right to existentially choose u_j (resp., in order to check if $w_i \neq u_j$ for all j $(1 \leq j \leq r')$, M first guesses some i $(1 \leq i \leq r)$, marks the symbol c just before w_i by the inkdot, and then moves to the right to universally choose u_j). After that, by universally checking if $w_i(t) = u_j(t)$ for all t $(1 \leq t \leq \lceil \log n \rceil)$ (resp., by existentially checking if $w_i(t) \neq u_j(t)$ for some t $(1 \leq t \leq \lceil \log n \rceil)$), M can check if $w_i = u_j$ (resp., $w_i \neq u_j$). For this check, it is sufficient to use only one counter and use only its contents of length $\log n$. It will be obvious that L_3 (resp., L_4) $= T(M)$.

The proofs of (3), (4) and (5). It is shown in [6] that '$L_3 \notin$ *weak*-$2\mathrm{ATM}(o(\log n))$', '$L_3 \notin$ *weak*-$2\Sigma_1\mathrm{TM}^*(o(\log n))$', and '$L_4 \notin$ *weak*-$2\Pi_1\mathrm{TM}^*(o(\log n))$'. From this result and Lemma 2.1, (3), (4) and (5) follow. □

From these two lemmas, we give the following theorem.

Theorem 4.1. For any function $s : N \to R$ such that $\log s(n) = o(\log n)$,
(1) *strong*-$2\Sigma_1\mathrm{CA}^*(1, \log n)$
$\quad - \bigcup_{1 \leq k < \infty}$ *weak*-$2\Pi_2\mathrm{CA}(k, s(n)) \neq \phi$,
(2) *strong*-$2\Pi_1\mathrm{CA}^*(1, \log n)$ −
$\quad - \bigcup_{1 \leq k < \infty}$ *weak*-$2\Sigma_2\mathrm{CA}(k, s(n)) \neq \phi$,
(3) *strong*-$2\Pi_3\mathrm{CA}^*(1, \log n)$
$\quad - \bigcup_{1 \leq k < \infty}$ *weak*-$2\mathrm{ACA}(k, s(n)) \neq \phi$, and
(4) *strong*-$2\Sigma_4\mathrm{CA}^*(1, \log n)$
$\quad - \bigcup_{1 \leq k < \infty}$ *weak*-$2\mathrm{ACA}(k, s(n)) \neq \phi$.

We conjecture that for any function $s : N \to R$ such that $\log s(n) = o(\log n)$,

strong-$2\Sigma_3\mathrm{CA}^*(1, \log n)$
$\quad - \bigcup_{1 \leq k < \infty}$ *weak*-$2\mathrm{ACA}(k, s(n)) \neq \phi$.

Unfortunately, we have no sufficient proof for this conjecture.

Corollary 4.1. For each $m \in \{strong, weak\}$ and each $k \geq 1$, and any function $s : N \to R$ such that $s(n) \geq \log n$ and $\log s(n) = o(\log n)$,
(1) m-$2\Pi_l\mathrm{CA}(k, s(n)) \subsetneq m$-$2\Pi_l\mathrm{CA}^*(k, s(n))$
for each $l \geq 1$, and
(2) m-$2\Sigma_l\mathrm{CA}(k, s(n)) \subsetneq m$-$2\Sigma_l\mathrm{CA}^*(k, s(n))$
for each $l \geq 1$ $(l \neq 3)$.

5. CONCLUSION

We conclude this paper by enumerating several open problems related to this paper.

Let $s : N \to R$ be a function such that $\log s(n) = o(\log n)$.

(1) *strong*-$2\Sigma_3\mathrm{CA}^*(1, \log n)$
$\quad - \bigcup_{1 \leq k < \infty}$ *weak*-$2\mathrm{ACA}(k, s(n)) \neq \phi$?

(2) For each $\mathrm{Y} \in \{\Sigma, \Pi\}$ and each $l \in \{1, 2\}$,
strong-$2\mathrm{Y}_l\mathrm{CA}^*(1, \log n)$
$\quad - \bigcup_{1 \leq k < \infty}$ *weak*-$2\mathrm{ACA}(k, s(n)) \neq \phi$?

Let $s : N \to R$ be a function such that $s(n) \geq \log n$ and $\log s(n) = o(\log n)$.

(3) For each $k_1 \in \{1, 2\}$, each $k_2 \in \{1, 2, 3\}$, each $\mathrm{Y} \in \{\Sigma, \Pi\}$, and each $l \geq 2$,

- *strong*-$2\Sigma_1\mathrm{CA}(k_1, s(n))$
$\quad \subsetneq$ *weak*-$2\Sigma_1\mathrm{CA}(k_1, s(n))$?,

- *strong*-$2\Pi_1\mathrm{CA}(k_2, s(n))$ $\subsetneq$ *weak*-$2\Pi_1\mathrm{CA}(k_2, s(n))$?, and
- *strong*-$2\mathrm{Y}_l\mathrm{CA}(k_1, s(n))$ $\subsetneq$ *weak*-$2\mathrm{Y}_l\mathrm{CA}(k_1, s(n))$?

Let *weak*-2DCA*$(k, s(n))$ (*strong*-2DCA*$(k, s(n))$) denote the class of sets accepted by weakly (strongly) $s(n)$ space-bounded two-way 1-inkdot deterministic k-counter automata, and let *weak*-2DTM$(s(n))$ (*strong*-2DTM$(s(n))$) and *weak*-2DTM*$(s(n))$ (*strong*-2DTM*$(s(n))$) denote the classes of sets accepted by weakly (strongly) $s(n)$ space-bounded two-way deterministic Turing machines and Turing machines with 1 inkdot, respectively. It is shown in [3], [5] that

$$m\text{-2DTM}^*(s(n)) = m\text{-2DTM}(s(n))$$

for each $m \in \{weak, strong\}$, and any $s : N \to R$ such that $s(n) \geq \log\log n$ and $s(n) = o(\log n)$.

(4) For each $m \in \{weak, strong\}$ and each $k \geq 1$, and any function $s : N \to R$ such that $s(n) \geq \log n$ and $\log s(n) = o(\log n)$,

- m-2DCA*$(k, s(n))$ $= m$-2DCA$(k, s(n))$?,
- $\bigcup_{1\leq k<\infty} m$-2DCA*$(k, s(n))$ $= \bigcup_{1\leq k<\infty} m$-2DCA$(k, s(n))$?, and
- what is the minimum of k's $(k' \geq k)$ such that m-2DCA*$(k, s(n))$ $\subseteq m$-2DCA$(k', s(n))$?

REFERENCES

1) A.K. Chandra, D.C. Kozen, and L.J. Stockmeyer, "Alternation," J. ACM, vol.28, no.1, pp.114–133, 1981.

2) B. von Braunmühl, R. Gengler, and R. Rettinger, "The alternation hierarchy for sublogarithmic space is infinite," Comput. Complexity, vol.3, pp.207–230, 1993.

3) D. Ranjan, R. Chang, and J. Hartmanis, "Space bounded computations: review and new separation results," Theoret. Comput. Sci., vol.80, pp.289–302, 1991.

4) K. Inoue, A. Ito, and I. Takanami, "A relationship between nondeterministic Turing machines and 1-inkdot Turing machines with small space, Infor. Process. Lett., vol.43, pp.225–227, 1992.

5) K. Inoue, A. Ito, I. Takanami, and T. Yoshinaga, "A note on multi-inkdot nondeterministic Turing machines with small space," Infor. Process. Lett., vol.48, pp.285–288, 1993.

6) K. Inoue, A. Ito, and I. Takanami, "On 1-inkdot alternating Turing machines with small space," Theoret. Comput. Sci., vol.127, pp.171–179, 1994.

7) K. Inoue, A. Ito, and I. Takanami, "A note on realtime one-way alternating multicounter machines," Theoret. Comput. Sci., vol.88, pp.287–296, 1991.

8) T. Yoshinaga, K. Inoue, and I. Takanami, "Hierarchical properties of realtime one-way alternating multistack-counter automata," IEICE Trans. Fundamentals, vol.E77-A, no.4, pp.621–629, Apr. 1994.

9) T. Yoshinaga and K. Inoue, "Alternating finite automata with counters and stack-counters operating in realtime," IEICE Trans. Inf. & Syst., vol.E78-D, no.8, pp.929–938, Aug. 1995.

10) T. Yoshinaga, and K. Inoue, "A Note on Alternating Multi-counter Automata with Small Space," Trans. IPS Japan, vol.36, no.12, pp.2741–2753, Dec. 1995.

11) P.C. Fisher, A.R. Meyer, and A.L. Rosenberg, "Counter machines and counter languages," Math. Systems Theory, vol.2, pp.265–283, 1968.

12) R. Book, and S. Ginsburg, "Multi-stack-counter languages," Math. Systems Theory, vol.6, pp.37–48, 1972.

13) R.E. Ladner, R.J. Lipton, and L.J. Stockmeyer, "Alternating Pushdown Automata," Proc. IEEE 19th Symp. on Foundations of Computer Science, Ann Arbor, MI., pp. 92–106, 1978.

14) K.N. King, "Alternating multihead finite automata," Theoret. Comput. Sci., vol.61, pp.149–174, 1985.

15) K. Inoue, I. Takanami, and R. Vollmar, "Alternating on-line Turing machines with only universal states and small space bounds," Theoret. Comput. Sci., vol.41, pp.331–339, 1985.

16) A. Ito, K. Inoue, and I. Takanami, "A note on alternating Turing machines using small space," IECE Trans., vol.E70, no.10, pp.990–996, 1987.

17) K. Iwama, ASPACE($o(\log\log n)$) is regular, SIAM J. Computing, vol.22, pp.136–146, 1993.

18) J.H. Chang, O.H. Ibarra, B. Ravikumar, and L. Berman, "Some observations concerning alternating Turing machines using small space," Infor. Process. Lett., vol.25, pp.1–9, 1987.

19) M. Liśkiewicz, and R. Reischuk, "Separating the lower levels of the sublogarithmic space hierarchy," Proc. STACS'93, Lecture Notes in Computer Science 665, Springer-Verlag, pp.17–27, 1993.

(Received May 15, 1998)

線型以下の空間量をもつ２方向（インクドット）マルチカウンタオートマタの交代性

義永　常宏，徐　建良，井上　克司

本論文では，線型以下の空間量をもつ交代性マルチカウンタオートマタに関する交代の階層性，及び線型以下の空間量をもつ２方向１インクドット交代性マルチカウンタオートマタに関する基本的ないくつかの性質について考察している．初めに，線型以下の空間量に制限された交代性マルチカウンタオートマタは，無限の交代階層性を有することを示す．次に，線型以下の空間量の交代性マルチカウンタオートマタにおいて，１インクドットをもつ場合とそうでない場合との受理能力の関係について考察する．例えば，各 $l \geq 1$ ($l \geq 1$ ($l \neq 3$)) に対し，任意の入力上の任意の計算路において，全称状態（存在状態）から始まり，高々 $l-1$ 回の交代を行う２方向交代性マルチカウンタオートマタに関しては，１インクドットをもつ場合の方が，もたない場合よりも真に受理能力が高いことなどを証明する．