1-Way versus 2-Way Alternating Multi-Counter Automata with Sublinear Space

Tsunehiro YOSHINAGA[*1] and Makoto SAKAMOTO[*2]

Abstract

This paper investigates the difference in the accepting powers between 1-way and 2-way operations of sublinear space-bounded alternating multi-counter automata. For each $l \geq 1$ and any function $L(n)$, let *weak*-1ACA(l, $L(n)$) (resp., *strong*-2ACA(l, $L(n)$)) denote the class of sets accepted by weakly 1-way (resp., strongly 2-way) $L(n)$ space-bounded alternating l-counter automata. We show that for any function $L(n)$ such $\log L(n) = o(\log n)$, *strong*-2ACA(1, $\log n$) $- \cup_{1 \leq l < \infty}$ *weak*-1ACA(l, $L(n)$) $\neq \phi$. So, we have m-1ACA(l, $L(n)$) $\subsetneq$ m-2ACA(l, $L(n)$) for each $l \geq 1$, each $m \in$ {*strong*, *weak*} and any function $L(n) \geq \log n$ such that $\log L(n) = o(\log n)$.

Key Words: alternating multi-counter automata, multi-inkdot, universal states, sublinear space, computational complexity

1. Introduction and preliminaries

A *multi-counter automaton* (mca) 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 is a fixed symbol). It is shown in Ref. 1) that 2-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 (See, for example, Refs. 2), 3)).

From a theoretical point of view, in this paper, we are interested in knowing fundamental properties of alternating mca's (amca's), and especially investigate the essential difference between 1-way and 2-way operations in the accepting powers of amca's which have sublinear space, in correspondence to the result in Ref. 4).

A 2-*way alternating multi-counter automaton M* is a generalization of a two-way nondeterministic multi-counter automaton in the same sense as Ref. 5).

The state set of M is partitioned into *universal* and *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. M has the left endmarker "¢" and the right endmarker "\$" on the input tape, reads the input tape right or left, and can enter an accepting state only when falling off \$. In one step M can also increment or decrement the contents (i.e., the length) of each counter by at most one.

For each $l \geq 1$, we denote a two-way alternating l-counter automaton by 2aca(l). An *instantaneous description* (ID) of 2aca(l) M is an element of

$$\Sigma^* \times \mathbb{N} \times S_M,$$

where Σ (\$, ¢ $\notin \Sigma$) is the input alphabet of M, $\mathbb{N}$ denotes the set of all non-negative integers, and

$$S_M = Q \times (\{Z\}^*)^l$$

where Q is the set of states. The first and second components, w and i, of an ID

$$I = (w, i, (q, (\alpha_1, \alpha_2, \ldots, \alpha_l)))$$

represent the input string and the input head position, respectively. The third component $(q, (\alpha_1, \alpha_2, \ldots, \alpha_l))$ of I represents the state of the finite control and the contents of the l counters. I is said to be a *universal* (*existential, accepting*) ID if q is a universal (an existential, an accepting) state. An element of S_M, $(q, (\alpha_1, \alpha_2, \ldots, \alpha_l))$, is called a *storage state* of M.

*1 Department of Computer Science and Electrics Engineering

*2 University of Miyazaki

The *initial* ID of M on $w \in \Sigma^*$ is

$$I_M(w) = (w, 0, (q_0, (\lambda_1, \lambda_2, \ldots, \lambda_l))),$$

where q_0 is the initial state of M and λ_i ($1 \le i \le l$) denotes the empty string.

We write $I \vdash_M I'$ and say that 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

$$I_0 \vdash_M I_1 \vdash_M \ldots \vdash_M I_n \quad (n \ge 0),$$

where $I_0 = I_M(w)$.

A *computation tree of M on input w* is a finite, nonempty tree such that the root is labeled by the initial ID I_0, and the children of any non-leaf node π labeled by a universal (an existential) ID, $\ell(\pi)$, include all (one) of the immediate successors of $\ell(\pi)$.

A computation tree of M on input w is *accepting* if all the leaves are labeled by accepting ID's. We say that M *accepts* w if there is an accepting computation tree of M on w.

For each storage state $(q, (\alpha_1, \alpha_2, \ldots, \alpha_l))$ and for each $w \in \Sigma^*$, let a *$(q, (\alpha_1, \alpha_2, \ldots, \alpha_l))$-computation tree* of M on w be a computation tree of M whose root is labeled with the ID $(q, (\alpha_1, \alpha_2, \ldots, \alpha_l))$. (That is, a $(q, (\alpha_1, \alpha_2, \ldots, \alpha_l))$-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 an d with the storage state $(q, (\alpha_1, \alpha_2, \ldots, \alpha_l))$).

A *$(q, (\alpha_1, \alpha_2, \ldots, \alpha_l))$-accepting computation tree* of M on w is a $(q, (\alpha_1, \alpha_2, \ldots, \alpha_l))$-computation tree of M on w whose leaves are all labeled with accepting ID's.

For any function $L(n)$, M is *weakly* (*strongly*) $L(n)$ *space-bounded* if for any $n \ge 1$ 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 π of τ (if for any $n \ge 1$ 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 of the ID $\ell(\pi)$ is bounded by $L(n)$. That is, for each α_i in the ID $\ell(\pi) = (w, i, (q, (\alpha_1, \alpha_2, \ldots, \alpha_l)))$, $|\alpha_i| \le L(n)$ ($1 \le i \le l$).

A 1-way alternating l-counter automata (1aca(l)) is a 2aca(l) whose input head cannot move to the left.

For each $l \ge 1$ and any function $L(n)$, let denote by *weak*-2ACA(l,$L(n)$) and *strong*-2ACA(l,$L(n)$) the classes of sets accepted by weakly and strongly $L(n)$ space-bounded 2aca(l), respectively and by *weak*-1ACA(l,$L(n)$) and *strong*-1ACA(l,$L(n)$) the classes of sets accepted by weakly and strongly $L(n)$ space- bounded 1aca(l), respectively.

For any function $L(n)$, we denote by *weak*-2ATM($L(n)$) (resp., *weak*-1ATM ($L(n)$)) the class of sets accepted by weakly 2-way (resp., 1-way) $L(n)$ space-bounded alternating Turing machines (aTm's). (If necessary, see Ref. 4) for weakly and strongly $L(n)$ space-bounded 1-way and 2-way aTm's).

Section 2 investigates the difference in the accepting power between 1-way and 2-way amca's with sublinear space, and shows that *strong*-2ACA(1, log n) $- \cup_{1 \le l < \infty}$ *weak*-1ACA(l, $L(n)$) $\neq \phi$ for any function $L(n)$ such that $\log L(n) = o(\log n)$. Section 3 concludes this paper by giving a few open problems.

2. Result

It is shown in Ref. 4) that

$$\textit{weak}\text{-2ATM}(\log\log n) - \textit{weak}\text{-1ATM}(o(\log n)) \neq \phi .$$

Therefore, for any function $L(n)$ such that $\log\log n \le L(n) = o(\log n)$, it follows that

$$\textit{weak}\text{-1ATM}(L(n)) \subsetneq \textit{weak}\text{-2ATM}(L(n)).$$

To obtain our corresponding result, we first need the following lemma. From now on, logarithms are base 2.

Lemma 2.1: Let

$$\begin{aligned} T = \{ & B(1)\#B(2)\#\ldots\#B(n)2wcw_1cw_2c\ldots cw_k \\ & \in \{0, 1, 2, c, \#\}^+ \mid n \ge 2 \\ & \& \ w \in \{0, 1\}^+ \ \& \ |w| = \lceil \log n \rceil \\ & \& \ k \ge 1 \ \& \ \forall i\,(1 \le i \le k)[\, w_i \in \{0, 1\}^+] \\ & \& \ \exists j\,(1 \le j \le k)[\, w = w_j]\}, \end{aligned}$$

where for each string v, $|v|$ denotes the length of v, and for each integer $m \ge 1$, $B(m)$ denotes the string in $\{0, 1\}^+$ that represents the integer m in binary notation (with no leading zeros), so $|B(m)| = \lceil \log m \rceil$. Then

(1) $T \in$ *strong*-2ACA(1, log n) and

(2) $T \notin \cup_{1 \le l < \infty}$ *weak*-1ACA(l, $L(n)$) for any function $L(n)$ such that $\log L(n) = o(\log n)$.

Proof: (1) One can construct a strongly log n space-bounded 2aca(1) M which acts as follows. Suppose that an input string:

$$¢\, y_1\#y_2\#\ldots y_n 2wcw_1cw_2c\ldots cw_k \$,$$

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

It is shown in Ref. 3) that the set $\{B(1)\# B(2)\#\ldots\#B(n) \mid n \ge 2\}$ can be accepted by a strongly log n space-bounded 2-way deterministic 1-counter automaton. So, M can store $\lceil \log n \rceil$ ($= |B(n)|$) stack symbols in the counter using the initial segment $B(1)\# B(2)\#\ldots\#B(n)$ of the input (Of course, M never enters an accepting state if $y_k \neq B(k)$ for some $1 \leq k \leq n$).

If M successfully complete this, then checks by using $\lceil \log n \rceil$ stack symbols stored in the counter, whether $|w| = \lceil \log n \rceil$.

After that, M again stores $\lceil \log n \rceil$ stack symbols in the counter using $|w|$ (= $\lceil \log n \rceil$) and existentially choses some j ($1 \leq j \leq k$) and checks $w = w_j$. This check can easily be done by first checking that $|w_j| = \lceil \log n \rceil$ and then universal checking that $w(p) = w_j(p)$ for each $1 \leq p \leq |w| = |w_j| = \lceil \log n \rceil$, where for each string v and each integer t ($1 \leq t \leq |v|$), $v(t)$ denote the t-th symbol (from the left) of v.

It will be obvious that $\lceil \log n \rceil$ space is sufficient, and M accepts the language T.

(2) Suppose to the contrary that there exists a weakly $L(n)$ space-bounded 1aca(l) M accepting the language T, where $\log L(n) = o(\log n)$ and $l \geq 1$ is some constant.

For each $n \geq 2$, let

$$V(n) = \{B(1)\#B(2)\#...\#B(n)2wcw_1cw_2c\ldots cw_n \in T \mid |w| = \lceil \log n \rceil \ \& \ \forall i\ (1 \leq i \leq n)[\,|w_i| = \lceil \log n \rceil]\} \text{ and}$$
$$W(n) = \{cw_1cw_2c\ldots cw_n \in \{0, 1, c\}^+ \mid \forall i\ (1 \leq i \leq n)[\,w_i \in \{0, 1\}^{\lceil \log n \rceil}]\}.$$

We consider the computations of M on the strings in $V(n)$.

Note that for each $x \in V(n)$,

- $|x| = |B(1)\#B(2)\#...\#B(n)| + (\lceil \log n \rceil + 1)(n + 1)$
  $= r(n)$
  $= O(n \log n)$ and
- there exists an accepting computation tree τ of M on x such that $|\alpha_i| \leq L(r(n))$ ($1 \leq i \leq l$), where α_i is in the ID $\ell(\pi) = (w, i, (q, (\alpha_1, \alpha_2, \ldots, \alpha_l)))$ for each node π of the tree τ.

Let $C(n)$ denote the set of all possible storage states of M when M in the computation uses at most $L(r(n))$ stack symbols in each counter, and let $u(n) = |C(n)|$. Then, $u(n) = O(L(r(n))^l)$.

For each storage state $(q, (\alpha_1, \alpha_2, \ldots, \alpha_l))$ of M and for each y in $W(n)$, let

$M_y(q, (\alpha_1, \alpha_2, \ldots, \alpha_l))$

= 1 if there exists a $(q, (\alpha_1, \alpha_2, \ldots, \alpha_l))$-accepting computation tree of M on y such that for each node π of the tree, the storage state $(q, (\alpha_1, \alpha_2, \ldots, \alpha_l))$ of the ID $\ell(\pi)$ is in $C(n)$,

= 0 otherwise.

For any strings y and z in $W(n)$, we say y and z are *M-equivalent* if for each storage state $(q, (\alpha_1, \alpha_2, \ldots, \alpha_l))$ of M with $|\alpha_i| \leq L(r(n))$ ($1 \leq i \leq l$),

$$M_y(q, (\alpha_1, \alpha_2, \ldots, \alpha_l)) = M_z(q, (\alpha_1, \alpha_2, \ldots, \alpha_l)).$$

Clearly M-equivalence is an equivalence relation on strings in $W(n)$, and there are at most

$$e(n) = O(t^{u(n)})$$

M-equivalent classes, where t is a constant. We denote these M-equivalence classes by $E_1, E_2, \ldots, E_{e(n)}$.

For each $y = wcw_1cw_2c\ldots cw_n$ in $W(n)$, let

$$b(y) = \{u \in \{0, 1\}^+ \mid \exists i\ (1 \leq i \leq n)[u = w_i]\}.$$

Furthermore, for each $n \geq 2$, let

$$R(n) = \{b(y) \mid y \in W(n)\}.$$

Then

$$|R(n)| = {}_nC_1 + {}_nC_2 + \ldots + {}_nC_n. = 2^n - 1.$$

(Intuitively, $|R(n)|$ is equal to the number of all the nonempty subsets of $\{0, 1\}^{\lceil \log n \rceil}$).

Since $e(n) = O(t^{u(n)})$, that is, $e(n) \leq t'^{u(n)}$, it follows that

$$\log\log e(n) \leq c_1 \log u(n)$$

for some constants $t > 0$, $t' > 0$ and $c_1 > 0$.

Since $u(n) = O(L(r(n))^l)$, that is, $u(n) \leq c_2 L(r(n))^l$, it follows that

$$\log u(n) = c_3 \log L(r(n))$$

for some constants $c_2 > 0$ and $c_3 > 0$.

Since $\log L(r(n)) = o(\log n)$, it follows that

$$\log L(r(n)) = o(\log r(n)).$$

Since $r(n) = O(n \log n)$, that is, $r(n) \leq c_4 n \log n$, it follows that

$$\log r(n) \leq c_5 \log n$$

for some constants $c_4 > 0$ and $c_5 > 0$. Hence, from the equations above, we have

$$\log\log e(n) \leq c \log u(n) \leq c' \log L(r(n)) = o(\log r(n)) \leq o(\log n).$$

for some constants $c > 0$ and $c' > 0$.

On the other hand, since $|R(n)| = 2^n - 1$, that is, $\log\log |R(n)| = \log n$, it follows that

$$\log\log e(n) < \log\log |R(n)|.$$

Therfore, we have

$$e(n) < |R(n)|$$

for n large enough. For such n, the must be some Q and Q' ($Q \neq Q'$) in $R(n)$ and some E_i ($1 \leq i \leq e(n)$) such that the following statement holds:

"There exist two strings $y' = B(1)\#B(2)\#...\#B(n)2wy$ and $z' = B(1)\#B(2)\#...\#B(n)2wz$ such that

(i) $|w| = \lceil \log n \rceil$,
(ii) $y, z \in W(n)$,
(iii) $b(y) = Q$ and $b(z) = Q'$,
(iv) w is in Q, but not in Q', and
(v) both y and z are in E_i (i.e., y and z are M-equivalent)" .

As is easily seen, y' is in $V(n)$, and so there exists an accepting computation tree of M on y' such that for each node π of the tree, the contents of each counter in $\ell(\pi)$ are bounded by $L(r(n))$. From this tree, we easily construct an accepting computation tree of M on z' such that for each node π of the tree, the contents of each counter in $\ell(\pi)$ are bounded by $L(r(n))$. Thus, we can conclude that z' is also accepted by M, which is a contradiction, because z' is not in T. □

From Lemma 2.1, we have:

Theorem 2.2:

$$strong\text{-}2ACA(1, \log n) - \cup_{1 \leq l < \infty} weak\text{-}1ACA(l, L(n)) = \phi$$

for any function $L(n)$ such that $\log L(n) = o(\log n)$.

Corollary 2.3: For each $m \in \{strong, weak\}$, each $l \geq 1$ and any function $L(n)$ such that $L(n) \geq \log n$ and $\log L(n) = o(\log n)$,

$$m\text{-}1ACA(l, L(n)) \subsetneq m\text{-}2ACA(l, L(n)).$$

3. Conclusion

We have investigated the accepting power of sublinear space-bounded 1-way and 2-way amca's and show that for any function $L(n)$ such that $\log L(n) = o(\log n)$,

$$strong\text{-}2ACA(1, \log n) - \cup_{1 \leq l < \infty} weak\text{-}1ACA(l, L(n)) = \phi.$$

Finally, we conclude this paper by giving two open problems relating this research:

For each $m \in \{weak, strong\}$, each $d \in \{1, 2\}$, each $l \geq 1$ and any function $\log n \leq L(n)$ such that $\log L(n) = o(\log n)$,

(1) does exist an infinite hierarchy among $m\text{-}d$ACA$(l, L(n))$'s? and

(2) is $m\text{-}d$ACA$(l, L(n))$ closed under Boolean operation, Kleene closure, concatenation, and homomorphism?

Acknowledgement

This work was supported by JSPS KAKENHI Grant Number 17K00025.

References

1) Minsky, M.L.: Recursive unsolvability of Post's problem of 'Tag' and other topics in the theory of Turing Machines, Annals of Math., Vol.74, No.3, pp.437–455 (1961).
2) 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).
3) Yoshinaga, T. and Inoue, K.: A note on alternating multi-counter automata with small space, Trans. of IPSJ., Vol.36, No.12, pp2741–2753 (1995).
4) Ito, A., Inoue, K. and Takanami, I.: A note alternating Turing machines using small space, IEICE Trans., Vol.E70, NO.10, pp.990–996 (1987).
5) Chandra, A.K., Kozen D.C. and Stockmeyer, L.J.: Alternation, J. ACM, Vol.28, No.1, pp.114–133 (1981).

(Received September 3, 2018)