Polynomial Time-Bounded Alternating Multi-Counter Automata

Kazue TAKESHIGE[*1], Tsunehiro YOSHINAGA[*2],
Jianliang XU[*3], and Katsushi INOUE[*4]

Abstract

An alternating multi-counter automaton (amca) is a generalization of a nondeterministic multi-counter automaton (nmca). The state set of an amca is partitioned into universal and existential states. Greibach showed in Ref. 1) several important properties of nmca's which have polynomial space and operate in polynomial time. In this paper, from a theoretical interest, we investigate a few fundamental properties of polynomial time-bounded amca's. We show, for example, that (1) there is a language accepted by a two-way deterministic 1-counter automaton operating in linear time, but not accepted by any one-way amca with only universal states operating in polynomial time, and (2) there is a language accepted by a two-way alternating 1-counter automaton with only universal (existential) states operating in linear time, but not accepted by any one-way amca operating in real time.

Key Words: alternating multi-counter automata, polynomial time, linear time, real time, computational complexity

1. Introduction and Preliminaries

A *multi-counter automaton* is a multi-pushdown automaton with only one pushdown symbol Z. That is, its pushdown tape is of the form Z^i.

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

We assume that M has the left endmarker "¢" and the right endmarker "$" on the input tape, and reads the input tape in two directions (that is, right or left). We also assume that in one step, M can increment or decrement the contents (that is, the length) of the 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

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

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

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

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

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

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

$$I_M(w) = (w, 0, (q_0, (\underbrace{\lambda, \lambda, \ldots, \lambda}_{k}))),$$

*1 Yamaguchi Agricultural Cooperative Association Electronic Data Processing Center Co., Ltd.

*2 Tokuyama College of Technology

*3 Ocean University of China

*4 Yamaguchi University

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

$$I_0 \vdash_M I_1 \vdash_M \cdots \vdash_M I_n \quad (n \geq 0),$$

where $I_0 = I_M(w)$.

A *computation tree* of M is a finite, nonempty labeled tree, and each node π of it is labeled with an ID, $\ell(\pi)$.

An *accepting computation tree of M on input w* is a computation tree of M whose root is labeled with the initial ID and whose leaves are all labeled with ID's relating to accepting states. We say that M *accepts* w if there is an accepting computation tree of M on w.

For each storage state s of M and 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 ID's according to accepting states.

For each $k \geq 1$, a *one-way alternating k-counter automaton* (1aca(k)) is a 2aca(k) which reads an input tape from left to right only.

For each $k \geq 1$ and each $x \in \{1, 2\}$, we denote by $x$uca(k) (resp., $x$nca(k)) an $x$aca(k) with only universal states (resp., existential states, i.e., an x-way nondeterministic k-counter automaton), and by $x$dca(k) an x-way deterministic k-counter automaton. (An $x$dca(k) is regarded as an $x$uca(k) without splitting into submachines or an $x$nca(k) whose nondeterministic choice in each step in the computation is bounded by at most one.)

Let $T(n)$ be any function. For each $x \in \{1, 2\}$ and each y $\in$ {a, u, n, d}, an $x$yca(k) M *operates in time* $T(n)$ if for each $n \geq 1$ and for each input of length n accepted by M, there is an accepting computation tree of M on the input such that the length of each computation path of the tree is at most $T(n)$. M operates in *real* (resp., *linear* and *polynomial*) *time* if $T(n) = n+1$ (resp., cn and n^r, where c and r are some positive integers).

For each $k \geq 1$, each $x \in \{1, 2\}$, each Y $\in$ {A, U, N, D}, and each $t \in$ {*real*, *linear*, *polynomial*}, xYCA(k, t) is the class of sets accepted by the corresponding x-way k-counter automata operating in t time.

Greibach showed in Ref. 1) some important results concerning the accepting powers of deterministic and nondeterministic multi-counter automata operating in polynomial time and having polynomial space. On the other hand, there are little investigations about properties of alternating multi-counter automata with polynomial time and space as far as we know. From a theoretical interest, we will investigate a few fundamental properties of polynomial time-bounded alternating multi-counter automata.

2. Results

Greibach in Ref .1) showed that

$$2\mathrm{DCA}(1, \textit{linear}) - \cup_{1 \leq k < \infty} 1\mathrm{NCA}(k, \textit{polynomial}) \neq \phi$$

using the language:

$$Ln = \{wcw^R(ca^{|w|})^{|w|} \mid w \in \{0, 1\}^*\},$$

where for any string v, $|v|$ and v^R denote the length and reversal of v, respectively. We can show the following result which is considered as the counterpart of the result above.

Theorem 2.1:

$$2\mathrm{DCA}(1, \textit{linear}) - \cup_{1 \leq k < \infty} 1\mathrm{UCA}(k, \textit{polynomial}) \neq \phi.$$

Proof: Let

$$Lu = \{wcw'(ca^{|w|})^{|w|} \mid w, w' \in \{0, 1\}^* \ \&\ w' \neq w^R\}.$$

It is sufficient to show that

(i) $Lu \in 2\mathrm{DCA}(1, \textit{linear})$ and

(ii) $Lu \notin \cup_{1 \leq k < \infty} 1\mathrm{UCA}(k, \textit{polynomial})$.

The proof of (i): A 2dca(1) M can check that a string y starts with $wcw'c$ and $w' \neq w^R$ in time proportional to $|w|^2$ using its counter C.

To check deterministically that $w' \neq w^R$, for example, M stores Z^l in C when M picks up the symbol $w(l)$, and compares with $w'(|w'|-l+1)$ by using Z^l, where for any string v, $v(i)$ denotes the i-th symbol (from the left) of v. (Note that immediately after this comparison, the input head H of M is on the symbol $w'(|w'|-l+1)$, and Z^l has been consumed, so C is void.) If $w(l) = w'(|w'|-l+1)$, then to compare $w(l+1)$ with $w'(|w'|-l)$, M moves H to the left by one cell with C keeping empty, and picks up the symbol $w'(|w'|-l)$. Then, M gets Z^{l+1} in C while moving H to the right up to the first occurrence of the symbol "c". After that, M moves H to the left endmaker $\not\in$ and compares $w(l+1)$ with $w'(|w'|-l)$ by using Z^{l+1} in C. In this manner, M can check if $w' \neq w^R$ while moving H right and left.

Furthermore, M can verify that y ends in $|w|$ occurrences of substrings in ca^*; this takes the time

proportional to y at worst.

Knowing now that

$$y = wcw' ca^{n_1} ca^{n_2} \ldots ca^{n_{|w|}},$$

M can check using C that $n_1=n_2=\ldots=n_{|w|}$ in time $O(|w|^2)$. Thus, the total time is linear in y. Therefore, Lu is in 2DCA(1, *linear*).

The proof of (ii): Suppose to the contrary that there exists a 1uca(k) M accepting Lu which operates in time n^r for some $k \geq 1$ and some constant $r > 1$. For each $n \geq 1$, let

$$V(n) = \{wcw^R(ca^{|w|})^{|w|} \mid w \in \{0, 1\}^n\}.$$

For each $x = wcw^R(ca^{|w|})^{|w|}$ in $V(n)$, there is at least one computation path of M on x in which M never enters an accepting state, because $x \notin Lu$. Fix such a computation path of M on x, and denote it by $c(x)$. Let $s(x)$ be the storage state of M just after the point where in $c(x)$ the input head H of M has left the symbol "c" between the substrings w and w^R of x. Then, the following proposition must hold:

Proposition 2.1: For any two different strings x, y in $V(n)$, $s(x) \neq s(y)$.

[**Proof:** For otherwise, suppose that $x = wcw^R(ca^{|w|})^{|w|}$, $y = w'cw'^R(ca^{|w'|})^{|w'|}$, $w \neq w'$, and $s(x) = s(y)$. Let $z = wcw'^R(ca^{|w'|})^{|w'|}$ $(w \neq w')$. Then, there is a computation path $I_M(z) \vdash_M \ldots \vdash_M (z, |wc|, s(x))$ of M on z. When, starting with ID $(z, |wc|, s(x))$, M proceeds to read the segment $w'^R(ca^{|w'|})^{|w'|}\$$ of $z\$$, there exists a sequence of steps of M in which M never enters an accepting state, since $s(x) = s(y)$.This means that z is rejected by M. This contradicts the fact that z is in Lu.]

proof of (ii) (continued): Clearly, $|V(n)| = O(2^n)$, where for any set S, $|S|$ denotes the number of elements of S. And $R(n) = O(n^{k \cdot r})$, where $R(n)$ denotes the number of possible storage states $s(n)$'s for x's in $V(n)$. Therefore, we have $|V(n)| > R(n)$ for large n, and so it follows that for such a large n, there must be two different stings x, y in $V(n)$ such that $s(x) = s(y)$. This contradicts Proposition 2.1, and completes the proof of (ii) of the theorem. □

From Greibach's and our results, we have:

Corollary 2.1: For each X $\in$ {U, N, D}, each $t \in$ {*linear*, *polynomial*}, and each $k \geq 1$,

$$1XCA(k, t) \subsetneq 2XCA(k, t).$$

Now, we have shown two-way multi-counter automata are more powerful than one-way ones in the case of determinism, nondeterminism and alternation with only universal states (except real time).

Next, we will investigate a relationship between the accepting powers of full alternating one-way and two-way multi-counter automata.

It is easily observed that the languages L_n and L_u above are in 1ACA(1, *real*). So, we need another consideration. Let

$$E = \{x_1\#x_2\#\ldots\#x_r \mid r \geq 1 \ \& \ \exists l\,(l \geq 0)[\forall i\,(1 \leq i \leq r)[x_i \in \{0, 1\}^l]]\},$$

and h be a function such that for each $x_1\#x_2\#\ldots\#x_r \in E$,

$h(x_1\#x_2\#\ldots\#x_r) = s(x_1)\#s(x_2)\#\ldots\#s(x_r)$, where

$s(a_1a_2\ldots a_{l-1}a_l) = a_1 2^{l+1} a_2 2^{l+2} \ldots a_{l-1} 2^{2l-1} a_l$, where

$a_i \in \{0, 1\}$ for each $1 \leq i \leq l$.

Now, we define the following language:

$$L = \{x_0\#h(x_1\#x_2\#\ldots\#x_r) \ \flat(0(12^{|h(x_1\#x_2\#\ldots\#x_k)|})^{|x_0|})^r \mid x_1\#x_2\#\ldots\#x_r \in E \ \& \ \exists j(1 \leq j \leq r)[x_0 = x_j]\}.$$

which is a modification of the language defined in Refs. 3)−5).

Lemma 2.1: Let L be the language defined above. Then,

(1) $L \in$ 2NCA(1, *linear*),

(2) $L \in$ 2UCA(1, *linear*), and

(3) $L \notin \cup_{1 \leq k < \infty}$1ACA($k$, *real*).

The proofs of (1) and (2): They can be shown using the ideas and the techniques in Refs 1), 4), 6), 7). So, the proofs are omitted here.

The proof of (3): Suppose that for some $k \geq 1$, there exists a 1aca(k) M which accepts L and operate in real time. For each $n \geq 1$, let

$$V(n) = \{x_0\#h(x_1\#x_2\#\ldots\#x_{2^n}) \ \flat(0(12^{|h(x_1\#x_2\#\ldots\#x_{2^n})|})^{|x_0|})^{2^n} \mid \forall i(0 \leq i \leq 2^n)[x_i \in \{0,1\}^n] \ \& \ \exists j(1 \leq j \leq 2^n)[x_0 = x_j]\} \subseteq L$$

and

$$W(n) = \{\#h(x_1\#x_2\#\ldots\#x_{2^n}) \ \flat(0(12^{|h(x_1\#x_2\#\ldots\#x_{2^n})|})^n)^{2^n} \mid \forall i(0 \leq i \leq 2^n)[x_i \in \{0,1\}^n]\}.$$

Note that for each $v \in V(n)$, there exists an accepting computation tree of M on v which has the properties:

(i) for each computation path P from the root to a leaf, the length of P is $|\¢ v \$|$ and P represents a computation in which the input head moves one cell to the right in each step, and thus

(ii) for each node π labeled with an ID which M enters just after the input head has read the initial seg-

ment x_0 of v, the length of each counter in $\ell(\pi)$ is bounded by n, since M operates in real time and we assume that M can enter an accepting state only when falling off the right endmarker \$.

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 of M on y such that for each computation path P from the root to a leaf, the length of P is $|\,\mathrm{¢}\,y\,\$\,|$ and P represents a computation in which the input head moves one cell to the right in each step,

=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 $s = (q, (\alpha_1, \alpha_2, \ldots, \alpha_k))$ of M with $0 \le |\alpha_i| \le n\ (1 \le i \le k)$. Clearly, M-equivalence is equivalence relation on strings in $W(n)$, and there are at most

$$E(n) = 2^{p(n+1)^k}$$

M-equivalence classes, where p denotes the number of states of the finite control of M. We denote these M-equivalence classes by $C_1, C_2, .., C_{E(n)}$.

For each y in $W(n)$, let

$$b(y) = \{u \in \{0,1\}^n \mid \exists i\,(1 \le i \le 2^n)[u = x_i]\}.$$

Furthermore, for each $n \ge 1$, let

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

Then,

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

So, we have $|R(n)| > E(n)$ for large n. For such n, there must be some $Q, Q'\ (Q \neq Q')$ in $R(n)$ and some $C_i\ (1 \le i \le E(n))$ such that the following statement holds:

"There are two string $y, z \in W(n)$ such that

(i) $b(y) = Q \neq Q' = b(z)$ and

(ii) $y, z \in C_i$ (i.e., y and z are M-equivalent)."

Because of (i), we can, without loss of generality, assume that there is some word $u \in \{0,1\}^n$ such that $u \in b(y) - b(z)$. Clearly, it implies that $y' = uy \in L$ and $z' = uz \notin L$.

Because of (ii), y' is accepted by M iff z' is accepted by M, which is a contradiction. □

From Lemma 2.1, we have the following theorem and corollary:

Theorem 2.2: For each X ∈ {U, N},

$2\mathrm{XCA}(1, \textit{linear}) - \cup_{1 \le k < \infty} 1\mathrm{ACA}(k, \textit{real}) \neq \phi$.

Corollary 2.2: For each $k \ge 1$ and each $t \in \{\textit{linear}, \textit{polynomial}\}$,

$1\mathrm{ACA}(k, \textit{real}) \subsetneq 2\mathrm{ACA}(k, t)$.

3. Conclusions

In this paper, we presented a few results in the accepting powers of alternating multi-counter automata operating in polynomial time. The main results are that for each X ∈ {U, N},

$2\mathrm{XCA}(1, \textit{linear}) - \cup_{1 \le k < \infty} 1\mathrm{ACA}(k, \textit{real}) \neq \phi$ and

$2\mathrm{DCA}(1, \textit{linear}) - \cup_{1 \le k < \infty} 1\mathrm{UCA}(k, \textit{polynomial}) \neq \phi$.

Finally, we conclude this paper by listing up interesting open problems. For each $k \ge 1$, each X ∈ {A, U, N, D}, and each $t \in \{\textit{linear}, \textit{polynomial}\}$,

(1) $2\mathrm{DCA}(1, \textit{linear}) - \cup_{1 \le k < \infty} 1\mathrm{ACA}(k, \textit{real}) \neq \phi$?,

(2) $2\mathrm{XCA}(k+1, \textit{linear}) - 1\mathrm{XCA}(k, t) \neq \phi$?, and

(3) are $2\mathrm{UCA}(k, t)$ and $2\mathrm{NCA}(k, t)$ incomparable?

References

1) Greibach, S.A.: Remarks on the complexity of nondeterministic counter languages, Theoretical computer Science, Vol. 1, pp. 269 −288 (1976).

2) Chandra, A.K., Kozen, D.C., and Stockmeyer, L.J.: Alternation, J. ACM, Vol. 28, PP. 114−133 (1981).

3) Chrobak, M.: Variations on the technique of Ďuriš and Galil, J. Computer and System Sciences, Vol. 30, PP. 77−85 (1985).

4) Chrobak, M.: Nondeterminism is essential for two-way counter machines, in MFCS'84, 11th Symp., Lecture Notes in Computer Science 176, Springer-Verlarg, PP. 240−244 (1984).

5) Ďuriš, P. and Galil, Z.: Fooling a two way automata or one push-down store is better than one counter for two way machines, Theoretical Computer Science, Vol. 21, PP. 39−53 (1982).

6) Yoshinaga, T. and Inoue, K.: Two-way alternating counter automata with only universal states, Research Reports of the Tokuyama College of Technology, No.26, PP.41−44 (2002).

7) Inoue, K., Tanaka, Y., Ito, A., and Wang, Y.: Self-verifying nondeterministic and Las Vegas multihead finite automata, IEICE Trans. Fundamentals, Vol.E84-A, PP. 1094 − 1101 (2001).

(Received September 3, 2004)