アルゴリズム 40－7
（1994．7．22）

同期型交代動作を行うカウンタ機械と有限オートマトン

松野浩嗣* 角川裕次** 井上克司***

* 大島商船高等専門学校 ** 広島大学工学部 *** 山口大学工学部

同期型交代性オートマトンは、交代性オートマトンの並列に動作するプロセス間に自然な形で通信を許すことで、このオートマトンを一般化したものである。本論文は、リアルタイムで動作する1方向同期型交代性マルチカウンタ機械 (1sartmcm) とプロセス数を定数に限定した同期型交代性有限オートマトン (safacl) の2つのタイプの同期型交代性オートマトンを扱う。まず1sartmcmの同期の回数をある関数によって制限した場合のカウンタの個数の階層性について調べる。マルチプロセッサ有限オートマトン (mpfa) は最も簡単な並列計算モデルのひとつであり、safacl と類似した構造をもつ。本論文では次に、mpfa と safacl の受理能力の差異について調べる。

Counter Machines and Finite Automata with Synchronized Alternation

Hiroshi MATSUNO*, Hirotsugu KAKUGAWA**, and Katsushi INOUE***

* Oshima National College of Maritime Technology
* Hiroshima University *** Yamaguchi University

Synchronized alternating automaton is a generalization of an alternating automaton enabling a simple, natural form of communications among parallel processes in an alternating computation. This paper handles two types of synchronized alternating automata. One is a synchronized alternating real-time one-way multicounter machine (1sartmcm), and the other is a synchronized alternating finite automaton whose number of processes is bounded by a constant (safacl). We firstly shows a hierarchy based on the number of counters of a 1sartmcm whose number of synchronizations is bounded by some function. A multiprocessor finite automata (mpfa) is one of the simplest parallel computation devices. We then examine the relationships between the accepting powers of mpfa's and safacl's, since the structures of mpfa's and safacl's are very similar.

1 Introduction

Alternating Turing machines were proposed in [4] as a model of parallel computation. Informally, an alternating Turing machine is a generalization of a nondeterministic Turing machine which can, at some point during a computation, split into several processes working in parallel and independently; an input is accepted if all parallel processes finish in accepting configurations. However, the alternating Turing machine is not a realistic model for real-world computers, because it does not allow any communications among its processes.

Synchronized alternating Turing machine was introduced in [6] as a generalization of alternating Turing machine enabling a simple, natural form of communication among parallel processes in an alternating computation. The synchronized alternating machine is an alternating machine with a special subset of internal states called synchronized state. Each synchronizing state is associated with a synchronizing symbol. If during the course of computation some process enters a synchronizing state, then it has to wait until each of all other processes enters an accepting state or a synchronizing state with the same synchronizing symbol. When this happens, all processes in synchronizing states are allowed to continue this computation.

A very interesting property of one-way finite automata is that neither two-way motion nor alternation helps to increase their computational powers [4]. In comparison with this fact, by adding synchronized alternation to two-way finite automata, we jump exactly two steps in Chomsky hierarchy. That is, two-way synchronized alternating finite automata recognize exactly context-sensitive languages [5].

This paper handles two types of synchronized alternating automata. One is a synchronized alternating real-time one-way multicounter machine, and the other is a leaf-size bounded synchronized alternating finite automaton.

Section 2 mainly shows a hierarchy based on the number of counters of a synchronized alternating real-time one-way multicounter machine whose number of synchronizations is bounded by some function. "Synchronization bounded alternating computation" is introduced in [8] as a new complexity measure, and the paper established an infinite hierarchy among classes of sets accepted by one-way synchronized alternating Turing machines with space and synchronization bounded between $\log\log n$ and $\log n$.

In Section 3, we investigate the relationships between the accepting powers of multiprocessor finite automata (mpfa's) and synchronized alternating finite automata with constant leaf-sizes (safacl's) . An mpfa was introduced by Buda [2], and it can be considered as one of the simplest parallel computation devices. An safacl is a synchronized alternating finite automaton whose number of parallel processes is bounded by a constant. An mpfa and an safacl have a similar structure, so it is natural to investigate a relationships between these two types of automata.

2 Real-Time Multicounter Machines

In [11], several properties of one-way alternating multicounter machines which operate in real-time are investigated and the following results are shown:

(1) For each $k \geq 1$, one-way alternating k-counter machines (1acm(k)'s) which operate in real-time (1acm($k, real$)'s) are less powerful than 1acm($k+1, real$)'s.

(2) For each $k \geq 2$, 1acm($k, real$)'s are less powerful than 1acm(k)'s which operate in linear-time.

On the other hand, in [6], a synchronized alternating device was introduced as a generalization of alternating device. The synchronized alternation enables the communication among parallel processes in alternating computations. Hromkovič and Inoue [9] introduced real-time synchronized alternating one-way multicounter machines and show that synchronized alternating one-way k-counter machines (1sacm(k)'s) which operate in real-time (1sacm($k, real$)'s) are more powerful than 1acm($k, real$)'s and 1sacm($1, real$)'s are more powerful than one-way synchronized alternating finite automata which operate in real-time.

However, it is still unknown whether or not such results as (1) and (2) above hold for synchronized alternation cases. That is,

1. Are 1sacm($k+1, real$)'s more powerful than 1sacm($k, real$)'s ?

2. Are 1sacm(k)'s which operate in linear-time more powerful than 1sacm($k, real$)'s ?

Partial solutions of these problems were given in [13], that is, the above problems positively hold for 'leaf-size bounded 1sacm(k)'. This section gives another partial solutions of these problems, that is, the above problems positively hold for 'synchronization bounded 1sacm(k)'.

A one-way multicounter machine is a one-way multipushdown machine whose pushdown stores operate as counters, i.e. have a single-letter alphabet. (See [1] for formal definitions of one-way multicounter machines.)

An *alternating one-way multicounter machine* (1amcm) [11] M is a generalization of a one-way nondeterministic multicounter machine in such a sense as in [4]. 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 a 1amcm has the right endmarker '$' on the input tape. We also assume that in one step a 1amcm can make an increment or a decrement in the contents of each counter by at most one.

A *synchronized alternating one-way multicounter machine* (1samcm) M is a 1amcm some states of which have *synchronizing symbols* (sync symbols) from some given finite set. That is, an internal state of a 1samcm can be either an internal state of M or a pair (internal state of M, sync symbol). The latter is called a *synchronizing state* (sync state), and the instantaneous descriptions (IDs, see

below) in the correspondence to the sync state are called the *synchronizing instantaneous descriptions* (sync IDs).

For each $k \geq 1$, we denote a synchronized alternating one-way k-counter machine by 1sacm(k).

An *instantaneous description* (ID) of a 1sacm(k) M is an element of

$$\Sigma^* \times N \times S_M$$

where Σ ($\$ \notin \Sigma$) is the input alphabet of M, N denotes the set of all positive integers and $S_M = Q \times (N \cup \{0\})^k$ (where Q is the set of states of the finite control of M). The first and second components x and i of ID $I=(x,i,(q,(j_1,...,j_k)))$ represent the input string and the input head position, respectively. [1] The third component $(q,(j_1,...,j_k))$ 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 $x \in \Sigma^*$ is $I_M(x) = (x, 1, (q_0, (0, ..., 0)))$, where q_0 is the initial state of M. 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 sequence of IDs of M $I_0, I_1, ..., I_m$ ($m \geq 0$), is called a *sequential computation* of M if $I_0 \vdash_M I_1 \vdash_M \cdots \vdash_M I_m$. If $I_0 = I_M(x)$ for some x, we call this sequence a *computation path* (sometimes called "process") of M on x. Let $\bar{I}$ be a sequential computation of M and $I_1, I_2, ..., I_r$ be a subsequence of $\bar{I}$ which consists of all sync IDs in $\bar{I}$. For each j ($1 \leq j \leq r$), let S_j be the sync symbol in I_j. Then the sequence $S_1, S_2, ..., S_r$ is called the *synchronizing sequence* (sync sequence) of $\bar{I}$.

A *computation tree* of M is a finite, nonempty labeled tree with the following properties:

1. Each node v of the tree is labeled with an ID $\ell(v)$.
2. If v is an internal node (a non-leaf) of the tree, $\ell(v)$ is universal and $\{I \mid \ell(v) \vdash_M I\} = \{I_1, ..., I_k\}$, then v has exactly k children $v_1, ..., v_k$ such that $\ell(v_i) = I_i$ ($1 \leq i \leq k$).
3. If v is an internal node of the tree and $\ell(v)$ is existential, then v has exactly one child u such that $\ell(v) \vdash_M \ell(u)$.
4. For any two sync sequences $S = S_1, ..., S_p$ and $T = T_1, ..., T_r$ corresponding to two paths of the tree beginning at the root, it must be satisfied that $S_i = T_i$ for each $i \in \{1, 2, ..., \min\{p, r\}\}$.

A *computation tree of M on input x* is a computation tree of M whose root is labeled with $I_M(x)$. An *accepting* computation tree of M on x is a computation tree of M on x whose leaves are all labeled with accepting IDs. We say that M *accepts* x if there is an accepting computation tree of M on x. Define $\mathrm{T}(M) = \{x \in \Sigma^* \mid M \text{ accepts } x\}$.

A 1sacm(k) M *operates in time $t(n)$* if for each input x accepted by M, there is an accepting computation tree of M on x such that the length of each computation path of the tree is at most $t(|x|)$. M operates in *real-time (linear-time)* if $t(n) = n + 1$ ($t(n) = cn$ for some positive constant c).

In order to avoid misunderstandings, we give a precise definition of an accepting computation of a real-time synchronized alternating machine M. "Real-time" means that the machine moves its input head on an input to the right in each computation step. We assume that the right endmarker '\$' is attached to the right of an input of M, and the input head of M fall off the right endmarker '\$' after reading it.

We next give the definitions of synchronization bounded 1sacm(k).

Let $f : N \to R$ be a function. We say that a 1sacm(k) is *$f(n)$ synchronization bounded* if for each $n \geq 1$ and for each input w (accepted by M) of length n, there is an accepting computation tree T of M on w such that the length of the longest sync sequence of T is bounded by $\lceil f(n) \rceil$.

We use the following notations to represent different kinds of 1samcm's.

1sacm($k, real, f(n)$)	$f(n)$ synchronization bounded 1sacm(k) which operates in real-time
1sacm($k, linear, f(n)$)	$f(n)$ synchronization bounded 1sacm(k) which operates in linear-time
1sucm($k, real, f(n)$)	1sacm($k, real, f(n)$) with only universal states
1sucm($k, linear, f(n)$)	1sacm($k, linear, f(n)$) with only universal states

Define

1SACM($k, real, f(n)$) = {T | T=T(M) for some 1sacm($k, real, f(n)$) M }

1SACM($k, linear, f(n)$) = {T | T=T(M) for some 1sacm($k, linear, f(n)$) M }

Similarly, we let 1SUCM(k, *real*, $f(n)$) (1SUCM(k, *linear*, $f(n)$)) denote the class of sets accepted by 1sucm(k, *real*, $f(n)$)'s (1sucm(k, *linear*, $f(n)$)'s).

We show the several hierarchies based on the number of counters of synchronized alternating multicounter machines when the number of synchronizations is bounded by some function of the input size.

We need some definitions. For each storage state $(q,(j_1,...,j_k))$ of M and each $w \in \Sigma^+$, let a $(q,(j_1,...,j_k))$-*computation tree* of M on w be a computation tree of M whose root is labeled with the configuration $(w, 1, (q,(j_1,...,j_k)))$. A $(q,(j_1,...,j_k))$-*accepting computation tree* of M on w is a $(q,(j_1,...,j_k))$-computation tree of M on w whose leaves are all labeled with accepting IDs.

Theorem 1 *Let $f : N \to R$ be any function such that $\lim_{n\to\infty} [f(n)/\log n] = 0$ and $f(n) \geq 0$. Then, for each $k \geq 1$, $1SACM(k, real, f(n)) \subsetneq 1SACM(k+1, real, f(n))$.*

[1] We note that $1 \leq i \leq |x| + 2$, where for any string w, $|w|$ denotes the length of w. '1', '$|x|$+1' and '$|x|$+2' represent the positions of the leftmost symbol of x, the right endmarker '\$', and the immediate right to '\$'.

Proof. For each integer $k \geq 1$, let

$$\begin{aligned} T_1(k) &= \{\#^n w\#w_1\#w_2\#\cdots\#w_r \in \{0,1,\#\}^+ \mid \\ &\quad n \geq 1 \ \&\ w \in \{0,1\}^+ \ \&\ |w|{=}n \ \&\ r{=}(n+1)^k \ \& \\ &\quad \forall i(1 \leq i \leq r)[w_i \in \{0,1\}^+ \ \&\ |w_i| = n] \ \& \\ &\quad \exists j(1 \leq j \leq r)[w = w_j]\}. \end{aligned}$$

It is shown in [11] that $T_1(k)$ is accepted by a one-way alternating $k+1$ counter machine which operates in real-time. Thus, we can get that $T_1(k) \in 1SACM(k{+}1, real, 0)$.

We next show that $T_1(k) \notin 1SACM(k, real, f(n))$. Suppose that there exists a $1\text{sacm}(k, real, f(n))$ that accepts $T_1(k)$. For each $n \geq 1$, let

$$\begin{aligned} V(n) &= \{\#^n w\#w_1\#w_2\#\cdots\#w_{g(n)} \mid \\ &\quad \forall i(1 \leq i \leq g(n))[w_i \in \{0,1\}^+ \ \&\ |w_i| = n] \ \& \\ &\quad \exists j(1 \leq j \leq g(n))[w = w_j]\} \subseteq T_1(k), \end{aligned}$$

where $g(n) = (n+1)^k$ and

$$\begin{aligned} W(n) &= \{\#w_1\#w_2\#\cdots\#w_{g(n)} \mid \\ &\quad \forall i(1 \leq i \leq g(n))[w_i \in \{0,1\}^+ \ \&\ |w_i| = n]\}. \end{aligned}$$

Note that for each $x = \#^n w\#w_1\#w_2\#\cdots\#w_{g(n)} \in V(n)$,

1. $|x| = 2n + (n+1)^{k+1} = r(n)$, and
2. there exists an accepting computation tree of M on x which has the following properties:
   (a) for each computation path p from the root to a leaf, the length of p is $|x\$| = r(n)+1$ and p represents a computation in which the input head moves one square to the right in each step and, thus,
   (b) for each node π labeled with an ID which M enters just after the input head has read the initial segment $\#^n w$ of x, the contents of each counter in $\ell(\pi)$ is bounded by $2n$,
   (c) the length of the sync sequence of the tree is bounded by $f(r(n))$,

since M operates in real-time and we assume that M can enter an accepting state only when falling off the right end-marker '\$'.

Let Q be the set of all the storage states $(q, (j_1, ..., j_k))$ with $\forall i(1 \leq i \leq k)[0 \leq j_i \leq 2n]$, and S be the set of all the sequences of sync symbols (of M) of length at most $\lceil f(r(n)) \rceil$. Then, for each y in $W(n)$, let the mapping $M_y : Q \to 2^S$ be defined as follows : for each $(q, (j_1, ..., j_k)) \in Q$,

- $\sigma \in M_y(q, (j_1, ..., j_k)) \Leftrightarrow$ there exists a $(q, (j_1, ..., j_k))$-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 $|y\$| = r(n) + 1 - 2n$ and p represents a computation in which the input head moves one square to the right in each step, and the longest sync sequence of the tree is σ,
- $M_y(q, (j_1, ..., j_k)) = \phi \Leftrightarrow$ for any sync sequence α with the length at most $f(r(n))$, there exists no $(q, (j_1, ..., j_k))$-accepting computation tree of M on y such that the sync sequence is α and for each computation path p from the root to a leaf, the length of p is $|y\$| = r(n) + 1 - 2n$ and p represents a computation in which the input head moves one square to th right in each step.

For any two strings y, z in $W(n)$, we say that y and z are *M-equivalent* if for each storage state $(q, (j_1, ..., j_k))$ of M with $0 \leq j_i \leq 2n$ $(1 \leq i \leq k)$, $M_y(q, (j_1, ..., j_k)) = M_z(q, (j_1, ..., j_k))$. Clearly, M-equivalence is an equivalence relation on strings in $W(n)$, and there are at most

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

M-equivalence classes, where $c(n) = 1 + a + a^2 + \cdots + a^{f(r(n))}$ (where a is the number of sync symbols of M) and t 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 $y = \#w_1\#w_2\#\cdots\#w_{g(n)}$ in $W(n)$, let $b(y) = \{u \in \{0,1\}^+ \mid \exists i(1 \leq i \leq g(n))[u = w_i]\}$. Furthermore, for each $n \geq 1$, let $R(n) = \{b(y) \mid \exists y \in W(n)\}$. Then,

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

From the assumption that $\lim_{n\to\infty}[f(n)/\log n] = 0$, it follows that $\lim_{n\to\infty}[f(r(n))/\log n] = 0$. Therefore, we have $|R(n)| > E(n)$ for large n. For such n, there must be some $U, U' (U \neq U')$ 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
  (i) $b(y) = U \neq U' = b(z)$ 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 string $w \in \{0,1\}^+$ such that $|w| = n$ and $w \in b(y) - b(z)$. Clearly, it implies that $y' = \#^n wy \in T_1(k)$ and $z' = \#^n wz \notin T_1(k)$. But because of (ii), y' is accepted by M iff z' is accepted by M, which is a contradiction. This completes the proof of the theorem. □

Finally, we show that for 1samcm's bounded in the number of synchronization $f(n)$ such that $\lim_{n\to\infty}[f(n)/\log n] = 0$ and $f(n) \geq 0$, linear-time is more powerful than real-time.

Theorem 2 *Let $f : N \to R$ be any function such that $\lim_{n\to\infty}[f(n)/\log n] = 0$ and $f(n) \geq 0$ $(n \geq 1)$. Then for each $k \geq 2$,*

(1) $1SACM(k, real, f(n)) \subsetneq 1SACM(k, linear, f(n))$,
(2) $1SUCM(k, real, f(n)) \subsetneq 1SUCM(k, linear, f(n))$,
(3) $\bigcup_{1\leq r<\infty} 1SACM(r, real, f(n)) \subsetneq \bigcup_{1\leq r<\infty} 1SACM(r, linear, f(n))$, *and*
(4) $\bigcup_{1\leq r<\infty} 1SUCM(r, real, f(n)) \subsetneq \bigcup_{1\leq r<\infty} 1SUCM(r, linear, f(n))$.

Proof. Let

$$T_2 = \{w\#0^{m_1}\#0^{m_2}\#\cdots\#0^{m_r} \mid w \in \{0,1\}^+ \ \& \ r \geq 1 \ \& \ \forall i(1 \leq i \leq r)[m_i \geq 1] \ \& \ \exists j(1 \leq j \leq r)[m_j = N(w)+1]\},$$

where $N(w)$ denotes the integer represented by w as a binary number (with the least significant bit in the right most position). It is shown in [11] that T_2 can be accepted by a one-way deterministic 2-counter machine which operates in linear-time.

We show below, by using the same technique as in the proof of Theorem 1, that $T_2 \notin \bigcup_{1 \leq r < \infty} 1SACM(r, real, f(n))$ for any function $f(n)$ such that $\lim_{n\to\infty}[f(n)/\log n] = 0$.

Suppose that for some $k \geq 1$, there exists a 1sacm(k,real,$f(n)$) M which accepts T_2. For each $n \geq 1$, let

$$V(n) = \{w\#0^{m_1}\#0^{m_2}\#\cdots\#0^{m_{g(n)}} \mid |w| = n \ \& \ w \in \{0,1\}^+ \ \& \ \forall i(1 \leq i \leq g(n))[1 \leq m_i \leq 2^n] \ \& \ \exists j(1 \leq j \leq g(n))[m_j = N(w)+1]\} \subseteq T_2,$$

where $g(n) = 2^n$ and

$$W(n) = \{\#0^{m_1}\#0^{m_2}\#\cdots\#0^{m_{g(n)}} \mid \forall i(1 \leq i \leq g(n))[1 \leq m_i \leq 2^n]\}.$$

Similarly, as in the proof of Theorem 4, we can divide $W(n)$ into at most

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

M-equivalence classes, where $c(n) = 1 + a + a^2 + \cdots + a^{f(r(n))}$ (where a is the number of sync symbols of M) and t denotes the number of states of the finite control of M.

For each $y = \#0^{m_1}\#0^{m_2}\#\cdots\#0^{m_{g(n)}} \in W(n)$, let

$$b(y) = \{m \in N \mid \exists i(1 \leq i \leq g(n))[m = m_i]\}.$$

Furthermore, for each $n \geq 1$, let $R(n) = \{b(y) \mid \exists y \in W(n)\}$. Then

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

From the assumption that $\lim_{n\to\infty}[f(n)/\log n] = 0$, it follows that $\lim_{n\to\infty}[f(g(n))/n] = 0$. Therefore, we have $|R(n)| > E(n)$ for large n. Now the proof of $T_2 \notin \bigcup_{1 \leq r < \infty} 1SACM(r, real, f(n))$ can be completed in the same way as in the proof of Theorem 4. □

3 Multiprocessor Automata and Synchronized Alternating Finite Automata

A multiprocessor finite automaton (mpfa) was introduced in [2] and it can be considered as one of the simplest parallel computation devices; it consists of more than one finite automata, called 'processors', reading information from the input string simultaneously, and the switching function that determines whether each processor does the action according to the transition functions or takes a rest.

On the other hand, Hromkovič et.al. studied the parallel complexity classes of synchronized alternating finite automata by introducing these automata with a constant number of processes (i.e., constant leaf-size) in [7]. They nicely characterized these leaf-size bounded automata in terms of multihead nondeterministic finite automata, and thus obtained tight hierarchies of these machines on the number of processes.

The mpfa has a structure similar to the synchronized alternating finite automaton with constant leaf-sizes, so it is natural to investigate a relationship between these automata.

A one-way deterministic k-processor finite automaton is a device $M = (k, Q, E, g, h, v_0)$, where k is the number of finite automata called *processors*, Q is a finite set of *states*, E is an input *alphabet*, g is a mapping from $Q \times (E \cup \{¢, \$\})$ to $Q \times \{0,1\}$ called the *transition function*, h is a mapping from $\{1, 2, ..., k\} \times Q^k$ to $\{0,1\}$ called the *switching function* and $v_0 \in Q^k$ is the k-tuple of the *initial states*. E does not contain '¢' and '\$', which are the left and right endmarkers, respectively. For all $p, p' \in Q$, $g(p, ¢) = (p', d)$ implies that $d \geq 0$, and $g(p, \$) = (p', d)$ implies that $d = 0$.

M is a one-way nondeterministic k-processor finite automaton, if g maps from $Q \times (E \cup \{¢, \$\})$ to $2^{Q \times \{0,1\}}$. M is a two-way deterministic k-processor finite automaton if g maps from $Q \times (E \cup \{¢, \$\})$ to $Q \times \{-1, 0, 1\}$. M is a two-way nondeterministic k-processor finite automaton if g maps from $Q \times (E \cup \{¢, \$\})$ to $2^{Q \times \{-1,0,1\}}$.

An input to mpfa M is a string $x = x_1 x_2 \cdots x_n$ ($n \geq 0$ & $\forall i(1 \leq i \leq n)[x_i \in E]$) which is written on the read only tape. We say that the position of the head of a processor is i if and only if the head is on the i-th symbol of $¢x\$$ ($¢$ is the first symbol). A *configuration* of M is a $2k$-tuple $(q_1, ..., q_k, i_1, ..., i_k)$, where $q_l \in Q$, $1 \leq i_l \leq |x| + 2$ for all $1 \leq l \leq k$. The configuration of M represents the states of processors and the positions of heads. An *initial configuration* of M is $C_0 = (q_0^1, ..., q_0^k, 1, ..., 1)$, where $v_0 = (q_0^1, ..., q_0^k)$. The configuration at time 0 is the initial configuration. Let $C_t = (p_1, ..., p_k, i_1, ..., i_k)$ be a configuration at time t ($t \geq 0$). Then, for each l ($1 \leq l \leq k$), the configuration $C_{t+1} = (q_1, ..., q_k, j_1, ..., j_k)$ at time $t+1$ is defined as:

$$(q_l, j_l) = \begin{cases} (r, i_l + d) & h(l, p_1, ..., p_k) = 1,\ (r, d) = g(p_l, e_{i_l}), \\ & r \in Q,\ 1 \leq i_l \leq |x| + 2, \text{ and } e_{i_l} \text{ is th} \\ & \text{symbol of } ¢x\$, \\ (p_l, i_l) & h(l, p_1, ..., p_k) = 0. \end{cases}$$

The relationship of this transition is denoted $C_t \vdash_{M,x} C_{t+1}$, and the reflexive and transitive closure of $\vdash_{M,x}$ is denoted $\vdash_{M,x}^*$. A configuration $C = (q_1, ..., q_k, i_1, ..., i_k)$ is an *accepting configuration* if and only if $h(l, q_1, ..., q_k) = 0$ for all $l = 1, ..., k$. We say that M *accepts* an input $x \in E^*$ if and only if there exists an accepting configuration C such that $C_0 \vdash_{M,x}^* C$.

The mpfa's above are sometimes called 'halting type mpfa's' in particular. In addition to this type of mpfa's, we introduce another type of mpfa's called 'accepting type mpfa's'. That is, we define an accepting type mpfa by changing the accepting condition of an mpfa as follows: 'an mpfa accepts an input if and only if all the processes enter accepting states'.

An *accepting type* one-way deterministic k-processor finite automaton is a device $M = (k, Q, E, g, h, v_0, F)$, where $F(\subseteq Q)$ is a set of accepting states. Other components are the same as the halting type mpfa. The configuration, the initial configuration C_0, the relations of configurations $\vdash_{M,x}$ and $\vdash^*_{M,x}$ can be defined as the above.

A configuration $C = (q_1, ..., q_k, i_1, ..., i_k)$ is an accepting configuration if and only if $q_l \in F$ for all $l = 1, 2, ..., k$. We say that M accepts an input $x \in E^*$ if and only if there exists an accepting configuration C such that $C_0 \vdash^*_{M,x} C$.

We below show the notations which are used in this paper.

1dp(k)fa one-way deterministic halting type k-processor finite automaton

1np(k)fa one-way nondeterministic halting type k-processor finite automaton

2dp(k)fa two-way deterministic halting type k-processor finite automaton

2np(k)fa two-way nondeterministic halting type k-processor finite automaton

Furthermore, 1da(k)fa, 1na(k)fa, 2da(k)fa, and 2na(k)fa represent accepting type mpfa's which correspond to the above halting type mpfa's.

For an mpfa M, let $T(M) = \{x \in E^* \mid M \text{ accepts } x\}$. The class of sets accepted by 1dp(k)fa's which is defined by $\{T(M) \mid M \text{ is a 1dp}(k)\text{fa}\}$ is denoted 1DP(k)FA. Similarly, classes 1NP(k)FA, 2DP(k)FA, 2NP(k)FA, 1DA(k)FA, 1NA(k)FA, 2DA(k)FA, and 2NA(k)FA are defined.

A *synchronized alternating one-way finite automaton* (1safa) can be defined as the 1sacm(k) in the previous section. That is, a 1safa is a 1sacm(k) which has no counters. A synchronized alternating *two-way* finite automaton (2safa) is the same as the 1safa except that the input head of the 2safa can move in two directions.

We then introduce 'leaf-size' for 1safa's and 2safa's. Let $L : N \to R$ be a function. For each tree t, let LEAF(t) denote the leaf-size of t (i.e., the number of leaves of t). We say that a 1safa (2safa) M is $L(n)$ *leaf-size bounded* if, for each n and for each input x of length n, if x is accepted by M, then there is an accepting computation tree t of M on x such that LEAF(t)$\leq \lceil L(n) \rceil$. An $L(n)$ leaf-size bounded 1safa (2safa) is denoted 1safa($L(n)$) (2safa($L(n)$)). In this paper, we only consider 1safa's and 2safa's which have constant leaf-sizes. We use the following notations.

1safa(k) k leaf-size bounded synchronized alternating one-way finite automaton

2safa(k) k leaf-size bounded synchronized alternating two-way finite automaton

1sufa(k) 1safa(k) with only universal states

2sufa(k) 2safa(k) with only universal states

We then investigate the relationships between multiprocessor finite automata and synchronized alternating finite automata with constant leaf-sizes. We first consider the case when nondeterministic actions are permitted.

Theorem 3 *For each $k \geq 1$,*
1NP(k)FA=1SAFA(k) and
2NP(k)FA=2SAFA(k).

Proof. For each X$\in \{1, 2\}$, let XNH(k)FA denote the class of sets accepted by X-way nondeterministic k-head finite automata. It is known that

- XNP(k)FA=XNH(k)FA [3]
- XNH(k)FA=XSAFA(k) [7]

for each X$\in \{1, 2\}$. From these facts, we can see that the theorem holds. □

We next consider the case when nondeterministic actions are not permitted.

Lemma 1 *1DP(2)FA$-\bigcup_{1\leq k<\infty}$2SUFA($k$)$\neq \phi$.*

Proof. Let $T_3 = \{w2w' \mid w \neq w' \ \& \ w, w' \in \{0,1\}^*\}$. We can show that $T \in$1DP(2)FA by using the same technique as in the proof of Theorem 4 in [12]. On the other hand, in Lemma 3.4 in [10], it is shown that the set T_3 can not be accepted by any $S(n)$ space bounded two-way Turing machines with only universal states. Thus, we can get that $T_3 \notin \bigcup_{1\leq k<\infty}$2SUFA($k$). □

Theorem 4
$\bigcup_{1\leq k<\infty}$*1SUFA(k)*$\subsetneq \bigcup_{1\leq k<\infty}$*1DP($k$)FA and*
$\bigcup_{1\leq k<\infty}$*2SUFA(k)*$\subsetneq \bigcup_{1\leq k<\infty}$*2DP($k$)FA.*

Proof. It is shown in Corollary 1 in [12] that $\bigcup_{1\leq k<\infty}$1DP(k)FA=$\bigcup_{1\leq k<\infty}$1DA(k)FA. Similarly, we can show that $\bigcup_{1\leq k<\infty}$2DP(k)FA=$\bigcup_{1\leq k<\infty}$2DA(k)FA. From these facts and the lemma above, it is sufficient to show that $\bigcup_{1\leq k<\infty}$XSUFA(k)$\subseteq \bigcup_{1\leq k<\infty}$XDA($k$)FA for each X$\in \{1, 2\}$.

Let M be a 1sufa(k), $k \geq 2$. We construct a 1da(2^{k-1})fa N which simulates M. Without loss of generality, we assume that each node of a computation tree of M labeled with a universal configuration has exactly two children. Each processor of N contains the distinct number from 0 to $2^{k-1} - 1$ as a component of a state, and the processor with the component i is denoted P_i. Furthermore, each processor of N has another number sn as a component of a state called *threshold number*. At the beginning of a computation of N, the threshold numbers of all processors are set to $sn = 2^{k-2} - 1$.

Until M firstly enters a universal state from an initial state, all 2^{k-1} processors of N simulates the action of M.

After this (or the initial state of M is a universal state), for each i $(1 \leq i \leq 2^{k-1})$, each processor P_i such that $i \leq sn$ $(=2^{k-2}-1)$ enters an ID of the left child of the node with that universal state (called *left ID*), and each processor P_i such that $i > sn$ enters an ID of the right child of the node (called *right ID*). (That is, 2^{k-1} processors of N split into two groups each of which consists of 2^{k-2} processors.) Afterward, the threshold numbers of the processors in the left ID are all reset to $sn = sn - 2^{k-3}$ and the threshold numbers of the processors in the right ID are all reset to $sn = sn + 2^{k-3}$.

For each ℓ $(1 \leq \ell \leq k-2)$, suppose that M enters an ℓ-th universal state numbered from an initial state. Then, for each of $2^{k-\ell}$ processors of N simulating the action of M, the threshold number of the processor P_j such that $j \leq sn$ is reset to $sn = sn - 2^{k-\ell}$, and the processor P_j enters a left ID. The threshold numbers of other processors (that is, processors P_j's such that $j > sn$) are reset to $sn = sn + 2^{k-\ell}$, and each of these processors enters a right ID. (That is, $2^{k-\ell}$ processors of N split into two groups each of which consists of $2^{k-\ell-1}$ processors.)

Suppose that M enters a $(k-2)$-th universal state. In this case, it is easily seen that the number of processors of N that simulate the actions of M is two. Then, one processor P_{sn} enters a left ID and the other processor P_{sn+1} enters a right ID. We must note that the number of universal IDs in any computation path of M is at most $k-1$, since the leaf-size of M is k.

Suppose that M enters an accepting ID (rejecting ID) in some computation path. Then, each processor of N simulating the action of M in the computation path enters the same accepting ID (rejecting ID) as the above.

Suppose that M enters a sync ID in some computation path. Then, N makes each processor of N simulating the action of M in the computation path inactive by the switching function. When some computation path of M enters a sync ID with sync symbol s, the switching function of N makes processors simulating the actions in sync IDs active only if each of computation paths of M enters a sync ID with the same sync symbol s or an accepting ID.

From the actions described above, it is easy to see that all processors of N enter accepting states only if all computation paths of M enter accepting IDs. It is obvious that N accepts T(M). Thus, 1SUFA(k)$\subseteq$1DA(2^{k-1})FA, it follows that $\bigcup_{1\leq k<\infty}$1SUFA(k)$\subseteq \bigcup_{1\leq k<\infty}$1DA($k$)FA. □

We next consider the problems such that

For each X$\in${1,2} and $k \geq 2$,

- Is XSUFA(k)$\subsetneq$XDP(k)FA ?
- Is XSUFA(k)$\subsetneq$XDA(k)FA ?

Unfortunately, we only know the proper inclusion only in the case of $k = 2$ for accepting type mpfa's. The proof of the following theorem is left to readers.

Theorem 5
1SUFA(2)$\subsetneq$1DA(2)FA and
2SUFA(2)$\subsetneq$2DA(2)FA.

By combining the above theorem and Theorem 2 in [12], we can prove the following theorem. (Theorem 2 in [12] states only the case of one-way, but the theorem is applicable to the two-way case.)

Theorem 6
1SUFA(2)$\subsetneq$1DP(3)FA and
2SUFA(2)$\subsetneq$2DP(3)FA.

4 Conclusion

We firstly showed that an additional counter increases the accepting power of a synchronized alternating real-time one-way multicounter machine (1sartmcm) whose the number of synchronizations is $o(n)$. We also showed that a linear-time machine is more powerful than a real-time machine for a synchronized alternating one-way multicounter machine whose number of synchronizations is $o(n)$. Similar results as the above could be found in [13], that is, the paper states the proper inclusions as the above for leaf-size bounded 1sartmcm's. It is unknown whether or not the similar inclusions hold for non-restricted 1sartmcm's.

We next investigated the relationships between the accepting powers of a multiprocessor finite automaton and a synchronized alternating finite automaton with constant leaf-size. We showed that the accepting powers of a nondeterministic k-processor finite automaton and a k leaf-size bounded synchronized alternating finite automaton are the same. On the other hand, we also got a result that if we exclude nondeterministic actions from the both automata, a different situation occurs, that is, a deterministic multiprocessor finite automaton is more powerful than a leaf-size bounded synchronized finite automaton with only universal states. However, the following is left as an open problem. "Is there a proper inclusion between the accepting powers of a deterministic k-processor finite automaton and a k leaf-size bounded synchronized alternating finite automaton with only universal states?"

Acknowledgements

The authors would like to thank Prof. Itsuo Takanami at Iwate University for his encouragement and helpful discussions. One of the authors, H. Matsuno expresses his gratitude to Prof. Satoru Miyano at Kyushu University for his constructive comments.

References

[1] R. Book and S. Ginsburg, "Multi-stack-counter languages", *Mathematical Systems Theory 6*, pp.37-48, 1972.

[2] A.C. Buda, "Multiprocessor automata", *Information Processing Letters 25*, pp.257-261, 1987.

[3] A.C. Buda, "Nondeterministic multihead and multiprocessor automata", *Comptes reudus de Acddéme Bulgare des Sciences* 41 (10), pp.5-7, 1988.

[4] A.K. Chandra, D.C. Kozen, and L.J. Stockmeyer, "Alternation", *J.ACM 28 (1)*, pp.114-133, 1981.

[5] J. Dassow, J. Hromkovič, J. Karhumäki, B. Rovan, and A. Slobodová, "On the power of synchronization in parallel computations" *Discrete Applied Mathematics 32* pp.155-182, 1991.

[6] J. Hromkovič, "How to organize the communication among parallel processes in alternating computations", *unpublished manuscript, Comenius University*, 1986.

[7] J. Hromkovič, J. Karhumäki, B. Rovan, and A. Slobodová, "On the power of synchronization in parallel computations", *Technical Report, Comenius University, Bratislava, Czechoslovakia, Department of Theoretical Cybernetics and Institute of Computer Science*, 1989.

[8] J. Hromkovič, K. Inoue, B. Rovan, A. Slobodová, I. Takanami, and K.W. Wagner, "On the power of one-way synchronized alternating machines with small space", *International Journal of Foundations of Computer Science 3 (1)*, pp.65-79,1992.

[9] J. Hromkovič and K. Inoue, "A note on realtime one-way synchronized alternating one-counter automata", *Theoretical Computer Science 108*, pp.393-400, 1993.

[10] O.H. Ibarra and N.Q. Trân, "On space-bounded synchronized alternating Turing machines", *Theoretical Computer Science 99*, pp.243-264, 1992.

[11] K. Inoue, A. Ito, and I. Takanami, "A note on real-time one-way alternating multicounter machines", *Theoretical Computer Science 88*, pp.287-296,1991.

[12] H. Kakugawa, H. Matsuno, K. Inoue, and I. Takanami, "Some properties of one-way multiprocessor finite automata", *The Transactions of the IEICE*, J75-D-I(11), pp.963-972, 1992.

[13] H. Matsuno, K. Inoue, and I. Takanami, "Leaf-size bounded real-time synchronized alternating one-way multicounter machines", *IEICE Transactions on Information and Systems*, E77-D(3), pp.351-354, 1994.