数理解析研究所講究録
第 871 巻 1994 年 1-7



# A Hierarchy Result of Cooperating Systems of Two-Way Counter Machines

王耀 ( Yue Wang ) 井上克司 ( Katsushi Inoue ) 高浪五男 ( Itsuo Takanami )
*Department of Computer Science and Systems Engineering,*
*Faculty of Engineering, Yamaguchi University,*
*Ube, 755 Japan*

## 1 Introduction

Cooperating systems of counter machines were introduced in [3] as language acceptors, which may be considered as a model of parallel computation. Informally, a cooperating system of $k$ counter machines consists of $k$ (one-)counter machines $CM_1$, $CM_2$, $\cdots$, $CM_k$, and a read-only input tape where these counter machines independently work step by step. Each step is assumed to require exactly one time for its completion. Those counter machines whose input heads scan the same cell of the input tape can communicate with each other, that is, every counter machine is allowed to know the internal states and the signs (positive or zero) of counters of other counter machines on the cell it is scanning at the moment. The input tape holds a string of input symbols delimited by left and right endmarkers. The system starts with each $CM_i$ on the left endmarker in its initial state with the counter empty, and accepts the input tape if each $CM_i$ enters an accepting state and halts when reading the right endmarker. In [3], several basic properties of cooperating systems of one-way counter machines were investigated, and some relations between cooperating systems of counter machines and multicounter machines regarding the polynomial time (space) complexity were established.

In this paper we show that for languages over a one-letter alphabet, cooperating systems of $k+1$ two-way counter machines are more powerful than cooperating systems of $k$ two-way counter machines, with space bound $n$. The method used here is based on a transformational method used in [1]. An open problem in [1] is whether for languages over a one-letter alphabet, two-way nondeterministic $(k+1)$-counter machines are more powerful than two-way nondeterministic $k$-counter machines, with $n$ space bound. (For the deterministic case, the answer is positive [1].) Our result can also be considered as an answer for another version of this problem.

## 2 Preliminaries

Let $M$ be a cooperating system of counter machines (see [3] for the formal definition of cooperating system of counter machines), and let $L(n)$ be a function from $\mathcal{N}$ to $\mathcal{N}$, where $\mathcal{N}$ is the set of natural numbers. $M$ is said to be $L(n)$-space bounded if for each input $w$ accepted by $M$, there is a computation of $M$ on $w$ in which each counter requires space not exceeding $L(|w|)$.

We denote a cooperating system of $k$ two-way deterministic (nondeterministic) counter machines by CS-DCM($k$) (CS-NCM($k$)), and for each M$\in${CS-DCM($k$), CS-NCM($k$)}, if M is $L(n)$-space bounded, we denote it by M[Space($L(n)$)].

Let $\mathcal{L}_\Sigma$[CS-DCM($k$)[Space($n$)]] ($\mathcal{L}_\Sigma$[CS-NCM($k$)[Space($n$)]]) denote the class of languages over the alphabet $\Sigma$ accepted by CS-DCM($k$)[Space($n$)]'s (CS-NCM($k$)[Space($n$)]'s), for each $k \geq 1$. Let DSPACE$_\Sigma(f(n))$ (NSPACE$_\Sigma(f(n))$) denote the class of languages over the alphabet $\Sigma$ accepted by deterministic (nondeterminsitic) Turing machines with space bound $f(n)$.

In the following we only consider languages $L \subseteq \{0^{2^n} | n \geq 1\}$. Let $\widetilde{\mathcal{L}}$[CS-DCM($k$)[Space($n$)]] ($\widetilde{\mathcal{L}}$[CS-NCM($k$)[Space ($n$)]]) be the class of all languages $L$ such that

$$L \in \mathcal{L}_{\{0\}}[\text{CS-DCM}(k)[\text{Space}(n)]] \ \ (L \in \mathcal{L}_{\{0\}}[\text{CS-NCM}(k)[\text{Space}(n)]])$$

and $L \subseteq \{0^{2^n} | n \geq 1\}$. Let D$\widetilde{\text{SPACE}}$($\log n$) (N$\widetilde{\text{SPACE}}$($\log n$)) be the class of all languages $L$ such that

$$L \in \text{DSPACE}_{\{0\}}(\log n) \ \ (L \in \text{NSPACE}_{\{0\}}(\log n))$$

and $L \subseteq \{0^{2^n} | n \geq 1\}$.

For each $L \subseteq \{0^{2^n} | n \geq 1\}$ and $j \geq 1$, let

$$T_j(L) = \{0^{2^{j \cdot n}} | n \geq 1 \ \& \ 0^{2^n} \in L\}.$$

In all the proofs there is no difference at all between the deterministic and the nondeterministic cases. Therefore we will always consider only the deterministic case.

## 3 Result

**Lemma 1.** For each $k \geq 1$,

(1) $\widetilde{\mathcal{L}}$[CS-DCM($k$)[Space($n$)]]$\subsetneq\widetilde{\text{DSPACE}}(\log n)$, and

(2) $\widetilde{\mathcal{L}}$[CS-NCM($k$)[Space($n$)]]$\subsetneq\widetilde{\text{NSPACE}}(\log n)$.

**Proof :** It was shown in [1] (Lemma 1) that for all $k \geq 1$, $\widetilde{\mathcal{L}}$[DHA($k$)]$\subsetneq\widetilde{\text{DSPACE}}(\log n)$ and $\widetilde{\mathcal{L}}$[NHA($k$)]$\subsetneq$ $\widetilde{\text{NSPACE}}$ $(\log n)$, where $\widetilde{\mathcal{L}}$[DHA($k$)] ($\widetilde{\mathcal{L}}$[NHA($k$)]) is the class of all languages $L$ such that $L \subseteq \{0^{2^n} | n \geq 1\}$ is accepted by a two-way deterministic (nondeterministic) $k$-head finite automaton. On the other hand, one can easily show that for each $k \geq 1$, $\widetilde{\mathcal{L}}$[CS-DCM($k$)[Space($n$)]]$\subseteq$ $\widetilde{\mathcal{L}}$[DHA($2k+1$)] and $\widetilde{\mathcal{L}}$[CS-NCM($k$)[Space($n$)]]$\subseteq$ $\widetilde{\mathcal{L}}$[NHA($2k+1$)]. From those facts, the lemma follows. □

**Lemma 2.** For each $L \in \widetilde{\text{DSPACE}}(\log n)$ ($\widetilde{\text{NSPACE}}(\log n)$), there exists an integer $j \geq 1$ such that $T_j(L) \in \widetilde{\mathcal{L}}$[CS-DCM(2)[Space($n$)]] ($\widetilde{\mathcal{L}}$[CS-NCM(2)[Space($n$)]]).

**Proof :** The proof is very similar to that of Lemma 2 in [1]. Let $L$ be any language in $\widetilde{\text{DSPACE}}(\log n)$, and $M$ be a deterministic Turing machine accepting $L$ within space bound $\log n$. Let $M'$ be the following modification of $M$:

(1) $M'$ writes bi($n$) on its working tape, where $n$ is the length of input tape and bi: the set of natural numbers, $\mathcal{N} \rightarrow \{0,1\}^*$, be the bijective mapping defined by: $\text{bi}(n) = \varphi \Leftrightarrow 1\varphi$ is the binary notation of $n$.

(2) During the rest of the computation $M'$ never uses its input tape again. $M'$ simulates $M$ and during this simulation its working tape is divided into 3 tracks. On the first track $M'$ stores bi($n$), on the second track the position of the input head of $M$ in binary notation and on the third track the inscription of the worktape of $M$.

Furthmore we can define $M'$ in such a way that it has only two worktape symbols. There exists some $j \geq 1$ such that $M'$ uses for every computation at most $j \cdot \log_2 n$ cells on its worktape.

We now define a deterministic CS-DCM(2) $M''$ accepting $T_j(L)$. $M''$ first tests whether the input tape is of the form $0^{2^{j \cdot n}}$, $n \geq 1$, using its two counters. If this is the case, then $M''$ simulates the action of $M'$ on the input tape $0^{2^n}$. The working tape of $M'$ is divided by the head position into two parts as described in Fig. 1, and $M''$ stores this during the simulation on its input tape by using the input heads of its two counter machines ($CM_1$ and $CM_2$) as described in Fig. 2 (with the counters empty).

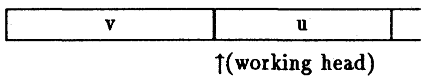


Fig. 1. The working tape of $M'$.

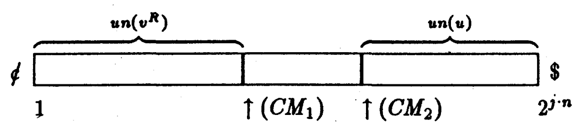


Fig. 2. The input tape of $M''$, where un: $\{0,1\}^* \rightarrow \mathcal{N}$ is the inverse mapping of bi.

In order to simulate one step of $M'$, $CM_1$ ($CM_2$) has to divide or multiply the number un($v^R$) (un($u$)) by two, and has to add $+1$ or $-1$ to the number un($v^R$) (un($u$)) (where "un" is the inverse mapping of "bi"). Moreover, $CM_1$ and $CM_2$ have to communicate with each other. All of this can be easily done with the help of their empty counters.

In order to initialize this simulation, $M''$ sets the position of the input head of $CM_1$ to $\text{un}(\text{bi}(2^n)^R) = \text{un}(\underbrace{00\cdots0}_{n}) = 2^n$ and then moves the input head of $CM_2$ to the right endmarker \$. This can be done using two counters.

It is clear that $M''$ accepts an input tape $0^{2^{j \cdot n}}$ if and only if $M'$ accepts $0^{2^n}$. □

**Lemma 3.** For each $L \subseteq \{0^{2^n} | n \geq 1\}$ and for $j, k \geq 1$:
$T_j(L) \in \widetilde{\mathcal{L}}$[CS-DCM($k$)[Space($n$)]] ($\widetilde{\mathcal{L}}$[CS-NCM($k$)[Space($n$)]])
$\Rightarrow L \in \widetilde{\mathcal{L}}$[CS-DCM($jk+1$)[Space($n$)]] ($\widetilde{\mathcal{L}}$[CS-NCM($jk+1$)[Space($n$)]]).

**Proof :** It was shown in [2] (Lemma 3.3) that $T_j(L) \in \widetilde{\mathcal{L}}$[SeDHA($k$)] ($\widetilde{\mathcal{L}}$[SeNHA($k$)]) $\Rightarrow L \in \widetilde{\mathcal{L}}$[SeDHA($jk$)] ($\widetilde{\mathcal{L}}$[SeNHA($jk$)]), where $\widetilde{\mathcal{L}}$[SeDHA($k$)] ($\widetilde{\mathcal{L}}$[SeNHA($k$)]) is the class of all languages $L$ such that $L \subseteq \{0^{2^n} | n \geq 1\}$ is accepted by a sensing two-way deterministic (nondeterministic) $k$-head finite automaton. On the other hand, one can easily show that for each $k \geq 1$,

$$\widetilde{\mathcal{L}}[\text{CS-DCM}(k)[\text{Space}(n)]]\ (\widetilde{\mathcal{L}}[\text{CS-NCM}(k)[\text{Space}(n)]]) \subseteq \widetilde{\mathcal{L}}[\text{SeDHA}(2k)]\ (\widetilde{\mathcal{L}}[\text{SeNHA}(2k)]),$$

and

$$\widetilde{\mathcal{L}}[\text{SeDHA}(k)]\ (\widetilde{\mathcal{L}}[\text{SeNHA}(k)]) \subseteq \widetilde{\mathcal{L}}[\text{CS-DCM}(k+1)[\text{Space}(n)]]\ (\widetilde{\mathcal{L}}[\text{CS-NCM}(k+1)[\text{Space}(n)]]).$$

From those facts, the lemma follows. □

**Lemma 4.** For each $L \subseteq \{0^{2^n} \mid n \geq 1\}$ and for $j > 3k$, $k \geq 1$:
$T_{j+1}(L) \in \widetilde{\mathcal{L}}[\text{CS-DCM}(k)[\text{Space}(n)]]\ (\widetilde{\mathcal{L}}[\text{CS-NCM}(k)[\text{Space}(n)]])$
$\Rightarrow T_j(L) \in \widetilde{\mathcal{L}}[\text{CS-DCM}(k+1)[\text{Space}(n)]]\ (\widetilde{\mathcal{L}}[\text{CS-NCM}(k+1)[\text{Space}(n)]])$.

**Proof :** Let $M = (CM_1, CM_2, \cdots, CM_k)$ be a CS-DCM($k$)[Space($n$)] accepting $T_{j+1}(L)$. We will construct a CS-DCM($k+1$)[Space($n$)] $M' = (CM_1', CM_2', \cdots, CM_{k+1}')$ accepting $T_j(L)$.

It can be tested easily (using 2 counters) whether the input is of the form $0^{2^{j\cdot n}}$ for some $n \geq 1$. If this is the case, then $M'$ has to test whether $0^{2^{(j+1)n}}$ is accepted by $M$. In order to do so $M'$ encodes the input head position, $h_v$, of each $CM_v$, and the counter contents, $c_v$, of each $CM_v$, where for each $1 \leq v \leq k$, $0 \leq h_v, c_v \leq 2^{(j+1)n} + 1$, by the input head position, $h_v'$, of $CM_v'$, the counter contents, $c_v'$, of $CM_v'$, and two additional numbers $\sigma_{2v}$, $\sigma_{2k+v}$, in such a form that always

$$0 \leq h_v', c_v' \leq 2^{j\cdot n} + 1, \quad 0 \leq \sigma_{2v}, \sigma_{2k+v} < 2^n,$$

and

$$h_v = h_v' + \sigma_{2v} \cdot 2^{j\cdot n}, \quad c_v = c_v' + \sigma_{2k+v} \cdot 2^{j\cdot n}.$$

Note that $h_v = 2^{(j+1)n} + 1$ iff $h_v' = 2^{j\cdot n} + 1$ and $\sigma_{2v} = 2^n - 1$, and that $h_v = h_u$ iff either ($h_v' = h_u'$ and $\sigma_{2v} = \sigma_{2u}$) or ($h_v' - h_u' = \pm 2^{j\cdot n}$ and $\sigma_{2v} - \sigma_{2u} = \mp 1$) holds.

$M'$ uses the counter of $CM_{k+1}'$ to store the $2k$ numbers $\sigma_2, \cdots, \sigma_{2k}, \sigma_{2k+1}, \cdots, \sigma_{3k}$ in the form

$$c_{k+1}' = \sigma_1 + \sigma_2 \cdot 2^n + \sigma_3 \cdot 2^{2n} + \cdots + \sigma_{3k} \cdot 2^{(3k-1)n} + 2^{(j-1)n}$$

where $c_{k+1}'$ denotes the counter contents of $CM_{k+1}'$ and $\sigma_{2i-1} = 1$ for $1 \leq i \leq k$. This is possible since $j > 3k$.

First $M'$ has to encode the initial head positions and initial counter contents of $M$. That means it has to set

$$c_{k+1}' \leftarrow 2^{0\cdot n} + 2^{2\cdot n} + \cdots + 2^{(2k-2)n} + 2^{(j-1)n}.$$

This can be done easily (using two counter machines).

During the simulation of one step of $M$, $CM_1'$, $CM_2'$, $\cdots$, $CM_k'$ can communicate with each other by means of $CM_{k+1}'$ (since the input head of $CM_{k+1}'$ is free in the computation), and every $CM_v'$ ($1 \leq v \leq k$) always stores in its finite memory which of the $\sigma_{2v}$'s, encoded by $c_{k+1}'$, are equal to $2^n - 1$ and the information whether $\sigma_{2v} = \sigma_{2u}$ or $\sigma_{2v} = \sigma_{2u} \pm 1$ holds for each $v, u \in \{1, 2, \cdots, k\}$, and $v \neq u$. Furthermore, in order to simulate the action of $CM_v$, $CM_v'$ has to distinguish two cases:

(1) $h_v' \neq 0$, $h_v' \neq 2^{j\cdot n} + 1$ (i.e., the input head of $CM_v'$ does not scan either of the endmarkers) and $0 < c_v' < 2^{j\cdot n} + 1$. (Note that $CM_v'$ can check with the help of $CM_{k+1}'$ whether or not its counter contents is equal to $2^{j\cdot n} + 1$. In this case, the input head of $CM_v$ does not scan either of the endmarkers, and $c_v > 0$. $CM_v'$ simply changes its input head position and its counter contents in the same way as $CM_v$ would do.

(2) $h_v' = 0$ or $h_v' = 2^{j\cdot n} + 1$ or $c_v' = 0$ or $c_v' = 2^{j\cdot n} + 1$. If $h_v' = 0$ (or $c_v' = 0$) and $CM_v$ would move the input head to the left (or would decrease the counter), then $CM_v'$ has to set $h_v' \leftarrow 2^{j\cdot n} - 1$ and $\sigma_{2v} \leftarrow \sigma_{2v} - 1$ (or $c_v' \leftarrow 2^{j\cdot n} - 1$ and $\sigma_{2k+v} \leftarrow \sigma_{2k+v} - 1$). If $h_v' = 2^{j\cdot n} + 1$ (or $c_v' = 2^{j\cdot n} + 1$) and $CM_v$ would move the input head to the right (or would increase the counter), then $CM_v'$ has to set $h_v' \leftarrow 2$ and $\sigma_{2v} \leftarrow \sigma_{2v} + 1$ (or $c_v' \leftarrow 2$ and $\sigma_{2k+v} \leftarrow \sigma_{2k+v} + 1$). Otherwise, $CM_v'$ simply changes its input head position and its counter contents in the same way as $CM_v$ would do.

In the simulation, performing the operation on $\sigma_{2v}$ (or $\sigma_{2k+v}$) is the difficulty in this proof, and the other operations may be easily done (with the help of $CM_{k+1}'$). In the following we will show how $M'$ can perform the operation on $\sigma_{2v}$'s (or $\sigma_{2k+v}$'s).

$M'$ can perform the operation $\pm 1$ on $\sigma_{2v}$ (or $\sigma_{2k+v}$) and test whether the new $\sigma_{2v}$ (or $\sigma_{2k+v}$) is equal to $2^n - 1$, using the algorithms described in 1 and 2 in the proof of Lemma 4 in [1]. Below, we refer to these algorithms as Algorithm 1 and Algorithm 2. For the sake of completeness, we recall them here, pointing out the necessary changes.

We will denote the counter contents of $CM_{k+1}'$ by $\lambda$ as follows:

$$\lambda = \sum_{\mu=1}^{j-1} \sigma_\mu \cdot 2^{(\mu-1)n} + 2^{(j-1)n}$$

with $\sigma_{2i-1} = 1$ for $1 \le i \le k$, and $\sigma_{3k+1} = \sigma_{3k+2} = \cdots = \sigma_{j-1} = 0$.

Furthermore we can assume that $c'_v < 2^{j\cdot n}$. (Note that $M'$ can test whether $c'_v < 2^{j\cdot n}$ by moving the input head two cells to the right.) If $c'_v \ge 2^{j\cdot n}$ then we set $c'_v = 2^{j\cdot n} - 1$ and store the difference in the finite control of $CM'_v$. We decompose $c'_v$ in the form

$$c'_v = \psi_{v1} + \psi_{v2} \cdot 2^n$$

with $0 \le \psi_{v1} < 2^n$, $0 \le \psi_{v2} < 2^{(j-1)n}$.

*Algorithm 1*: Now suppose that $h'_v = 2^{j\cdot n} + 1$ and $CM_v$ would move the input head to the right. (The cases $h'_v = 0$, $c'_v = 0$ and $c'_v = 2^{j\cdot n} + 1$ will lead to analogous considerations.) Note that the input head of $CM'_v$ does not store anything and therefore it is free for intermediate computations.

$CM'_v$ and $CM'_{k+1}$ change $c'_v = \psi_{v1} + \psi_{v2} \cdot 2^n$, $0 \le \psi_{v1} < 2^n$, $0 \le \psi_{v2} < 2^{(j-1)n}$, and $\lambda = \sum_{\mu=1}^{j-1} \sigma_\mu \cdot 2^{(\mu-1)n} + 2^{(j-1)n}$, $\sigma_{2i-1} = 1$ for $1 \le i \le k$, $\sigma_{3k+1} = \sigma_{3k+2} = \cdots = \sigma_{j-1} = 0$, into

$$c'_v = \psi_{v2} + 2^{(j-1)n},$$

$$\lambda = R_n(\psi_{v1}) + \sigma_1 \cdot 2^n + \cdots + \sigma_{j-1} \cdot 2^{(j-1)n},$$

where for any $x < 2^n$, $R_n(x)$ is defined in the following way:

Let $\varphi_n(x) \in \{0,1\}^*$, $|\varphi_n(x)| = n$, be the binary notation of $x$ lengthened by an appropriate number of leading zeros. Then $R_n(x) < 2^n$ is that number whose binary notation of length $n$ (again allowing leading zeros) is the reversal of $\varphi_n(x)$. Note that $R_n(R_n(x)) = x$ for all $x < 2^n$.

$CM'_v$ and $CM'_{k+1}$ can do the above by executing the following:

$c'_v \leftarrow c'_v + 2^{j\cdot n}$,
*While*$_1$ $\lambda < 2^{j\cdot n}$ *Do*$_1$
*Begin*$_1$
  $c'_v \leftarrow \left[\frac{c'_v}{2}\right]$, and $\alpha \leftarrow c'_v - 2\left[\frac{c'_v}{2}\right]$,
  $\lambda \leftarrow \alpha + 2\lambda$,
*End*$_1$
$\lambda \leftarrow \lambda - 2^{j\cdot n}$.

It is clear that $CM'_v$ and $CM'_{k+1}$ can perform this computation, since their input heads are free now. The loop (*Begin*$_1$, $\cdots$, *End*$_1$) is carried out exactly $n$ times and this leads to the counter contents $c'_v$ and $\lambda$ which we wanted. Note that $2^{j\cdot n}$ is given by the position of the right endmarker, and during the computation, if $c'_v, \lambda \ge 2^{j\cdot n} + 1$, we can, in fact, store $\left[\frac{c'_v}{2^{j\cdot n}+1}\right]$, $\left[\frac{\lambda}{2^{j\cdot n}+1}\right]$ in the finite controls of $CM'_v$ and $CM'_{k+1}$, and the residues $c'_v - (2^{j\cdot n}+1)\left[\frac{c'_v}{2^{j\cdot n}+1}\right]$, $\lambda - (2^{j\cdot n}+1)\left[\frac{\lambda}{2^{j\cdot n}+1}\right]$ in the counters of $CM'_v$ and $CM'_{k+1}$, respectively.

*Algorithm 2*: $CM'_v$ and $CM'_{k+1}$ change $c'_v$ and $\lambda$ into

$$c'_v = \sigma_{j-1} + \psi_{v2} \cdot 2^n,$$

$$\lambda = R_n(\psi_{v1}) + \sigma_1 \cdot 2^n + \cdots + \sigma_{j-2} \cdot 2^{(j-2)n} + 2^{(j-1)n}.$$

Let $Bit_m(x)$ denote the $m$-th least significant bit of the binary notation of $x$ with length $\ge m$ (allowing leading zeros). $CM'_{k+1}$ first changes $\lambda$ into

$$\lambda = R'_n(\psi_{v1}) + \sigma_1 \cdot 2^n + \cdots + \sigma_{j-2} \cdot 2^{(j-2)n} + 2^{(j-1)n},$$

where

$$R'_n(\psi_{v1}) = \begin{cases} R_n(\psi_{v1}) & \text{if } Bit_1(\psi_{v1}) = 1, \\ R_n(\psi_{v1}) + 1 & \text{otherwise.} \end{cases}$$

That is, only the lowest bit of $\lambda$ is changed in such a way that $\lambda$ becomes an odd number. (It is stored in the finite control whether $R_n(\psi_{v1})$ is odd or even.)

Afterwards $CM'_v$ and $CM'_{k+1}$ execute the following:

*While*$_2$ $c'_v < 2^{j\cdot n}$ *Do*$_2$
*Begin*$_2$
  *If*$_1$ $2\lambda \ge 2^{j\cdot n}$
  *then*$_1$
  $\lambda \leftarrow 2\lambda - 2^{j\cdot n}$, $c'_v \leftarrow 1 + 2c'_v$,

*else*$_1$
$\lambda \leftarrow 2\lambda$, $c'_v \leftarrow 2c'_v$,
*Endif*$_1$
*End*$_2$

In order to see what is done by the above execution, we consider the decompsition

$$\lambda = \lambda' + \alpha \cdot 2^{j \cdot n - 1}, \quad \lambda' < 2^{j \cdot n - 1}.$$

Then $\alpha = 1$ if and only if $2\lambda \geq 2^{j \cdot n}$. Note that $Bit_{j \cdot n}(\lambda) = \alpha$. As in *Algorithm 1* it can be seen that the loop (*Begin*$_2$, $\cdots$, *End*$_2$) is carried out exactly $n$ times and the number $\sigma_{j-1}$ is carried over from $\lambda$ to $c'_v$ bit by bit. Therefore, $c'_v$ and $\lambda$ are changed by this execution into

$$c'_v = \sigma_{j-1} + \psi_{v2} \cdot 2^n + 2^{j \cdot n},$$
$$\lambda = R'_n(\psi_{v1})2^n + \sigma_1 \cdot 2^{2n} + \cdots + \sigma_{j-2} \cdot 2^{(j-1)n}.$$

Then, we obtain the counter contents which we wanted by:

(a) subtracting $2^{j \cdot n}$ from $c'_v$,

(b) adding $2^{j \cdot n}$ to $\lambda$,

(c) dividing $\lambda$ by 2 as long as the remainder is 0,

(d) changing $R'_n(\psi_{v1})$ into $R_n(\psi_{v1})$.

Instead of

$$c'_v = \psi_{v1} + \psi_{v2} \cdot 2^n, \quad \lambda = \sum_{\mu=1}^{j-1} \sigma_\mu \cdot 2^{(\mu-1)n} + 2^{(j-1)n},$$

we write

$$(c'_v; \lambda) = (\psi_{v1}; \sigma_1, \cdots, \sigma_{j-1}).$$

The application of *Algorithms 1, 2* induces the transition

$$(\psi_{v1}; \sigma_1, \cdots, \sigma_{j-1}) \rightarrow (\sigma_{j-1}; R_n(\psi_{v1}), \sigma_1, \cdots, \sigma_{j-2}).$$

Therefore, by $j - 2v$ applications, we get

$$(\psi_{v1}; \sigma_1, \cdots, \sigma_{j-1}) \rightarrow (\sigma_{j-1}; R_n(\psi_{v1}), \sigma_1, \cdots, \sigma_{j-2}) \rightarrow \cdots$$
$$\rightarrow (\sigma_{2v}; R_n(\sigma_{2v+1}), \cdots, R_n(\sigma_{j-1}), R_n(\psi_{v1}), \sigma_1, \cdots, \sigma_{2v-1}).$$

Now *Algorithm 1* is applied again. Since during the computation $\sigma_{2v}$ is carried over from $c'_v$ to $\lambda$ bit by bit, $CM'_v$ and $CM'_{k+1}$ are able to add +1 and to test whether the new $\sigma_{2v}$ is equal to $2^n - 1$ (this is true if and only if all bits which are carried over are equal to one). Afterwards we apply *Algorithm 2* and get

$$(\sigma_{2v-1}; R_n(\sigma_{2v} + 1), R_n(\sigma_{2v+1}), \cdots, R_n(\sigma_{j-1}), R_n(\psi_{v1}), \sigma_1, \cdots, \sigma_{2v-2}).$$

Since $R_n(R_n(x)) = x$ for all $x < 2^n$, further applications of *Algorithms 1, 2* lead to

$$(\psi_{v1}; \sigma_1, \cdots, \sigma_{2v-1}, \sigma_{2v} + 1, \sigma_{2v+1}, \cdots, \sigma_{j-1}).$$

This shows that by the applications of *Algorithms 1,2*, $M'$ can perform the operation $\pm 1$ on $\sigma_{2v}$ (or $\sigma_{2k+v}$) and test whether the new $\sigma_{2v}$ (or $\sigma_{2k+v}$) is equal to $2^n - 1$.

Next we show how $M'$ can decide whether the new $\sigma_{2v} = \sigma_{2u}$ for each $u \in \{1, 2, \cdots, k\}$ and $v \neq u$ (when $k \geq 2$), by the application of *Algorithm 3* below. We will use $CM'_v$, $CM'_u$ and $CM'_{k+1}$. (Note that the input heads of $CM'_v$ and $CM'_{k+1}$ are free.) As in the above, we decompose $c'_v$ and $c'_u$ in the form

$$c'_v = \psi_{v1} + \psi_{v2} \cdot 2^n$$

with $0 \leq \psi_{v1} < 2^n$, $0 \leq \psi_{v2} < 2^{(j-1)n}$, and

$$c'_u = \psi_{u1} + \psi_{u2} \cdot 2^n$$

with $0 \leq \psi_{u1} < 2^n$, $0 \leq \psi_{u2} < 2^{(j-1)n}$. The counter contents of $CM'_{k+1}$ is denoted by $\lambda$ as follows:

$$\lambda = \sum_{\mu=1}^{j-1} \sigma_\mu \cdot 2^{(\mu-1)n} + 2^{(j-1)n}$$

with $\sigma_{2i-1} = 1$ for $1 \le i \le k$, and $\sigma_{3k+1} = \sigma_{3k+2} = \cdots = \sigma_{j-1} = 0$.

Algorithm 3: We first apply *Algorithms 1, 2* to $(c'_v; \lambda) = (\psi_{v1}; \sigma_1, \cdots, \sigma_{j-1})$ by $(j-2v)$ times to get

$(\psi_{v1}; \sigma_1, \cdots, \sigma_{j-1})$
$\Rightarrow (\sigma_{2v}; R_n(\sigma_{2v+1}), \cdots, R_n(\sigma_{j-1}), R_n(\psi_{v1}), \sigma_1, \cdots, \sigma_{2v-1})$.

After that, we apply *Algorithms 1, 2* to $(c'_u; \lambda) = (\psi_{u1}; R_n(\sigma_{2v+1}), \cdots, R_n(\sigma_{j-1}), R_n(\psi_{v1}), \sigma_1, \cdots, \sigma_{2v-1})$ by $2(v-u)$ times to get

$(\psi_{u1}; R_n(\sigma_{2v+1}), \cdots, R_n(\sigma_{j-1}), R_n(\psi_{v1}), \sigma_1, \cdots, \sigma_{2v-1})$
$\Rightarrow (\sigma_{2u}; R_n(\sigma_{2u+1}), \cdots, R_n(\sigma_{2v-1}), R_n(\psi_{u1}),$
$R_n(\sigma_{2v+1}), \cdots, R_n(\sigma_{j-1}), R_n(\psi_{v1}), \sigma_1, \cdots, \sigma_{2u-1})$,

where we suppose, without loss of generality, that $1 \le u < v \le j$.

Now $CM'_v$, $CM'_u$ and $CM'_{k+1}$ compare $\sigma_{2v}$ with $\sigma_{2u}$ from $Bit_1(\sigma_{2v})$ to $Bit_{\lceil \frac{n}{2} \rceil}(\sigma_{2v})$ by executing the following computation. Note that in the following computation, $Bit_2(\lambda)$ (or its reverse) is used as an identifier for distinguishing the bits of $\sigma_{2v}$ ($\sigma_{2u}$) from $\lambda$ when the bits of $\sigma_{2v}$ ($\sigma_{2u}$) are carried over to $\lambda$ by using a rotation technique.

$\alpha \leftarrow Bit_2(\lambda)$,
$\beta = \gamma \leftarrow 0$,
*While*$_3$ $\lambda < 2^{j \cdot n}$ *and* $\beta = \gamma$ *Do*$_3$
*Begin*$_3$
  $\beta \leftarrow Bit_1(c'_v)$,
  $\gamma \leftarrow Bit_1(c'_u)$,
  *If*$_2$ $\beta = \gamma$
  *then*$_2$
    $c'_v \leftarrow \left[\frac{c'_v}{2}\right]$,
    $c'_u \leftarrow \left[\frac{c'_u}{2}\right]$,
    *If*$_3$ $4\lambda < 2^{j \cdot n}$
    *then*$_3$
      $\lambda \leftarrow \bar{\alpha} + 2\lambda$,
      $\lambda \leftarrow \beta + 2\lambda$,
    *else*$_3$
      *If*$_3$ $2\lambda < 2^{j \cdot n}$
      *then*$_4$
        $T \leftarrow 0$,
      *else*$_4$
        $T \leftarrow 1$,
      *Endif*$_4$,
      $\lambda \leftarrow \alpha + 2\lambda$,
      $\lambda \leftarrow \beta + 2\lambda$,
    *Endif*$_3$
  *Endif*$_2$
*End*$_3$,

where $\bar{\alpha}$ denotes the reverse of $\alpha$, that is, if $\alpha = 1$ then $\bar{\alpha} = 0$ else $\bar{\alpha} = 1$.

Note that in the above (*While*$_3$ $\cdots$ *Do*$_3$) only one of control conditions ($\lambda < 2^{j \cdot n}$) and ($\beta = \gamma$) is changed. It is clear that after (*While*$_3$ $\cdots$ *Do*$_3$) is carried out, if $\beta \neq \gamma$, then this means $\sigma_{2v} \neq \sigma_{2u}$ (so $M'$ will restore the used counter machines, respectively, as jsut before (*While*$_3$ $\cdots$ *Do*$_3$) is done, by the application of procedure *Replace* below), and otherwise (i.e., if $\beta = \gamma$), the loop (*Begin*$_3$ $\cdots$ *End*$_3$) is carried out exactly $\lceil \frac{n}{2} \rceil$ times, and this leads to

$$\lambda = R_n(\sigma_{2u+1}) \cdot 2^{2\lceil \frac{n}{2} \rceil} + \cdots + \sigma_{2u-1} \cdot 2^{(j-2)n} \cdot 2^{2\lceil \frac{n}{2} \rceil} + 2^{(j-1)n} \cdot 2^{2\lceil \frac{n}{2} \rceil}.$$

Note that $\sigma_{2u-1} = 1$ and $2^{(j-1)n} \cdot 2^{2\lceil \frac{n}{2} \rceil} = (T+1)2^{j \cdot n}$. $CM'_v$, $CM'_u$ and $CM'_{k+1}$ then execute the following and compare $\sigma_{2v}$ with $\sigma_{2u}$ from $Bit_{\lceil \frac{n}{2} \rceil + 1}(\sigma_{2v})$ to $Bit_n(\sigma_{2v})$ if necessary.

*If*$_5$ $\beta \neq \gamma$
*then*$_5$
  $Replace(\lambda, \kappa_1, \kappa_2; \alpha)$,
*else*$_5$
  $\lambda \leftarrow \lambda - (T+1)2^{j \cdot n}$,
  *While*$_4$ $\lambda < 2^{j \cdot n}$ *and* $\beta = \gamma$ *Do*$_4$
  *Begin*$_4$
    $\beta \leftarrow Bit_1(c'_v)$,
    $\gamma \leftarrow Bit_1(c'_u)$,
    *If*$_6$ $\beta = \gamma$
    *then*$_6$
      $c'_v \leftarrow \left[\frac{c'_v}{2}\right]$,
      $c'_u \leftarrow \left[\frac{c'_u}{2}\right]$,
      $\lambda \leftarrow \bar{\alpha} + 2\lambda$,
      $\lambda \leftarrow \beta + 2\lambda$,
    *Endif*$_6$
  *End*$_4$,
  $Replace(\lambda, c'_v, c'_u; \alpha)$,
  $\lambda \leftarrow \lambda + (T+1)2^{j \cdot n}$,
  $\pi \leftarrow Bit_1(\lambda)$,
  $c'_v \leftarrow \pi + 2c'_v$,
  $c'_u \leftarrow \pi + 2c'_u$,
  $\lambda \leftarrow \left[\frac{\lambda}{2^2}\right]$,
  $Replace(\lambda, \kappa_1, \kappa_2; \alpha)$,
*Endif*$_5$

*Procedure* $Replace(\lambda, c'_v, c'_u; \alpha)$ :
  *While* $Bit_2(\lambda) \neq \alpha$ *Do*
  *Begin*

$\pi \leftarrow Bit_1(\lambda),\ c'_v \leftarrow \pi + 2c'_v,\ c'_u \leftarrow \pi + 2c'_u,\ \lambda \leftarrow \left[\frac{\lambda}{2^2}\right],$

*End*

*Endprocedure*

It will be easily seen that the loop ($Begin_4 \cdots End_4$) is carried out at most $\lfloor \frac{n}{2} \rfloor$ times, and that during the above computation, the number $\sigma_{2v}$ ($\sigma_{2u}$) is carried over to $\lambda$ bit by bit as long as $\beta = \gamma$ holds. (See Fig. 3 and Fig. 4.)

| $\beta_m$ | $\bar{\alpha}$ | ... | $\beta_1$ | $\bar{\alpha}$ | 0/1 | $\alpha$ | ...... |
|---|---|---|---|---|---|---|---|

$$\underbrace{bit_1\ \ bit_2\ \ \cdots}_{<2\cdot\lceil\frac{n}{2}\rceil bits}$$

Fig. 3. The binary notation of $\lambda$ after ($While_3 \cdots Do_3$) ends with $\beta \neq \gamma$, where for each $1 \leq i \leq m$, $\beta_i = Bit_i(\sigma_{2v}) = Bit_i(\sigma_{2u})$ (but $Bit_{m+1}(\sigma_{2v}) \neq Bit_{m+1}(\sigma_{2u})$).

| $\beta_m$ | $\bar{\alpha}$ | ... | $\beta_{s+1}$ | $\bar{\alpha}$ | $\beta_s$ | $\alpha$ | $\beta_{s-1}$ | $\bar{\alpha}$ | ... | $\beta_1$ | $\bar{\alpha}$ | 0/1 | $\alpha$ | ...... |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

$$\underbrace{bit_1\ \ bit_2\ \ \cdots}_{\leq 2\cdot\lfloor\frac{n}{2}\rfloor bits} \qquad \underbrace{\qquad\qquad\qquad}_{2\cdot\lceil\frac{n}{2}\rceil bits}$$

Fig. 4. The binary notation of $\lambda$ after ($While_4 \cdots Do_4$) is carried out, where $s = \lceil\frac{n}{2}\rceil$ and if $m = n$ (i.e., $\beta = \gamma$) then $\beta_n\beta_{n-1}\cdots\beta_1$ is the binary representation of $\sigma_{2v}$ ($= \sigma_{2u}$).

Now $M'$ has finished the comparison between $\sigma_{2v}$ and $\sigma_{2u}$ (if $\beta = \gamma$, then $\sigma_{2v} = \sigma_{2u}$, and otherwise $\sigma_{2v} \neq \sigma_{2u}$) and restored the used counter contents as just before ($While_3 \cdots Do_3$). Then, further applications of *Algorithms 1, 2* lead to

$$(c'_u; \lambda) = (\psi_{u1}; R_n(\sigma_{2v+1}), \cdots, R_n(\sigma_{j-1}), R_n(\psi_{v1}), \sigma_1, \cdots, \sigma_{2v-1})$$

and

$$(c'_v; \lambda) = (\psi_{v1}; \sigma_1, \cdots, \sigma_{j-1}).$$

This shows that $M'$ is able to simulate $M$ step by step. □

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

(1) $\mathcal{L}_\Sigma$[CS-DCM($k$)[Space($n$)]]$\subsetneq\mathcal{L}_\Sigma$[CS-DCM($k+1$)[Space($n$)]],

(2) $\mathcal{L}_\Sigma$[CS-NCM($k$)[Space($n$)]]$\subsetneq\mathcal{L}_\Sigma$[CS-NCM($k+1$)[Space($n$)]],

even if the alphabet $\Sigma$ is restricted to a one-letter.

**Proof :** Suppose there exists some $k \geq 1$ such that $\mathcal{L}_{\{0\}}$[CS-DCM($k$)[Space($n$)]]$=\mathcal{L}_{\{0\}}$[CS-DCM($k+1$)[Space($n$)]]. This implies $\widetilde{\mathcal{L}}$[CS-DCM($k$)[Space($n$)]]$=\widetilde{\mathcal{L}}$[CS-DCM($k+1$)[Space($n$)]], and therefore the following is true:

$\forall L \in \widetilde{\text{DSPACE}}(\log n)$

$\Rightarrow \exists j(> 3k), T_j(L) \in \widetilde{\mathcal{L}}$[CS-DCM(2)[Space($n$)]] (Lemma 2)

$\Rightarrow T_j(L) \in \widetilde{\mathcal{L}}$[CS-DCM($k+1$)[Space($n$)]] $=\widetilde{\mathcal{L}}$[CS-DCM($k$)[Space($n$)]]

$\Rightarrow T_{j-1}(L) \in \widetilde{\mathcal{L}}$[CS-DCM($k+1$)[Space($n$)]] $=\widetilde{\mathcal{L}}$[CS-DCM($k$)[Space($n$)]] (Lemma 4)

$\Rightarrow \cdots$

$\Rightarrow T_{3k+1}(L) \in \widetilde{\mathcal{L}}$[CS-DCM($k+1$)[Space($n$)]] $=\widetilde{\mathcal{L}}$[CS-DCM($k$)[Space($n$)]] (Lemma 4)

$\Rightarrow L \in \widetilde{\mathcal{L}}$[CS-DCM($k(3k+1)+1$)[Space($n$)]] (Lemma 3)

Therefore $\mathcal{L}_{\{0\}}$[CS-DCM($k$)[Space($n$)]]$=\mathcal{L}_{\{0\}}$[CS-DCM($k+1$)[Space($n$)]] implies $\widetilde{\text{DSPACE}}$ ($\log n$) $\subseteq \widetilde{\mathcal{L}}$[CS-DCM($k(3k+1)+1$)[Space($n$)]], which is a contradiction to Lemma 1. □

**REFERENCES**


1. B. Monien, "Two-way multihead automata over a one-letter alphabet", *RAIRO Inform. Theor.* 14, (1980) 67–82.
2. Y. Wang, K. Inoue and I. Takanami, Some hierarchy results on multihead automata over a one-letter alphabet, *IEICE TRANS. INF. & SYST.*, Vol. E76-D, No.6, (1993)625–633.
3. Y. Wang, K. Inoue and I. Takanami, Cooperating systems of one-Way counter machines (in Japanese), *RIMS Kokyuroku 790*, (Kyoto University, June 1992)8–14.