Sufficient spaces for seven-way four-dimensional Turing machines to simulate four-dimensional one-marker automata

Makoto Nagatomo, Makoto Sakamoto, Hikaru Susaki, Tuo Zhang, Satoshi Ikeda and Hiroshi Furutani
Faculty of Engineering, University of Miyazaki, 1-1 Gakuen Kibanadai Nishi
Miyazaki, Miyazaki 889-2192, Japan
E-mail: sakamoto@cs.miyazaki-u.ac.jp

Takao Ito
Institute of Engineering, Hiroshima University, 4-1, Kagamiyama1-chome
Higashi-Hiroshima, Hiroshima 739-8527, Japan
E-mail: itotakao@horoshima-u.ac.jp

Yasuo Uchida
Department of Business Administration, Ube National College of Technology, Tokiwadai
Ube, Yamaguchi 755-8555, Japan
E-mail:uchida@ube-k.ac.jp

Tsunehiro Yoshinaga
Department of Computer Science & Electronic Engineering,
National Institute of Technology, Tokuyama College, Gakuendai
Shunan, Yamaguchi 745-8585, Japan
E-mail:yosinaga@tokuyama.ac.jp

Abstract

A multi-marker automaton is a finite automaton which keeps marks as 'pebbles' in the finite control, and cannot rewrite any input symbols but can make marks on its input with the restriction that only a bounded number of these marks can exist at any given time. An improvement of picture recognizability of the finite automaton is the reason why the marker automaton was introduced. That is, a one-marker automaton can recognize connected picture. This automaton has been investigated in the two- or three-dimensional case. As is well known among the researchers of automata theory, one-marker automata are equivalent to ordinary finite state automata. In other words, there is no working space usage such as Turing machines to calculate the space complexities. In this paper, we deal with four-dimensional one-marker automata in terms of the space complexities that seven-way four-dimensional Turing machines, which can move east, west, south, north, up, down, or in the future, but not in the past on four-dimensional rectangular input tapes, suffice to simulate one-marker automata.

Keywords: computational complexity, finite automaton, marker, simulation, Turing machine, upper bounds.

1. Introduction

A multi-marker automaton is a finite automaton which keeps marks as 'pebbles' in the finite control, and cannot rewrite any input symbols but can make marks on its input with the restriction that only a bounded number of these marks can exist at any given time[1]. An improvement of picture recognizability of the finite automaton is the reason why the marker automaton was introduced. That is, a two-dimensional one-marker automaton can recognize connected pictures. This automaton has been widely investigated in the two- or three-dimensional case [2].

As is well known among the researchers of automata theory, one-dimensional one-marker automata are equivalent to ordinary finite state automata. In other words, there is no need of working space usage for

one-way Turing machines to simulate one-marker automata, as well as finite state automata.

In the two-dimensional case, the following facts are known : the necessary and sufficient space for three-way two-dimensional deterministic Turing machines *TR*2-*DTM*'s to simulate two-dimensional deterministic (nondeterministic) finite automata 2-*DFA*'s (2-*NFA*'s) is *m*log*m* (m^2) and the corresponding space for three-way two-dimensional nondeterministic Turing machines *TR*2-*NTM*'s is *m* (*m*), whereas the necessary and sufficient space for three-way two-dimensional deterministic Turing machines *TR*2-*DTM*'s to simulate two-dimensional deterministic (nondeterministic) one-marker automata 2-DMA_1's (2-NMA_1's) is $2^{m\log m}$ (2^{m^2}) and the corresponding space for *TR*2-*NTM*'s is *m*log*m* (m^2), where *m* is the number of columns of two-dimensional rectangular input tapes.

In the three-dimensional case, the following facts are known : the necessary and sufficient space for five-way three-dimensional deterministic Turing machines *FV3-DTM*'s to simulate three-dimensional deterministic (nondeterministic) finite automata 3-*DFA*'s (3-*NFA*'s) is m^2log*m* (m^3) and the corresponding space for five-way three-dimensional nondeterministic Turing machines *FV3-NTM*'s is m^2 (m^2), whereas the necessary and sufficient space for five-way three-dimensional deterministic Turing machines *FV3-DTM*'s to simulate three-dimensional deterministic (nondeterministic) one-marker automata 3-DMA_1's (3-NMA_1's) is $2^{lm\log lm}$ ($2^{l^2m^2}$) and the corresponding space for *FV3-NTM*'s is *lm*log*lm* (l^2m^2), where *l*(*m*) is the number of rows (columns) on each plane of three-dimensional rectangular input tapes.

In this paper, we deal with four-dimensional one-marker automata in terms of the space complexities that seven-way four-dimensional Turing machines suffice to simulate four-dimensional one-marker automata.

2. Preliminaries

An ordinary finite automaton cannot rewrite any symbols on input tape, but a marker automaton can make a mark on the input tape. We can think of the mark as a 'pebble' that *M* puts down in a specified position. If *M* has already put down the mark, and wants to put it down elsewhere, *M* must first go to the position of the mark and pick it up. Formally, we define it as follows.

Definition 2.1. A *four-dimensional nondeterministic one-marker automaton* (4-NMA_1) is defined by the 6-tuple $M = (Q, q_0, F, \Sigma, \{+, -\}, \delta)$, where
(1) Q is a finite set of *states* ;
(2) $q_0 \in Q$ is the *initial state* ;
(3) $F \subseteq Q$ is the set of *accepting states* ;
(4) Σ is a finite input *alphabet* ($\# \notin \Sigma$ is the *boundary symbol*);
(5) $\{+,-\}$ is the pair of *signs of presence and absence of the marker* ; and
(6) $\delta : (Q \times \{+,-\}) \times ((\Sigma \cup \{\#\}) \times \{+,-\}) \mapsto \times 2^{Q\times\{+,-\}} \times ((\Sigma \cup \{\#\}) \times \{+,-\}) \times \{\text{east}, \text{west}, \text{south}, \text{north}, \text{up}, \text{down}, \text{future}, \text{past}, \text{no move}\}$ is the *next-move function*, satisfying the following : For any q, $q' \in Q$, any a, $a' \in \Sigma$, any u, u', v, $v' \in \{+,-\}$, and any $d \in \{\text{east}, \text{west}, \text{south}, \text{north}, \text{up}, \text{down}, \text{future}, \text{past}, \text{no move}\}$, if $((q', u'), (a', v'), d) \in \delta((q,v), (a,v))$ then $a = a'$ and $(u, v, u', v') \in \{(+,-,+,-), (+,-,-,+), (-,+,-,+), (-,+,+,-), (-,-,-,-)\}$.

We call a pair (q, u) in $Q \times \{+,-\}$ an *extended state*, representing the situation that *M* holds or does not hold the marker in the finite control according to the sign $u = +$ or $u = -$, respectively. A pair (a, v) in $\Sigma \times \{+,-\}$ represents an input tape cell on which the marker exists or does not exist according to the sign $u = +$ or $u = -$, respectively.

Therefore, the restrictions on δ imply the following conditions. (i) When holding the marker, *M* can put it down or keep on holding. (ii) When not holding the marker, and ① if the marker exists on the current cell, *M* can pick it up or leave it there, or ② if the marker does not exist on the current cell, *M* cannot create a new marker any more.

Definition 2.2. Let Σ be the input alphabet of 4-NMA_1 *M*. An *extended input tape* $\tilde{x}$ of *M* is any four-dimensional tape over $\Sigma \times \{+,-\}$ such that for some $x \in \Sigma^{(4)}$,
(i) for each $j(1 \le j \le 4)$, $l_j(\tilde{x}) = l_j(x)$,
(ii) for each $i_1\big(1 \le i_1 \le l_1(\tilde{x})\big)$, $i_2\big(1 \le i_2 \le l_2(\tilde{x})\big)$, $i_3\big(1 \le i_3 \le l_3(\tilde{x})\big)$, and $i_4\big(1 \le i_4 \le l_4(\tilde{x})\big)$, $\tilde{x}(i_1, i_2, i_3, i_4) = x\big((i_1, i_2, i_3, i_4), u\big)$ for some $u \in \{+,-\}$.

Definition 2.3. A *configuration* of 4-NMA_1 $M = (Q, q_0, F, \Sigma, \{+, -\}, \delta)$ is a pair of an element of $\left((\Sigma \cup \{\#\}) \times \{+,-\}\right)^{(4)}$ and an element of $C_M = (\mathbf{N} \cup \{0\})^{(4)} \times (Q \times \{+,-\})$. The first component of a configuration $c = (\tilde{x}, ((i_1, i_2, i_3, i_4), (q, u)))$ represents the extended input tape of M. The second component (i_1, i_2, i_3, i_4) of c represents the input head position. The third component (q, u) of c represents the extended state. An element of C_M is called a *semi-configuration* of M. If q is the state associated with configuration c, then c is said to be an *accepting configuration* if q is an accepting state. The *initial configuration* of M on input x is $I_M(x) = (x^-, ((1,1,1,1), (q_0, +)))$, where x^- is the special extended input tape of M such that $x^-(i_1, i_2, i_3, i_4) = (x(i_1, i_2, i_3, i_4), -)$ for each $i_1, i_2, i_3, i_4 \left(1 \le i_1 \le l_1(x^-), 1 \le i_2 \le l_2(x^-), 1 \le i_3 \le l_3(x^-),\ 1 \le i_4 \le l_4(x^-)\right)$. If M moves deterministically, we call M a four-dimensional deterministic one-marker automaton (4-DMA_1).

Definition 2.4. Given a 4-NMA_1 $M = (Q, q_0, F, \Sigma, \{+, -\}, \delta)$, we write $c \vdash_M c'$ and say c' is a *successor* of c if configuration c' follows from configuration c in one step of M, according to the transition rules δ. $\vdash_M^*$ denotes the reflexive transitive closure of $\vdash_M$. The relation $\vdash_M$ is not necessarily single-valued, because δ is not. A *computation path* of M on x is a sequence $c_0 \vdash_M c_1 \vdash_M \ldots \vdash_M c_n (n \ge 0)$, where $c_0 = I_M(x)$. An *accepting computation path* of M on x is a computation path of M on x which ends in an accepting configuration. We say that M *accepts* x if there is an accepting computation path of M on input x.

Let $S(l, m, n, t): \mathbf{N^4} \mapsto \mathbf{R}$ be a function. A seven-way four-dimensional Turing machine M is said to be $S(l, m, n, t)$ *space-bounded* if for each $l, m, n, t \ge 1$ and for each x with $l_1(x) = l$, $l_2(x) = m$, $l_3(x) = n$, and $l_4(x) = t$, if x is accepted by M, then there is an accepting computation path of M on x in which M uses no more than $S(l, m, n, t)$ cells of the storage tape. We denote an $S(l, m, n, t)$ space-bounded *SV*4-*DTM* (*SV*4-*NTM*) by *SV*4-*DTM*($S(l, m, n, t)$) (*SV*4-*NTM* ($S(l, m, n, t)$)).

Let $L(l, m, n) : \mathbf{N^3} \mapsto \mathbf{R}$ be a function. A seven-way four-dimensional Turing machine M is said to be $L(l, m, n)$ *space-bounded* if for each l, m, n$\ge$ 1 and for each x with $l_1(x) = l$, $l_2(x) = m$, and $l_3(x) = n$, if x is accepted by M, then there is an accepting computation path of M on x in which M uses no more than $L(l, m, n)$ cells of the storage tape. We denote an $L(l, m, n)$ space-bounded *SV*4-*DTM* (*SV*4-*NTM*) by *SV*4-*DTM*($L(l, m, n)$) (*SV*4-*NTM*($L(l, m, n)$)).

Definition 2.5. For any four-dimensional automaton M with input alphabet Σ, define T(M) = $\{x \in \Sigma^{(4)} \mid M$ accepts $x\}$. Furthermore, for each $X \in \{D,N\}$, define

$\mathcal{L}$[4-XMA_1] = $\{T \mid T = T(M)$ for some 4-$XMA_1\}$,

$\mathcal{L}$[*SV*4-*XTM*(*S*(*l*,*m*,*n*))] = $\{T \mid T = T(M)$ for some *SV*4-*XTM*(*S*(*l*,*m*,*n*)) $M\}$, and

$\mathcal{L}$[*SV*4-*XTM*(*L*(*l*,*m*))] = $\{T \mid T = T(M)$ for some *SV*4-*XTM*((*l*,*m*)) $M\}$.

By using the same technique as in the proof of Lemma 5.4 in [2], we can easily prove the following theorem.

Lemma 2.1. For *any function* $L(l, m, n) \ge \log lmn$, $\mathcal{L}$[*SV*4-*XTM*(*L*(*l*,*m*,*n*))] $\subseteq$ $\cup_{c>0}$ $\mathcal{L}$[*SV*4-*DTM*($2^{c(L(l,m,n))}$)].

3. Sufficient spaces

In this section, we investigate the sufficient spaces (i.e., upper bounds) for seven-way Turing machines to simulate one-marker automata. We first show that *lmn* log*lmn* space is sufficient for *SV*4-*NTM*'s to simulate 4-DMA_1's.

Theorem 3.1. $\mathcal{L}$[4-DMA_1] $\subseteq$ $\mathcal{L}$[*SV*4-*NTM*(*lmn*log*lmn*)].

Proof : Suppose that a 4-DMA_1 $M = (Q, q_0, F, \Sigma, \delta)$ is given. We partition the extended states $Q \times \{+,-\}$ into disjoint subsets $Q^+ = Q \times \{+\}$ and $Q^- = Q \times \{-\}$ which correspond to the extended states when M is holding and not holding the marker in the finite control, respectively. We assume that M has a unique accepting state q_a, i.e., | F | = 1. In order to make our proof clear, we also assume that M begins to move with its input head on the position (l + 1, m + 1, n + 1, t + 1) and, when M accepts an input, it enters the accepting state at the same position (l + 1, m + 1, n + 1, t + 1) with the marker held in the finite control.

Suppose that an input tape x with $l_1(x) = l$, $l_2(x) = m$, $l_3(x) = n$, and $l_4(x) = t$ is given to M. For M and x, we define three types of mappings

$$f_r^{\uparrow -} : S^- \times \{0,1,\ldots,l+1\} \times \{0,1,\ldots,m+1\} \times \{0,1,\ldots,n+1\} \mapsto S^- \times \{0,1,\ldots,l+1\}$$

$\times \{0,1,\dots,m+1\} \times \{0,1,\dots,n+1\} \cup \{l\}$,
$f_r^{\uparrow+} : S^+ \times \{0,1,\dots,l+1\} \times \{0,1,\dots,m+1\}$
$\times \{0,1,\dots,n+1\} \mapsto S^+ \times \{0,1,\dots,l+1\}$
$\times \{0,1,\dots,m+1\} \times \{0,1,\dots,n+1\} \cup \{l\}$, and
$f_r^{\downarrow-} : S^- \times \{0,1,\dots,l+1\} \times \{0,1,\dots,m+1\}$
$\times \{0,1,\dots,n+1\} \mapsto S^- \times \{0,1,\dots,l+1\}$
$\times \{0,1,\dots,m+1\} \times \{0,1,\dots,n+1\} \cup \{l\}$,

(r = 0,1, ... , t+1) as follows. (Below, we attach the superscripts '+', '−' to any extended states in (Q^+, Q^-, respectively.)

$f_r^{\uparrow-}(q^-,i,j,k) = (q'^-,i',j',k')$: Suppose that we make M start from the configuration $(x^-,((i,j,k,r-1),q^-))$, i.e., no marker existing either on the input x or in the finite control of M. After that, if M reaches the r-th three-dimensional rectangular array of x in some time, the configuration corresponding to the first arrival is $(x^-,((i',j',k',r),q'^-))$;

$f_r^{\uparrow-}(q^-,i,j,k) = l$: Starting from the configuration $(x^-,((i,j,k,r-1),q^-))$ with no marker on the input tape, M never reaches the r-th three-dimensional rectangular array of x.

$f_r^{\uparrow+}(q^+,i,j,k) = (q'^+,i',j',k')$: Suppose that we make M start from the configuration $(x^-,((i,j,k,r-1),q^+))$, i.e., holding the marker in the finite control of M. After that, if M reaches the r-th three-dimensional rectangular array of x with its marker held in the finite control in some time (so, when M puts down the marker on the way, it must return to this position again and pick up the marker), the configuration corresponding to the first arrival is $(x^-,((i',j',k',r),q'^+))$;

$f_r^{\uparrow+}(q^+,i,j,k) = l$: Starting from the configuration $(x^-,((i,j,k,r-1),q^+))$, M never reaches the r-th three-dimensional rectangular array of x with its marker held in the finite control.

$f_r^{\downarrow-}(q^-,i,j,k) = (q'^-,i',j',k')$: Suppose that we make M start from the configuration $(x^-,((i,j,k,r+1),q^-))$, i.e., no marker existing either on the input tape or in the finite control of M. After that, if M reaches the r-th three-dimensional rectangular array of x in some time, the configuration corresponding to the first arrival is$(x^-,((i',j',k',r),q'^-))$;

$f_r^{\downarrow-}(q^-,i,j,k) = l$: Starting from the configuration $(x^-,((i,j,k,r+1),q^-))$, M never reaches the r-th three-dimensional rectangular array of x.

Then, we can show that there exists an SV4-NTM(lmloglm) M' such that $T(M')$= $T(M)$. Roughly speaking, while scanning from the top three-dimensional rectangular array down to the bottom three-dimensional rectangular array of the input, M' guesses $f_r^{\downarrow-}$, constructs $f_{r+1}^{\uparrow-}$ and $f_{r+1}^{\uparrow+}$, checks $f_{r-1}^{\downarrow-}$, and finally at the bottom three-dimensional rectangular array of the input, M' decides by using $f_{t+1}^{\uparrow-}$ and $f_{t+1}^{\uparrow+}$ whether or not M accepts x.

In order to record these mappings for each r, $O(lmn)$ blocks of $O(\log lmn)$ size suffice, so in total, $O(lmn\log lmn)$ cell of the working tape suffice. More precisely, the working tape must be used as a 'multi-track' tape, but we omit the detailed construction of the working tape of M'. It will be obvious that $T(M) = T(M')$. □

From Lemma 2.1 and Theorem 3.1, we get the following.

Corollary 3.1. $\mathcal{L}$[4-DMA_1] ⊆ $\mathcal{L}$[SV4-$DTM(2^{O(lmn\log lmn)})$].

Next, we can show that $l^2m^2n^2$ space is sufficient for SV4-NTM's to simulate 4-NMA_1's. The basic idea and outline of the proof are the same as those of Theorem 2.1.

Theorem 3.2. $\mathcal{L}$[4-NMA_1] ⊆ $\mathcal{L}$[SV4-$NTM(l^2m^2n^2)$].

From Lemma 2.1 and Theorem 3.2, we get the following.

Corollary 3.2. $\mathcal{L}$[4-NMA_1] ⊆ $\mathcal{L}$[SV4-$DTM(2^{O((l^2m^2n^2))})$].

4. Conclusion

In this paper, we showed the sufficient space for SV4-DTM's to simulate 4-DMA_1's (4-NMA_1's) is $2^{lmn\log lmn}(2^{l^2m^2n^2})$ and the sufficient space for SV4-NTM's to simulate 4-DMA_1's (4-NMA_1's) is $lmn\log lmn$ ($l^2m^2n^2$). It will be interesting to investigate how much space is necessary for SV4-DTM's (or SV4-NTM's) to simulate 4-DMA_1's (or 4-NMA_1's).

References

1. M. Blum and C. Hewitt, "Automata on a two-dimensional tape", IEEE Symposium on Switching and Automata Theory (1967), pp.155-160.
2. M. Sakamoto, "Three-dimensional alternating Turing machines", Ph.D. Thesis, Yamaguchi University (1999).