A space lower-bound technique for four-dimensional alternating Turing machines

Makoto Nagatomo, Shinnosuke Yano, Makoto Sakamoto, Satoshi Ikeda and Hiroshi Furutani
Faculty of Engineering, University of Miyazaki,
1-1 Gakuen Kibanadai Nishi, Miyazaki, Miyazaki 889-2192, Japan
E-mail: je.suis.makoto@gmail.com, shinchandx@ezweb.ne.jp, sakamoto@cs.miyazaki-u.ac.jp, bisu@cs.miyazaki-u.ac.jp, furutani@cs.miyazaki-u.ac.jp

Takao Ito, Tsutomu Ito
Institute of Engineering, Hiroshima University, 4-1, Kagamiyama 1-chome
Higashi-Hiroshima, Hiroshima 739-8527, Japan
E-mail: itotakao@horoshima-u.ac.jp, 0va71-2538f211n@ezweb.ne.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

Alternating Turing machines were introduced in 1981 as a generalization of nondeterministic Turing machines and as a mechanism to model parallel computation. On the other hand, we have no enough techniques which we can show that some concrete four-dimensional language is not accepted by any space-bounded four-dimensional alternating Turing machines. The main purpose of this paper is to present a technique which we can show that some four-dimensional language is not accepted by any space-bounded four-dimensional alternating Turing machines. Concretely speaking, we show that the set of all four-dimensional input tapes over {0,l}, which each top half part is equal to each bottom half part, is not accepted by any $L(m)$ space-bounded four-dimensional alternating Turing machines for any function $L(m)$ smaller than log m.

Keywords: Alternation, Complexity, Computation Tree, Configuration, Four-Dimension, Turing machine

1. Introduction and Preliminaries

We have no enough techniques which we can show that some concrete four-dimensional language is not accepted by any space-bounded four-dimensional alternating Turing machines [1, 2]. The main purpose of this paper is to present a technique which we can show that some four-dimensional language is not accepted by any space-bounded four-dimensional alternating Turing machines. Concretely speaking, we show that the set of all four-dimensional input tapes over {0,l}, which each top half part is equal to each bottom half part, is not accepted by any $L(m)$ space-bounded four-dimensional alternating Turing machines for any function $L(m)$ such that $\lim_{m\to\infty}[L(m)/\log m] = 0$. We let each side-length of each four-dimensional input tape of these automata be equivalent in order to increase the theoretical interest.

Let Σ be a finite set of symbols. A three-dimensional tape over Σ is a four-dimensional rectangular array of elements of Σ. The set of all four-dimensional tapes over Σ is denoted by $\Sigma^{(4)}$. Given a tape $x \in \Sigma^{(4)}$, for each integer j($1 \leq j \leq 4$), we let $m_j(x)$ be the length of x along the jth axis. The set of all $x \in \Sigma$ with $l_1(x)=m_1$, $l_2(x)=m_2$, $l_3(x)=m_3$, and $l_4(x)=m_4$ denoted by $\Sigma^{(m_1,m_2,m_3,m_4)}$. If $1 \leq i_j \leq lj(x)$ for each $j(1\leq j \leq 4)$, let $x(i_1,i_2,i_3,i_4)$ denote the symbol in x with coordinates (i_1,i_2,i_3,i_4). Furthermore, we define $x[(i_1,i_2,i_3,i_4),(i_1',i_2',i_3',i_4')]$, when $1 \leq i_j \leq i'_j \leq l_j(x)$ for each integer $j(1\leq j \leq 4)$, as the four-dimensional tape y satisfying the following (i) and (ii):

(i) for each $j(1\leq j\leq 4)$, $l_j(y)=i'_j-i_j+1$;

(ii) for each r_1, r_2, r_3, r_4($1\leq r_1\leq l_1(y)$, $1\leq r_2\leq l_2(y)$, $1\leq r_3\leq l_3(y)$, $1\leq r_4\leq l_4(y)$), $y(r_1, r_2, r_3, r_4) = x(r_1+i_1-1, r_2+i_2-1, r_3+i_3-1, r_4+i_4-1)$. (We call $x[(i_1, i_2, i_3, i_4), (i'_1, i'_2, i'_3, i'_4)]$ $x[(i_1, i_2, i_3, i_4), (i'_1, i'_2, i'_3, i'_4)]$-*segment* of x.);

A four-dimensional alternating Turing machine (4-*ATM*) M is defined by the 7-tuple M = (Q, q_0, U, F, Σ, Γ, δ), where (1) Q is a finite set of states; (2) $q_0 \in Q$ is the *initial state*; (3) $U \subseteq Q$ is the set of *universal states*; (4) $F \subseteq Q$ is the set of *accepting states*; (5) Σ is a finite *input alphabet* ($\# \notin \Sigma$ is the *boundary symbol*); (6) Γ is a finite *storage-tape alphabet* ($B \in \Gamma$ is the *boundary symbol*), and (7) $\delta \subseteq (Q \times (\Sigma \cup \{\#\}) \times \Gamma) \times (Q \times (\Gamma - \{B\}) \times$ {east, west, south, north, up, down, past, future, no move} x {right, left, no move}) is the *next-move relation*.

A state q in Q - U is said to be *existential*. The machine M has a read-only four-dimensional input tape with boundary symbols # 's and one semi-infinite *storage tape*, initially blank. Of course, M has a *finite control*, an *input head*, and a *storage-tape head*. A *position* is assigned to each cell of the read-only input tape and to each cell of the storage tape. A *step* of M consists of reading one symbol from each tape, writing a symbol on the storage tape, moving the input and storage heads in specified directions, and entering a new state, in accordance with the next-move relation δ. Note that the machine cannot write the blank symbol. If the input head falls off the input tape, or if the storage head falls off the storage tape (by moving left), then the machine M can make no further move.

A *configuration* of a 4-*ATM* M = (Q, q_0, U, F, Σ, Γ, δ) is a pair of an element of $\Sigma^{(4)}$ and an element of $C_M = (\boldsymbol{N}\cup\{0\})^3 \times S_M$, where $S_M = Q \times (\Gamma-\{B\})^* \times \boldsymbol{N}$ and $\boldsymbol{N}$ denotes the set of all the positive integers. The first component x of a configuration $c = (x, ((i_1, i_2, i_3, i_4), (q, \alpha, j)))$ represents the input to M. The second component (i_1, i_2, i_3, i_4) of c represents the input-head position. The third component (q, α, j) of c represents the state of the finite control, nonblank contents of the storage tape, and the storage-head position. An element of C_M is called a *semi-configuration* of M and an element of S_M is called a storage state of M. If q is the state associated with configuration c, then c is said to be a *universal* (*existential*, *accepting*) configuration if q is a universal (existential, accepting) state. The *initial configuration* of M on input x is $I_M(x) = (x, (1, 1, 1, 1), (q_0, \lambda, 1))$, where λ is the null string.

Given M = (Q, q_0, U, F, Σ, Γ, δ), 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. The relation $\vdash_M$ is not necessarily single-valued, because δ is not. A *computation path* of M on x is a sequence. $C_0 \vdash_M C_1 \vdash_M \cdots \vdash_M C_n$ ($n \geq 0$), where $C_0 = I_M(x)$. 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 a configuration $l(v)$, (2) If v is an internal node (a nonleaf) of the tree, $l(v)$ is universal and $\{c \mid l(v) \vdash_M c\} = \{c_1,\cdots, c_k\}$, then v has exactly k children $v_1,\cdots,v_k$ such that $l(v_i) = c_i$ ($1\leq i \leq k$), and (3) If v is an internal node of the tree and $l(v)$ is existential, then v has exactly one child u such that $l(v) \vdash_M l(u)$. 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 configurations. We say that M accepts x if there is an accepting computation tree of M on input x. Define $T(M) = \{x \in \Sigma^{(4)} \mid M \text{ accepts } x\}$.

In this paper, we shall concentrate on investigating the properties of 4-*ATM*'s whose each side-length of each four-dimensional input tape is equivalent and whose storage tapes are bounded (in length) to use.

Let $L(m) : \boldsymbol{N} \to \boldsymbol{N}$ be a function with one variable m. With each 4-*ATM* M we associate a *space complexity function* SPACE that takes configurations to natural numbers. That is, for each configuration $c = (x, ((i_1, i_2, i_3, i_4), (q, \alpha, j)))$, let SPACE$(c) = |\alpha|$. We say that M is $L(m)$ *space-bounded* if for all $m \geq 1$ and for each x with $l_1(x) = l_2(x) = l_3(x) = l_4(x) = m$, if x is accepted by M, then there is an accepting computation tree of M on input x such that for each node v of the tree, SPACE$(l(v)) \leq L(m)$. We denote an $L(m)$ space-bounded 4-*ATM* by 4-*ATM* $(L(m))$. $\mathfrak{L}$[4-*ATM* $(L(m))$] $= \{T \mid T = T(M)$ for some 4-*ATM* $(L(m))$ $M\}$.

2. Main Result

[**Theorem 1**] *Let* $T = \{x \in \{0, 1\}^{(4)} \mid {}^{\exists}m \leq 1\ [l_1(x) = l_2(x) = l_3(x) = l_4(x) = 2m\ \&\ x[(1, 1, 1, 1), (2m, 2m, 2m, m)] = x[(1, 1, 1, m+1), (2m, 2m, 2m, 2m)]]\}$. *Then*, $T \notin \mathfrak{L}$[4-*ATM* $(L(m))$] *for any* $L(m) = o(\log m)$.

(**Proof**) Suppose that there exists a 4-*ATM* $(L(m))$ M accepting T, where $L(m) = o(\log m)$. We assume, without loss of generality, that M moves a storage-tape head after changing its state and writing a new symbol on the storage tape, and moves an input head finally. For each $m \leq 1$, let $V(m) = \{x \in T \mid l_1(x) = l_2(x) = l_3(x) = l_4(x) = 2m\}$. For each x in $V(m)$, let $t(x)$ be one fixed accepting computation tree of M on x such that each node v of the tree satisfies SPACE$(l(v)) \leq L(2m)$, where for each node v of $t(x)$, $l(v)$ represents the label of v. Without loss of generality, we assume that for any $t(x)$;

(i) any two different nodes on any path of $t(x)$ are labeled by different configurations, and,

(ii) if any different nodes of $t(x)$ have the same label, then the subtrees [of $t(x)$] with these nodes as the roots are identical.

For each x in $V(m)$, let $t(m)$, which we call the *reduced accepting computation tree* of M on x, be a tree obtained from $t(x)$ by the following procedure [for each node v of $t(x)$, we denote by $d(v)$ the length of the path from the root of $t(x)$ to v (i.e., the number of edges from the root of $t(x)$ to v)]:

Begin

1. $T_r = t(x)$
2. $i = 1$
3. Let $N(i) \triangleq \{v \mid v$ is node of T_r *and* $d(v) \leq i\}$. Divide $N(i)$ as follows: $N(i) = P(1) \cup P(2) \cup \cdots \cup P(j_i)$, where: (1) if $i_a = i_b (1 \leq i_a, i_b \leq j_i)$, then $P(i_a) \cap P(i_b) = \phi$, and (2) for each $i_a (1 \leq i_a \leq j_i)$ and for each $v_a, V_b \in P(i_a)$, $l(v_a) = l(v_b)$ (i.e., the labels of v_a and v_b are identical). For each i_a $(1 \leq i_a \leq j_i)$, let $dis(i_a) = \min\{d(v) \mid v \in P(i_a)\}$ and let $n(i_a)$ be the leftmost node among those nodes v in $P(i_a)$ such that $d(v) = dis(i_a)$. Further, let $N'(i) = N(i) - \{n(1), n(2), \cdots, n(j_i)\}$. By removing from T_r all the subtrees whose roots are included in $N'(i)$, we make the new T_r.
4. If the height of T_r (i.e., the length of the longest path of T_r) is less than or equal to i, then we let $t'(x) = T_r$. Otherwise, we let $i = i + 1$ and go to step 3.

end

[**Example 1**] Let $x \in V(m)$ and $t(x)$ be a tree. Here, suppose that nodes A and D have the same label, nodes B and C have the same label, and other nodes each have different labels. [From the preceding assumption (ii) concerning $t(x)$, identical.] Then, $t'(x)$ is a tree. That is, $t'(x)$ is obtained from $t(x)$ by moving the subtree with nodes C and D as the roots from $t(x)$.

It is easily seen that for each x in $V(m)$, all the nodes of $t'(x)$ have labels different from one another, and the set of all the paths from root of $t'(x)$ to the leaves of $t'(x)$ represents necessary and sufficient accepting computations of M on x. From $t'(x)$, we now define an *extended crossing sequence* (*ECS*) at the boundary between the top and bottom halves of x. The concept of *ECS* was first introduced in [3]. We relabel each node v of $t'(x)$, as follows. (We denote this new labeling by l'.) For each node v of $t'(x)$, let $f(v)$ denote the father node of v. Then, for each node v of $t'(x)$, where $x \in V(m)$, let

if, for some storage states (q, α, j) and (q', α', j'),

(i) $l(f(v)) = (x, (i_1, i_2, i_3, m), (q', \alpha', j'))$ and
$l(v) = (x, (i_1, i_2, i_3, m+1), (q, \alpha, j))$, or

(ii) $l(f(v)) = (x, (i_1, i_2, i_3, m+1), (q', \alpha', j'))$ and
$l(v) = (x, (i_1, i_2, i_3, m), (q, \alpha, j))$, then

$l'(v) = ((i_1, i_2, i_3), (q, \alpha, j))$

else

$l'(v) = *$.

That is, if the movement of M from $f(v)$ to v represents the action of crossing the boundary between the top and bottom halves of x, then v is newly labeled by $(i_1, i_2, i_3), (q, \alpha, j))$, where (q, α, j) is the storage state component of $l(v)$. Otherwise, v is newly labeled by $*$. From the newly labeled $t'(x)$, we extract those nodes v such that $l'(rn) = *$, and by using these nodes, we construct a tree $t''(x)$ satisfying the following condition:

(A) For any node v of $t''(x)$, nodes $v_1, v_2, \cdots, v_S$ are children of v if and only if $v_1, v_2, \cdots, v_S$ are descendants of v in $t'(x)$ and $l'(u) = *$ for each node u on the path from v to each v_i. In general, there can be two or more such trees $t''(x)$. Let these trees be $t_1''(x),\cdots, t_n''(x)$. For each node v of each $t_i''(x)$ ($1 \le i \le n$), we now define an *element* of *ECS* (*EECS*) inductively as follows: Let

$$l'(v) = ((i_1, i_2, i_3), (q, \alpha, j)).$$

(1) If v is a leaf, then $[((i_1, i_2, i_3), (q, \alpha, j))]$ is an *EECS* of v.

(2) If v has only one child v_1 and $Q_1 = [((i_{11}, i_{21}, i_{31}), (q_1, \alpha_1, j_1))P]$ is an *EECS* of v_1, then $[((i_1, i_2, i_3), (q, \alpha, j)) ((i_{11}, i_{21}, i_{31}), (q_1, \alpha_1, j_1))P]$ is an *EECS* of v.

(3) If v has $d(\geq 2)$ children $v_1, \cdots, v_d$ and $Q_1, \cdots, Q_d$ are *EECS*'s of $v_1, \cdots, v_d$, respectively, then $[((i_1, i_2, i_3), (q, \alpha, j))\, Q_{\sigma(1)}\cdots Q_{\sigma(d)}]$ is an *EECS* of v for any permutation $\sigma : \{1, \cdots, d\}\rightarrow\{1, \cdots, d\}$.

(4) An *EECS* of v is defined only by the preceding statements (1), (2), and (3).

Now, let $Q_1, \cdots, Q_n$ be *EECS*'s of the root nodes of $t_1''(x),\cdots, t_n''(x)$, respectively. Then, for any permutation $\sigma : \{1, \cdots, n\}\rightarrow\{1, \cdots, n\}$, we call $Q_{\sigma(1)}, \cdots, Q_{\sigma(n)}$ an *ECS* of x. As is easily seen from the definitions, there can be two or more *EECS*'s of each node v of each $t''(x)$, and there can be two or more *ECS*'s of x. Let Q_1 and Q_2 be any two *EECS*'s. If the following condition (B) is satisfied, we say Q_1 and Q_2 are *equivalent* and write $Q_1 \equiv Q_2$:

(B) Let $Q_1 = [((i_{11}, i_{21}, i_{31}), (q_1, \alpha_1, j_1)) \cdots((i_{1n}, i_{2n}, i_{3n}), (q_n, \alpha_n, j_n))\, P_1 \cdots P_S]$, $Q_2= [((i'_{11}, i'_{21}, i'_{31}), (q'_1, \alpha'_1, j'_1)) \cdots((i'_{1n'}, i'_{2n'}, i'_{3n'}), (q'_{n'}, \alpha'_{n'}, j'_{n'}))\, P'_1 \cdots P'_{S'}]$. Then $n = n'$, $s = s'$, and $((i_{1k}, i_{2k}, i_{3k}), (q_k, \alpha_k, j_k))= ((i'_{1k}, i'_{2k}, i'_{3k}), (q'_k, \alpha'_k, j'_k))$ for each k ($1 \le k \le n$), and there exists a permutation $\sigma : \{1, \cdots, s\} \rightarrow\{1, \cdots, s\}$ such that $P_i \equiv P'_{\sigma(i)}$ for each $i(1 \le i \le s)$, where $n, s \geq 0$, and $((i_1, i_2, i_3), (q, \alpha, j))$'s and $((i'_1, i'_2, i'_3), (q', \alpha', j'))$'s are pairs (coordinates along the fourth axis, storage state), and further P, P' are *EECS*'s.

Let $Q = Q_1 \cdots Q_n$, $Q' = Q'_1 \cdots Q'_n$ be any two *ECS*'s. We say that Q and Q' are equivalent if $n = n'$ and there exists a permutation $\sigma : \{1, \cdots, n\} \rightarrow\{1, \cdots, n\}$ such that $Q_i\equiv Q'_{\sigma(i)}$ for each i ($1 \le i \le n$). [As is easily seen from the definition, any two *ECS*'s of x are equivalent for any x in $V(m)$.] For any *ECS* Q, the length of Q is the number of pairs (coordinates along the fourth axis, storage state) in Q, and is denoted by $|Q|$. For each $m\geq 1$, let $E(m) = \{Q \mid Q$ is an *ECS* of x for some x in $V(m)\}$. Then, the following two propositions must hold :

[**Proposition 1**] $|E(m)| = Z(m)^{dZ(m)}$, *where* $Z(m) = (2m + 2)^3 rL(2m)s^{L(2m)}$, *r and s are the numbers of states* (*of the finite control*) *and storage-tape symbols of M, and d is a positive constant*.

[**Proposition 2**] *Let x and y be any two different tapes in* $V(m)$, *and let* Q_x *and* Q_y *be any ECS's of x and y, respectively. Then,* Q_x *and* Q_y *are not equivalent*.

Clearly, $|V(m)| = 2^{8t}$ ($t=m^4$) . Because $L(m)=o(\log m)$, it follows from Proposition 1 that $|V(m)| > |E(m)|$ for large m. For such a large m, there must exist two different tapes $x, y \in V(m)$ such that some *ECS* of x and some *ECS* of y are equivalent, which contradicts Proposition 2. This completes the proof. □

3. Conclusion

In this paper, we presented a technique which we can show that a four-dimensional language is not accepted by any space-bounded alternating Turing machines. It will be interesting to investigate infinite space hierarchy properties of the classes of sets accepted by 4-*ATM*'s with spaces of size smaller than log m.

References

[1] A.K.Chandra, D.C.kozen, and L.J.Stockmeyer: Alternation, *J.ACM*, Vol. 28, No. 1 pp.114-133 (1981).
[2] M.Sakamoto, K.Inoue, and I.Takanami: A note on three-dimensional alternating Turing machines with space smaller than log *m*, *Inform. Sci.*, *ELSEVIER*, Vol. 72, pp. 225-249 (1993).
[3] H.Yamamoto and S.Noguchi: Time-and-leaf bounded 1-tape alternating Turing machines, *The Trans. of IECE*, J68-D, pp.1719-1726 (1985).