数理解析研究所講究録
1041 巻 1998 年 135-142

Alternating Automata Characterizations of One-Way Iterative Arrays

山口大学工学部　伊藤暁，井上克司，王躍
Akira ITO, Katsushi INOUE, and Yue WANG

Abstract Let $k\mathbf{OIA}(T(n))$ denote the class of languages accepted by k-dimensional one-way iterative arrays in $T(n)$ time. We abbreviate notions such that opt $= (k+1)n-k$, $k\mathbf{OIA}$(linear) $= \bigcup_{c>0} k\mathbf{OIA}(\text{opt}+cn)$, and $k\mathbf{OIA}$(poly-l) $= k\mathbf{OIA}(\text{opt}+n^l)$. Further, let $1\mathbf{AFA}(k)$, $2\mathbf{AFA}(k)$ denote the classes of languages accepted by one-way and two-way alternating k-head finite automata, respectively. The main purpose of this paper is to show that for each $k, l \geq 1$, (1) $k\mathbf{OIA}(\text{opt})^{\text{R}} = 1\mathbf{AFA}(k+1)$, (2) $k\mathbf{OIA}$(linear) = 1-turn $2\mathbf{AFA}(k+1)$, and (3) $k\mathbf{OIA}$(poly-l) $\subseteq 2\mathbf{AFA}(k+l)$, where "1-turn" of $2\mathbf{AFA}(k)$ means that only one head among the k heads can turn at the right end of the input and move to the left, while all the other heads must stop after arriving at the right end. The superscript 'R' stands for reversal operation. Moreover, we show that (4) 1-turn $2\mathbf{AFA}(k) \subseteq 1\mathbf{AFA}(k+1)$.

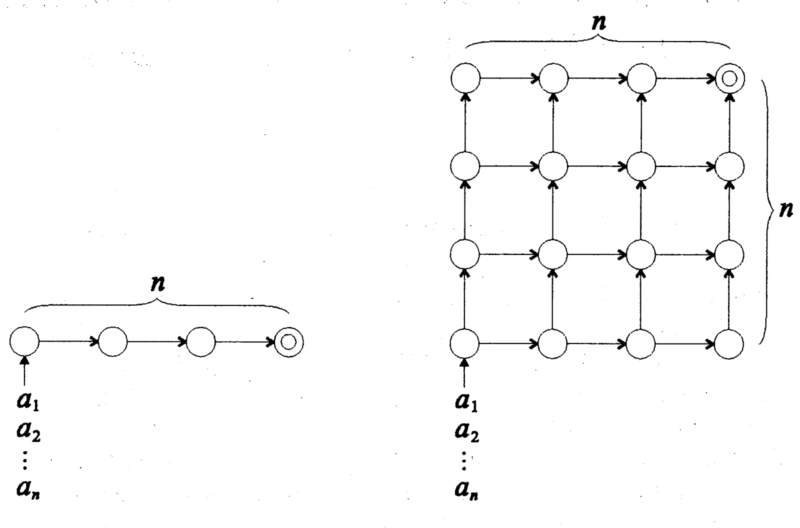
1. Introduction

One of the simplest models of parallel computation is the *one-way iterative array* (OIA) [2, 3]. Figure 1 shows one-dimensional iterative array (1OIA) and two-dimensional iterative array (2OIA). These lower-dimensional arrays can be generalized to k dimensions ($k \geq 1$), i.e., to a k-dimensional one-way iterative array (kOIA). Such an array has size $n \times \cdots \times n$ (k times). The input cell is at position $(1, \ldots, 1)$ and the accepting cell is at position $(n, \ldots, n)$. A kOIA operates in time $T(n)$ is denoted by $k\text{OIA}(T(n))$. Let $k\mathbf{OIA}(T(n))$ denote the class of languages accepted by $k\text{OIA}(T(n))$s. For special complexity functions, define opt $= (k+1)n-k$, $k\mathbf{OIA}$(linear) $= \bigcup_{c>0} k\mathbf{OIA}(\text{opt}+cn)$, and $k\mathbf{OIA}$(poly-l) $= k\mathbf{OIA}(\text{opt}+n^l)$. Note that time complexity is at least $(k+1)n-k$ for any $k \geq 1$ [2, 6].† Meanwhile, multihead finite automata has being studied especially from theoretical interest [4, 11] as a simplest extension of finite automaton. A *k-head finite automaton* is a finite-state automaton with k-heads on a single read-only input tape. Let $1\mathbf{AFA}(k)$ ($1\mathbf{NFA}(k)$) denote the class of languages accepted by one-way k-head alternating (nondeterministic) finite automata. We denote the two-way version of $1\mathbf{AFA}(k)$ by $2\mathbf{AFA}(k)$. As a relationship between multihead finite automata and one-way iterative arrays, it is known [6] that $1\mathbf{NFA}(k+1) \subseteq k\mathbf{OIA}(\text{opt})$ for each $k \geq 1$. Section 3 of this paper generalizes it to the best result $1\mathbf{AFA}(k+1)^{\text{R}} = k\mathbf{OIA}(\text{opt})$, where 'R' stands for reversal operation.

One of the well-known open problems concerning to OIA is whether $1\mathbf{OIA}(\text{opt}) \subsetneq 1\mathbf{OIA}(\text{linear})$. Section 4 of this paper connects this problem to AFA-side open problems whether $1\mathbf{AFA}(2) \subsetneq 2\mathbf{AFA}(2)$ and whether $1\mathbf{AFA}(2) \subsetneq 1\mathbf{AFA}(3)$, by showing that $k\mathbf{OIA}$(linear) = 1-turn $2\mathbf{AFA}(k+1)$ = finite-turn $2\mathbf{AFA}(k+1) \subseteq 1\mathbf{AFA}(k+2)$, where "1-turn" of $2\mathbf{AFA}(k)$ means that only one head among the k heads can turn at the right end of the input and move to the left, while all the other heads must stop after arriving at the right end, starting at the left end. The term "finite-turn" means that the number of full scans of each two-way head between both ends of the input (i.e., from the left end to the right end or vice versa) is restricted to be finite. This section also shows that $k\mathbf{OIA}(\text{poly-}l) \subseteq 2\mathbf{AFA}(k+l)$.

Section 6 summarizes the paper with concluding remarks.

†In [7, 9], $(k+1)n-k$ time is called 'pseudo-real-time' when $k=1$.



(1) 1OIA (2) 2OIA

Figure 1. One-dimensional and two-dimensional one-way iterative arrays.

2. Definitions

"Unrolling" the computation of a kOIA in space and time [1, 5], we obtain an array of combinational circuits: the $(k+1)$-dimensional trellis automaton.

Definition 2.1. A $(k+1)$-*dimensional trellis automaton* (kTA) is a system $A = (\mathbf{E}, \Gamma, \Sigma, \Delta, f)$, where $\mathbf{E}$ is the non-zero quadrant $\{(i_0, i_1, \ldots, i_k) \in \mathbb{Z}^{k+1} \mid i_0, i_1, \ldots, i_k \geq 0\}$ of $(k+1)$-dimensional discrete space $\mathbb{Z}^{k+1}$. At each point $(i_0, i_1, \ldots, i_k)$ of $\mathbf{E}$, an identical element, called $(i_0, i_1, \ldots, i_k)$-*element*, computes a partial function $f : \Gamma^{k+1} \to \Gamma$, where Γ (which contains the blank symbol λ) is a finite *operational alphabet.* Output $s(i_0, i_1, \ldots, i_k)$ of $(i_0, i_1, \ldots, i_k)$-element is recursively defined: $s(i_0, i_1, \ldots, i_k) = f(s_0, s_1, \ldots, s_k)$, where $s_0 = s(i_0 - 1, i_1, \ldots, i_k)$, $s_1 = s(i_0, i_1 - 1, \ldots, i_k), \ldots$, and $s_k = s(i_0, i_1, \ldots, i_k - 1)$, with boundary condition $s(0, i_1, i_2, \ldots, i_k) = s(i_0, 0, i_2, \ldots, i_k) = \cdots = s(i_0, i_1, \ldots, i_{k-1}, 0) = \lambda$ for each $i_0, i_1, \ldots, i_k \geq 1$ except that $s(i_0, 1, \ldots, 1, 0) = a_{i_0}$, $1 \leq i_0 \leq n$. We says that A *accepts* input $a_1 a_2 \ldots a_n \in \Sigma^*$ in time $T(n)$ if $s(T(n), n, \ldots, n) \in \Delta$, where $\Sigma, \Delta \subseteq \Gamma - \{\lambda\}$ are the sets of *input symbols* and *accepting symbols*, respectively.

Let $(k+1)\mathrm{TA}(T(n))$ denote a $(k+1)$TA operates in time $T(n)$ and $(k+1)\mathbf{TA}(T(n))$ denote the class of languages accepted by $(k+1)\mathrm{TA}(T(n))$'s. Figure 2 illustrates $2\mathrm{TA}(T(n))$. It is clear that $(k+1)\mathbf{TA}(T(n)) = k\mathbf{OIA}(T(n))$ for any $k \geq 1$ and any $T(n) \geq (k+1)n - k$.

Next, we introduce a variety of alternating multihead finite automaton, ranging from ordinary one-way machine to two-way one [11]. We first define the most general type of multihead finite automaton: the two-way alternating finite automaton.

Definition 2.2. An *alternating finite automaton with k heads* (AFA(k)) is a structure $M = (Q, \Sigma, \delta, q_0, U, F)$, where (1) Q is a finite, nonempty set of states; (2) Σ is the input alphabet (Σ does not contain the symbols ¢ and \$); (3) δ is the transition function, mapping $Q \times (\Sigma \bigcup \{¢, \$\})^k$ into the subsets of $Q \times \{-1, 0, +1\}^k$, with the restriction that for any transition $(q, (d_1, \ldots, d_k)) \in \delta(p, (a_1, \ldots, a_k))$, $a_j = ¢$ implies $d_j \geq 0$ and $a_j = \$$ implies $d_j \leq 0$ for any $j (1 \leq j \leq k)$; (4) $q_0 \in Q$ is the initial state; (5) $U \subseteq Q$ is the set of universal states; and (6) $F \subseteq Q$ is the set of accepting states.

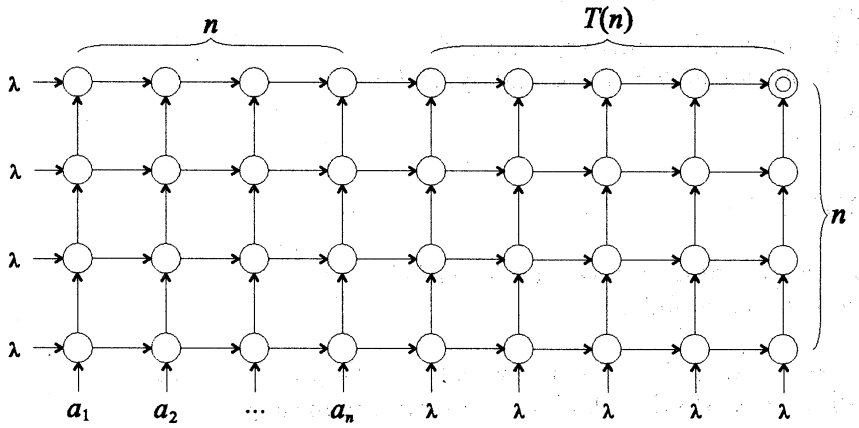


Figure 2. 2TA($T(n)$).

The *language accepted by* M is $L(M) = \{x \in \Sigma^* \mid M \text{ accepts } x\}$. The class of languages accepted by AFA(k)'s is denoted by $\mathbf{AFA}(k)$.

We next introduce special types of AFA(k) whose input heads are restricted in various ways. A *sweeping* head is a two-way head whose motions are restricted to be sweeping ones between both ends of the input (i.e., from the left end to the right end or vice versa). A *finite-turn* head is a sweeping head whose number of sweeps is restricted to be finite. A *l-turn* head ($l \geq 0$) is a finite-turn head which can reverse its direction at most l times. A *one-way* head is an alias of 0-turn head. An AFA(k) with k_0 two-way heads, k_1 sweeping heads, k_2 finite-turn heads, k_3 1-turn heads, and k_4 one-way heads is denoted by AFA($k_0 + \overline{k_1} + \widetilde{k_2} + \widehat{k_3} + \overrightarrow{k_4}$).

The superscript 'b' of head notion implies their *blindness*, i.e., they cannot read input symbols except the left and right boundary symbols ¢ and \$, respectively [4, 12].

3. Optimal-Time kOIA

In this section, we shows the equivalence of optimal-time OIA and one-way AFA through reversal operation of languages. In order to clarify our proofs, we introduce a special type of head of AFA, a *reversal-one-way* head which can move right-to-left, rather than left-to-right as an ordinary one-way head. An AFA(k) with k' reversal-one-way heads is denoted by AFA($\cdots + \overleftarrow{k'} + \cdots$). It is clear that $\mathbf{AFA}(\overleftarrow{k}) = \mathbf{AFA}(\overrightarrow{k})^{\mathrm{R}}$ and $\mathbf{AFA}(\overleftarrow{1} + {}^{\mathrm{b}}\overleftarrow{k}) = \mathbf{AFA}(\overrightarrow{1} + {}^{\mathrm{b}}\overrightarrow{k})^{\mathrm{R}}$.

Lemma 3.1. *For each $k \geq 1$, k***OIA**(opt) $\subseteq$ **AFA**$(\overleftarrow{1} + {}^{\mathrm{b}}\overleftarrow{k})$.

Proof. Let $A = (\mathbf{E}, \Gamma, \Sigma, \Delta, f)$ be a $(k+1)$TA(opt) equivalent to a kOIA(opt) accepting some language L. It is easily seen that, given an input $a_1 \ldots a_n$, each $(i, 1, \ldots, 1)$-element $(1 \leq i \leq n)$ of A can distribute its input symbol a_i to all the elements having the same i-coordinates. We therefore assume that the initial condition of A is the following: For each $1 \leq i, i_1, \ldots, i_k \leq n$, $s(0, i_1, i_2, \ldots, i_k) = \lambda$, and $s(i, 0, i_2, \ldots, i_k) = s(i, i_1, 0, \ldots, i_k) = \cdots = s(i, i_1, \ldots, i_{k-1}, 0) = a_i$.

Consider an AFA$(\overleftarrow{1} + {}^{\mathrm{b}}\overleftarrow{k})$ $M = (Q, \Sigma, \delta, q_0, U, F)$ which executes the program shown in Fig. 3 on a tape x. We can imagine M to be an ordinary one-head finite automaton whose input tape x is a $(k+1)$-dimensional 'hyper-rectangle.' Correctness of the algorithm can be proved by induction as in the same way with that of Proposition 3.1 in [10]. We therefore conclude that $L = L(M)$. □

Main program
begin // start from the right boundary symbol \$ //
  moves all heads one square left (to the last symbol of x);
  guess $\dot{s}_a \in \Delta$;
  OUTPUT($\dot{s}_a$)
end
Procedure OUTPUT($\dot{s}$)
begin
  if some head reaches $\not\in$ **and** reading head h_0 reads $\dot{s}$
    then accept **else** reject and halt
  else
    guess $\dot{s}_0, \dot{s}_1, \dot{s}_2, \ldots, \dot{s}_k \in \Gamma$ in such a way that $\dot{s} = f(\dot{s}_0, \dot{s}_1, \dot{s}_2, \ldots, \dot{s}_k)$;
    for each $j (0 \leq j \leq k)$ **do** the following in universal branching
      begin
        moves head h_j one square to the left;
        OUTPUT($\dot{s}_j$)
      end
end

Figure 3. Algorithm of reversal-one-way AFA simulating a OIA.

Lemma 3.2. *For each* $k \geq 2$, $\mathbf{AFA}(\overleftarrow{k}) \subseteq (k-1)\mathbf{OIA}(\mathrm{opt})$.

Proof. Our technique is essentially the same as that used in the proof of Theorem 6.2 in [11], i.e., 'reverse breadth-first search' of the directed graph induced from all possible configurations of an AFA.

Without loss of generality, we impose the following assumptions on any AFA($\overleftarrow{k}$) $M = (Q, \Sigma, \delta, q_0, U, F)$, $k \geq 2$.

(1) Initially, all heads of M stay at the one left square to the right boundary symbol \$, i.e., the initial configuration of M on a tape x is $(q_0, (n, n, \ldots, n))$, where $n = |x|$.

(2) At each step, any two heads of M does not move left simultaneously. From this, we modify the original transition function δ of M to be a mapping δ' from $Q \times (\Sigma \bigcup \{\$\})^k$ into the subsets of $Q \times \{0, 1, 2, \ldots, k\}$, where $(q, h) \in \delta'(p, \boldsymbol{a})$ implies a transition such that the hth head moves right and all other heads keep stationary (especially $h = 0$ means all heads stationary).

(3) If some head reaches $\not\in$, all heads keep stationary afterward, i.e., $(q, h) \in \delta'(p, (a_1, \ldots, \not\in, \ldots, a_k))$ implies $h = 0$. This is possible because we can modify M to the desired machine M' as follows. Each time when some head h of M is going to move left, M' guesses whether the symbol of the square that h will enter is $\not\in$ or not and universally branches into two machines, one of which really moves h to the left and checks the correctness of the guess (and halts), other of which continues the simulation of M, assuming $\not\in$ on the left neighbor square (and keeping h stationary).

Let $M' = (\{q_0, q_1, \ldots, q_s\}, \Sigma, \delta', q_0, U, F)$ be an AFA($\overleftarrow{k}$) which satisfies the above-mentioned conditions. Prior to the simulation of M', we prepare a look-up table $g : (\{0,1\}^{s+1})^k \times (\Sigma \bigcup \{\$\})^k \rightarrow \{0,1\}^{s+1}$, which is constructed by the algorithm shown in Fig. 4.

Note that the algorithm only uses the information of M', not of individual input string and that the resultant table g is of finite memory size.

Consider a kTA $A = (\mathbf{E}, \Gamma, \Sigma, \Delta, f)$ on a tape $x = a_1 a_2 \ldots a_n$. By using a standard 'folding' (and distribution) technique as in the proof of Theorem 3 in [6], we can assume that partial data

Procedure $g(\boldsymbol{u}^{(1)}, \boldsymbol{u}^{(2)}, \ldots \boldsymbol{u}^{(k)}, \boldsymbol{a})$
begin
  for each $r(0 \leq r \leq s)$ initialize $u_r^{(0)} := \begin{cases} 1, & \text{if } q_r \in F, \\ 0, & \text{otherwise.} \end{cases}$
  repeat s times **do**
    begin
      for each r's $(0 \leq r \leq s)$ such that $u_r^{(0)} = 0$ **do**
        begin
          if $q_r \in U$ **then**
            if *for all* pair (l, h) $(0 \leq l \leq s, 0 \leq h \leq k)$ such that
            $(q_l, h) \in \delta'(q_r, \boldsymbol{a})$, it holds that $u_l^{(h)} = 1$ **then** $u_r^{(0)} := 1$
          else // $q_r \in Q - U$ //
            if *at least one* pair (l, h) $(0 \leq l \leq s, 0 \leq h \leq k)$ such that
            $(q_l, h) \in \delta'(q_r, \boldsymbol{a})$, it holds that $u_l^{(h)} = 1$ **then** $u_r^{(0)} := 1$
        end
    end
  return $\boldsymbol{u}^{(0)} = (u_0^{(0)}, u_1^{(0)}, \ldots, u_s^{(0)})$
end

Figure 4. Construction algorithm of look-up table g of AFA($\overleftarrow{k}$).

$(a_{i_1}, a_{i_2}, \ldots, a_{i_k})$ of input x is available for each $(i_1, i_2, \ldots, i_k)$-element of A $(1 \leq i_1, i_2, \ldots, i_k \leq n)$. The operational function f of A with such a distributed input is thus defined as follows.

For each $1 \leq i_1, i_2, \ldots, i_k \leq n$, $(i_1, i_2, \ldots, i_k)$-element outputs the vector $\boldsymbol{v}^{(0)} = g(\boldsymbol{v}^{(1)}, \boldsymbol{v}^{(2)}, \ldots, \boldsymbol{v}^{(k)}, (a_{i_1}, a_{i_2}, \ldots, a_{i_k}))$, where $\boldsymbol{v}^{(1)}$ is the input vector from $(i_1 - 1, i_2, \ldots, i_k)$-element, $\boldsymbol{v}^{(2)}$ is the input vector from $(i_1, i_2 - 1, i_3, \ldots, i_k)$-element, $\ldots$, and $\boldsymbol{v}^{(k)}$ is the input vector from $(i_1, i_2, \ldots, i_{k-1}, i_k - 1)$-element except that when $i_j = 1$, $\boldsymbol{v}^{(j)} = g(\boldsymbol{v}^*, \ldots, \boldsymbol{v}^*, (a_1, \ldots, \underset{\substack{\uparrow \\ j}}{\not\epsilon}, \ldots, a_k))$, where '$\boldsymbol{v}^*$' stands for an arbitrary (don't-care) vector.

Recall from the assumption (3) that all heads stop their motions after some head reaches $\not\epsilon$, so the actual values of parameters $\boldsymbol{u}$'s have no relevance to the reference of g when another parameter $\boldsymbol{a}$ contains symbol $\not\epsilon$.

Correctness of the whole algorithm can be proved by induction as in the same way with that of Proposition 3.2 in [10]. Therefore, we can conclude that A accepts the same language as M'. It follows that a $(k-1)$OIA(opt) equivalent to A accepts $L(M')$. □

From Lemma 3.1 and 3.2, we get the main theorem of this section.

Theorem 3.1. *For each* $k \geq 1$, k**OIA**(opt)$^{\mathrm{R}}$ = **AFA**$(\vec{1} + \vec{k}^{\mathrm{b}})$ = **AFA**$(k + \vec{1})$. □

4. Linear-Time and Polynomial-Time kOIA's

In this section, we investigate the relationships between non-optimal-time OIA's and AFA's. Notably, we show that linear-time OIA's are completely characterized with finite-turn AFA's. Almost proofs are based on the characterization of time-bounded OIA in terms of a special type of one-way AFA.

Definition 4.1. For any language L and any non-negative function $T(n)$, define $L\#^{T(n)} = \{x\#^{T(n)} \mid x \in L\}$ and $\#^{T(n)}L = \{\#^{T(n)}x \mid x \in L\}$, where $\#$ is not in the alphabet of L.

In the below, superscript "#" on $\vec{k}$ means that the corresponding $\vec{k}$ heads initially stay at the rightmost symbol of padding substring #'s, i.e., one square left to substring x.

As an easy generalization of Theorem 3.1, we have the following characterization of arbitrary-time bounded OIA (recall that $L\#^0 = L$).

Theorem 4.1. *Let L be any language and $T(n)$ be any non-negative function. Then, for each $k \geq 1$, $L \in k\mathbf{OIA}(\mathrm{opt} + T(n)) \iff \#^{T(n)}L^{\mathrm{R}} \in \mathbf{AFA}(\vec{1} + {}^{\#}\vec{k}^{\mathrm{b}})$.* □

By using the above characterization, we get the following speed-up theorem for multi-dimensional OIA, which includes one- and two-dimensional versions [7, 8] as special cases.

Theorem 4.2 (multi-dimensional speed-up). *Let $T(n)$ be any non-negative function and $d > 0$ be any constant. Then, $k\mathbf{OIA}(\mathrm{opt} + T(n)) = k\mathbf{OIA}(\mathrm{opt} + T(n)/d)$ for each $k \geq 1$.*

Proof. From Theorem 4.1, it is sufficient to show that $\#^{T(n)}L \in \mathbf{AFA}(\vec{1} + {}^{\#}\vec{k}^{\mathrm{b}}) \Rightarrow \#^{T(n)/d}L \in \mathbf{AFA}(\vec{1} + {}^{\#}\vec{k}^{\mathrm{b}})$. Let M be an AFA$(\vec{1} + {}^{\#}\vec{k}^{\mathrm{b}})$ accepting $\#^{T(n)}L$. An AFA$(\vec{1} + {}^{\#}\vec{k}^{\mathrm{b}})$ M' that simulates M behaves in the same way as M, except when the reading head h of M is on padding symbols #'s of the given tape: Each time h moves d squares to the right, M' moves its reading head h' one square to the right. It is clear that M' accepts $\#^{T(n)/d}L$. □

Substituting $T(n) = dn$, we have the speed-up theorem in linear-time range.

Corollary 4.1. *For each $k \geq 1$, $k\mathbf{OIA}(\mathrm{linear}) = k\mathbf{OIA}(\mathrm{opt} + n)$.* □

This fact will be used in the following discussions. In the case of linear-time OIA, we can relax the restriction of initial head positions on the equivalent one-way AFA:

Proposition 4.1. *Let L be any language. Then, for each $k \geq 1$, $\#^n L \in \mathbf{AFA}(\vec{1} + {}^{\#}\vec{k}^{\mathrm{b}}) \iff \#^n L \in \mathbf{AFA}(\vec{1} + \vec{k}^{\mathrm{b}})$.* □

The following pair of lemmas leads to the theorem which asserts the equivalence of linear-time OIA and finite-turn AFA.

Lemma 4.1. *For each $k \geq 1$, $k\mathbf{OIA}(\mathrm{opt} + n)^{\mathrm{R}} \subseteq \mathbf{AFA}(\hat{1} + \vec{k}^{\mathrm{b}})$.*

Proof. From Theorem 4.1 and Proposition 4.1, it is sufficient to show that $\#^n L \in \mathbf{AFA}(\vec{1} + \vec{k}^{\mathrm{b}}) \Rightarrow L \in \mathbf{AFA}(\hat{1} + \vec{k}^{\mathrm{b}})$. Let M be an AFA$(\vec{1} + \vec{k}^{\mathrm{b}})$ accepting $\#^n L$. We construct an AFA$(\hat{1} + \vec{k}^{\mathrm{b}})$ M' which acts as follows. M' behaves in the same way as M, except that some blind head b' is used for the reading head r of M (and its reading head r' is used for the corresponding blind head b of M). When b' wants to read input symbol, M' performs a guess-and-check procedure similar to that used in the assumption (3) of the proof of Lemma 3.2. It is clear that M' accepts L. □

Lemma 4.2. *For each $k \geq 1$, $\mathbf{AFA}(\widetilde{k+1}) \subseteq k\mathbf{OIA}(\mathrm{opt} + n)$.*

Proof. Indirect simulation using one-way AFA (Theorem 4.1) are rather complicated. Instead, we briefly describe the direct simulation of AFA$(\widetilde{k+1})$ by kOIA$(\mathrm{opt} + n)$. Figure 5 illustrates a trellis automaton equivalent to a 1OIA$(\mathrm{opt} + n)$ that simulates an AFA$(\tilde{2})$ whose first head is 1-turn and second is 2-turn. The shaded area in the figure is used for actual simulation. The remaining part is used only for sending the input information to the simulation area. The detailed construction is omitted. □

From Lemma 4.1 and Lemma 4.2, we have the following equivalence.

Theorem 4.3. *For each $k \geq 1$, $k\mathbf{OIA}(\mathrm{linear}) = \mathbf{AFA}(\widetilde{k+1}) = \mathbf{AFA}(\hat{1} + \vec{k}^{\mathrm{b}})$.* □

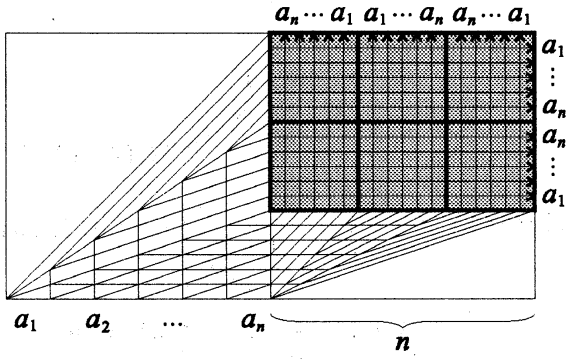


Figure 5. Simulation of AFA($\tilde{2}$) by 1OIA(opt $+ n$).

Exchanging the blindness of zero-turn head and non-zero-turn heads of AFA($\hat{1} + \vec{k}^{\mathrm{b}}$) (or strengthening the one-way blind heads of AFA($\vec{1} + \vec{k}^{\mathrm{b}}$) to finite-turn ones), we get a refined hierarchy between k**OIA**(opt) and k**OIA**(linear):

Theorem 4.4. $\mathbf{AFA}(\overrightarrow{1} + \overrightarrow{k-1}^{\mathrm{b}} + \hat{1}^{\mathrm{b}}) = \mathbf{AFA}(\hat{1}^{\mathrm{b}} + \vec{k}) = \mathbf{AFA}(\vec{1} + \tilde{k}^{\mathrm{b}})$. □

Next, we show that finite-turning ability of input heads can be discarded at the cost of one additional head.

Theorem 4.5. *For each* $k \geq 2$, $\mathbf{AFA}(\tilde{k}) \subseteq \mathbf{AFA}(\overrightarrow{k+1})$.

Proof. From Theorem 3.1 and Theorem 4.3, it is sufficient to show that $\mathbf{AFA}(\vec{k}^{\mathrm{b}} + \hat{1})^{\mathrm{R}} \subseteq \mathbf{AFA}(\vec{k}^{\mathrm{b}}+\vec{2})$. Let M be an AFA($\vec{k}^{\mathrm{b}}+\hat{1}$) accepting some language L. We construct an AFA($\vec{k}^{\mathrm{b}}+\vec{2}$) M' which accepts L^{R} as follows. The $\vec{k}^{\mathrm{b}}$ heads of M' behave in the same way as the $\vec{k}^{\mathrm{b}}$ heads of M, except that they move in the reversal direction of that of M. The simulation of the $\hat{1}$-type head h of M consists of two stages. Stage 1: Before h turns at the left end of the input, M' moves one of its $\vec{2}$ heads, say h', in the same way as h except that it moves in the reversal direction, the remaining head h'' being kept stationary at the left end. When h' wants to read input symbol, M' performs a guess-and-check procedure similar to that used in the assumption (3) of the proof of Lemma 3.2. Stage 2: After h turns at the left end (h' arrives at the right end), M' begins to move h'' to the right as h moves right. □

As a final result, we show that a polynomial-time kOIA can be simulated by a two-way $(l+k)$-head AFA, where l is the degree of the polynomial.

Theorem 4.6. *For each* $k, l \geq 1$, k**OIA**(poly-l) $\subseteq \mathbf{AFA}(\overrightarrow{l-1} + \overrightarrow{k+1}^{\mathrm{b}})$. □

5. Conclusion

Figure 6 summarizes the main results obtained in this paper. In the figure, we are omitting prefix "**AFA**" from language class notation of AFA's, e.g., "k+1" is the abbreviation of "$\mathbf{AFA}(k+1)$." It is unknown whether any one of the inclusions is proper or not.

References

[1] C. Choffrut and K. Culik II, "On real time cellular automata and trellis automata," *Acta Informatica* **21**, pp.393–407 (1984).

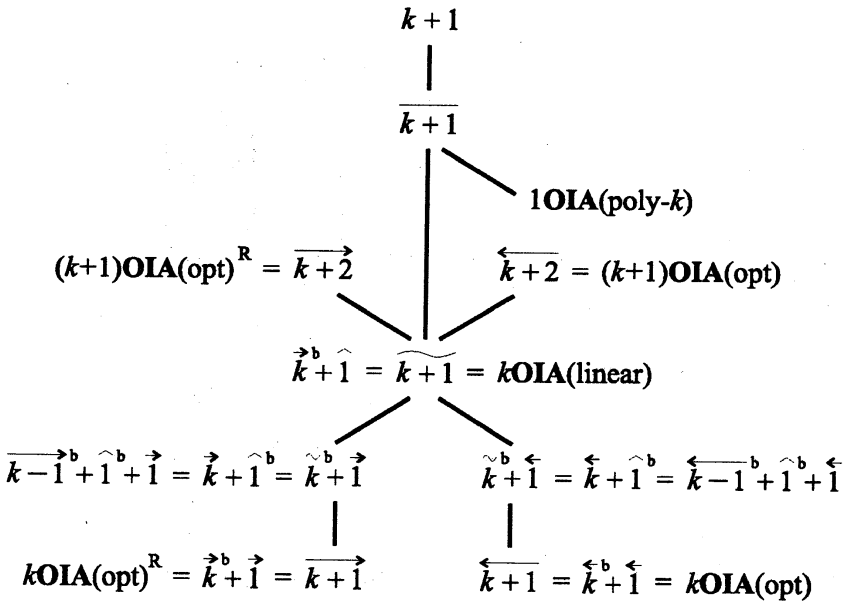


Figure 6. Relationships between classes of languages accepted by time-bounded OIA's and multihead AFA's ($k \geq 1$).

[2] J. H. Chang, O. H. Ibarra, and A. Vergis, "On the power of one-way communication," *JACM* **35**, No.3, pp.697–726 (1988).

[3] J. Gruska, "Synthesis, structure and power of systolic computations," *Theoret. Comput. Sci.* **71**, pp.47–77 (1990).

[4] J. Hromkovic, "On the power of alternation in automata theory," *JCSS* **31**, pp.28–39 (1985).

[5] O. H. Ibarra, S. M. Kim, and S. Moran, "Sequential machine characterization of trellis and cellular automata and applications," *SIAM J. Comput.* **14**, No.2, pp.426–447 (1985).

[6] O. H. Ibarra, "Systolic arrays: characterizations and complexity," *Proc. MFCS'86, LNCS* **233**, pp.140–153 (1986).

[7] O. H. Ibarra, and T. Jiang, "On one-way cellular arrays," *SIAM J. Comput.* **16**, No.6, pp.1135–1154 (1987).

[8] O. H. Ibarra and M. A. Palis, "Two-dimensional iterative arrays: characterizations and applications," *Theoret. Comput. Sci.* **57**, pp.47–86 (1988).

[9] O. H. Ibarra and T. Jiang, "Relating the power of cellular arrays to their closure properties," *Theoret. Comput. Sci.* **57**, pp.225–238 (1988).

[10] A. Ito, K. Inoue, and I. Takanami, "Determinisitic two-dimensional on-line tessellation acceptors are equivalent to two-way two-dimensional alternating finite automata through 180°-rotation," *Theoret. Comput. Sci.* **66**, pp.273–287 (1989).

[11] K. N. King, "Alternating multihead finite automata," *Theoret. Comput. Sci.* **61**, pp.149–174 (1988).

[12] H. Matsuno, K. Inoue, H. Taniguchi, and I. Takanami, "Alternating simple multihead finite automata," *Theoret. Comput. Sci.* **36**, pp.299–308 (1985).