Bulletin of Informatics and Cybernetics, Vol. 23, No. 1~2, 1988

INDEXING ALTERNATING FINITE AUTOMATA AND BINARY TREE LIKE CIRCUITS

By

Satoru MIYANO*

Abstract

We introduce indexing alternating finite automata and establish a relationship with uniform boolean circuits.

1. Introduction

NC^k is the class of functions computable by uniform boolean circuit families of size $n^{O(1)}$ and depth $O((\log n)^k)$ for $k \geqq 1$, where n is the number of inputs [2-3]. The class $NC=\bigcup_{k\geq 1}NC^k$ is understood as the class of efficiently parallelizable problems independent of the choice of parallel computation models [2-5]. Especially, the subclasses NC^1 and NC^2 are of particular interest since many important problems in NC are also in NC^2 [2] and some of them fall in NC^1. Unfortunately, the class NC^1 may not be the same under the log space uniformity and the U_E*-uniformity introduced by Ruzzo [4] but NC^k is stable for $k \geqq 2$. With respect to the recognition of sets, Ruzzo [4] characterized the class NC^k in terms of the time and space complexities of indexing alternating Turing machines as

$$NC^k = ATISP((\log n)^k, \log n),$$

where $ATISP((\log n)^k, \log n)$ is the class of sets accepted by indexing alternating Turing machines running in time $O((\log n)^k)$ and simultaneously in space $O(\log n)$.

This paper is concerned with a subclass of NC^1 and we deal only with sets instead of functions. The choice of a uniformity definition for NC^1 does not matter here. We introduce indexing alternating finite automata and establish a relationship with some kind of uniform boolean circuits in the same spirit as [4]. A conventional Turing machine reads a given word from left to right. Instead of scanning a word sequentially, a random accessing ability to the input tape is invented in [4] with the use of an *index tape*. A binary integer shall be written on the index tape. Then by a read instruction, the input symbol of the position specified by the binary integer on the index tape is accessed in one step. The indexing mechanism speeds up the input accessing.

Our main result is that the class of sets accepted by indexing alternating finite automata (denoted IAFA) coincides with the class of sets accepted by binary tree like

* Research Institute of Fundamental Information Science, Kyushu University 33, Fukuoka 812, Japan.

circuit families. A binary tree like circuit family consists of circuits of very regular forms similar to full binary trees. This yields that IAFA is a subclass of NC^1. It is not trivial from the definition of indexing alternating finite automata that IAFA contains all regular sets over $\{0, 1\}$. We show that it is the case. A non-regular set in IAFA and a set not in NC^1—IAFA are also shown. Thus IAFA is a proper subclass of NC^1 containing regular sets properly.

2. Definitions

Since a finite automaton is not allowed any worktape, it is difficult to write various binary integers on the index tape. In order to relax this difficulty, we introduce alternation [1] and the following writing method for the index tape. Given an input of length n, initially the index tape head is placed on the $(\lfloor \log(n-1) \rfloor + 1)$st square and the symbol $\sharp$ is on the 0th square as in Fig. 1, where for $n=0, 1$ its value is assume to be 0 and 1, respectively. It should be noted that $\lfloor \log(n-1) \rfloor + 1$ is the number of bits required to represent the number $n-1$ in binary. The index tape head writes 0

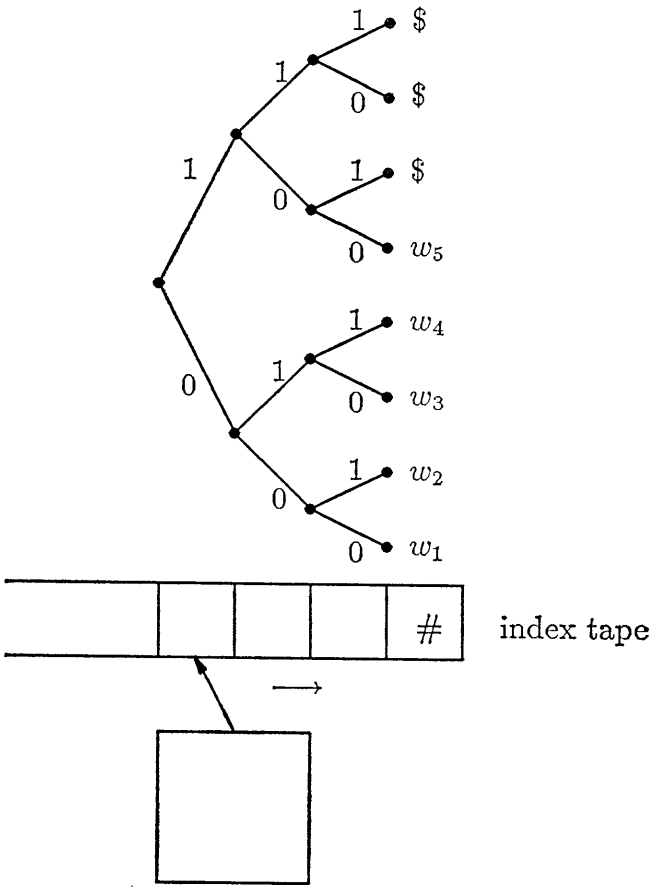


Fig. 1.

or 1 and moves to the right. When it reaches ♯, it can access the input symbol of the position $\nu(x)+1$, where x is the binary string on the index tape and $\nu(x)$ is the number it represents. For example, if the input length is 6 and the index tape contains 010♯, then the third input symbol is accessed. The position greater than n may be accessed. In such case we assume that a special symbol \$ is accessed.

DEFINITION 1. An *indexing alternating finite automaton* (abbreviated as *indxing afa*) M consists of the following data:

(1) There are two kinds of states.

(i) *Existential states*: $Q_\exists$ denotes the set of existential states.

(ii) *Universal states*: $Q_\forall$ denotes the set of universal states.

(2) There are special states.

(i) The *initial state* $q_1 \in Q_\exists \cup Q_\forall$.

(ii) The *read states* $r_1, \cdots, r_t \in Q_\exists$: The input position specified by the index tape contents is accessed if the index tape head is on the marker ♯. Q_R denotes the set of the read states.

(iii) The accepting state **accept** $\in Q_\exists$.

(iv) The rejecting state **reject** $\in Q_\exists$.

(3) There are three kinds of operations for the index tape and the input tape.

(i) *write*(x) $(x \in \{0, 1\})$: If the index tape head is scanning the blank symbol B, then it writes the symbol x and moves one square right.

(ii) A $(A \in \{0, 1, \$\})$: This operation is executable only when the current state is a read state and the index tape head is on the marker ♯. The operation A $(A \in \{0, 1\})$ checks if the input symbol accessed is A. The operation \$ checks if the input head position specified by the index tape contents is greater than the input length.

(iii) ε: This is a null operation. Nothing will be executed except state transitions. A transition with ε is called an *ε-move*.

Let $Q_T=(Q_\exists \cup Q_\forall)-\{r_1, \cdots, r_t,$ **accept, reject**$\}$. Then the *transition relation* δ is defined as a subset

$$\delta \subseteq Q_T \times \{write\ (0), write\ (1), \varepsilon\} \times (Q_T \cup Q_R)$$

$$\cup Q_R \times \{0, 1, \$\} \times \{\textbf{accept, reject}\}.$$

Informally, an indexing afa M on an input w moves in the following way: The computation starts with the initial state. When it is in an existential state, it chooses a transition nondeterministically and moves to the next state. When it is in a universal state, it branches into several states according to the transitions involving the current universal state. During these movements, it writes 0's and 1's on the index tape following the write operations specified in the transitions. After the index tape is filled up, it enters a read state where it reads the input symbol of the position specified by the contents of the index tape as mentioned above. Then it transits into the accepting or rejecting state. When the position specified by the index tape contents is greater than the input length, it can also change its state to the accepting or rejecting state. If all branching paths starting at the initial state reach the accepting state, it accepts the input w.

DEFINITION 2. Let M be an indexing afa. We say that an input $w \in \{0, 1\}^*$ is

accepted by M if there is a finite tree called an *accepting tree* satisfying the following conditions:

(1) Each vertex is labelled with a state of M and each edge is labelled with an operation of M. In particular, the root is labelled with the initial state and each leaf is labelled with **accept.**

(2) If an internal vertex u is labelled with $q \in Q_\exists$, then u has exactly one child v labelled with p such that $(q, \varphi, p) \in \delta$ and the edge (u, v) is labelled with φ.

(3) If an internal vertex u is labelled with $q \in Q_\forall$, then, for each $(q, \varphi, p) \in \delta$, u has a child v labelled with p and the edge (u, v) is labelled with φ.

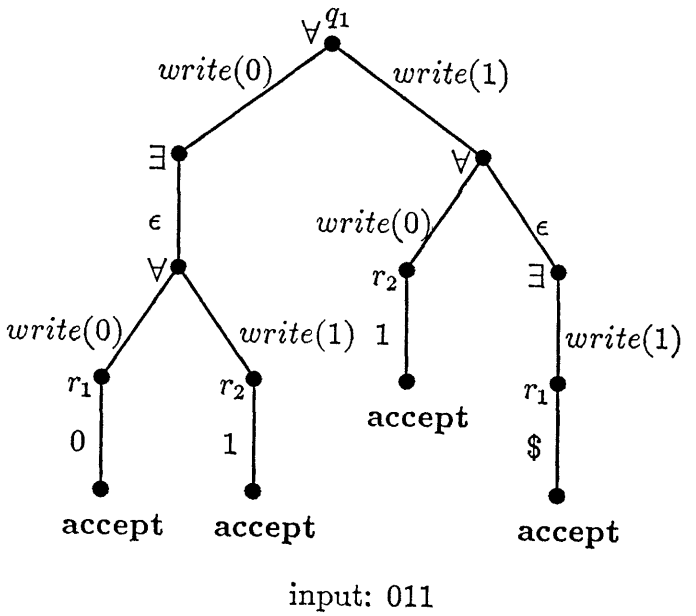
(4) A vertex u adjacent to a leaf is labelled with a read state. The vertex u has exactly one child v labelled with **accept.** As to the label of the edge (u, v), we consider the path from the root to the vertex u. Note that the edges on this path are labelled with *write*(0), *write*(1) or ε. Let x be the binary string written on the index tape by this sequence of write operations. If the input symbol of the $(\nu(x)+1)$st position is A, then the label of the edge (u, v) is A for $A \in \{0, 1\}$. If $\nu(x)+1$ is greater than the input length, the label is \$ (see Fig. 2).

DEFINITION 3. For an indexing afa M, the set of words accepted by M is denoted by $L(M)$. Then we define the class IAFA as follows:

$$\text{IAFA} = \{L \mid L \subseteq \{0, 1\}^*,\ L = L(M) \text{ for some indexing afa } M\}.$$

REMARK. Without loss of generality, we may assume the following restrictions (1)-(5) on indexing afa's:

(1) Each $w \in L(M)$ is accepted with an accepting tree such that the maximum



input: 011

Fig. 2.

length of contiguous ε-moves in the tree is bounded by the number of states in M. Hence the height of the accepting tree is $O(\log n)$.

(2) For each state $q \in Q_\exists \cup Q_\forall$, $|\{p|(q, \varphi, p) \in \delta\}| \leqq 2$.

(3) Moreover, for each state q, the operation $write(x)$ satisfies $|\{p|(q, write(x), p) \in \delta\}| \leqq 1$ and $|\{p|(p, write(x), q) \in \delta\}| \leqq 1$ for each $x \in \{0, 1\}$.

(4) A read state r is accessible from Q_T only via an ε-move $(q, \varepsilon, r) \in \delta$. Furthermore, we assume that for each $q \in Q_T$, $|\{r|r \in Q_R, (q, \varepsilon, r) \in \delta\}| \leqq 1$ and for each $r \in Q_R$, $|\{q|q \in Q_T, (q, \varepsilon, r) \in \delta\}| \leqq 1$.

(5) For each read state r and each $A \in \{0, 1, \$\}$, there exists a unique state f in {**accept, reject**} with $(r, A, f) \in \delta$.

DEFINITION 4. Let $\{\alpha_n\}_{n \geqq 0}$ be a circuit family such that each α_n has n inputs $w_1, \cdots, w_n$ and one output z. We say that a set $L \subseteq \{0, 1\}^*$ is accepted by a circuit family $\{\alpha_n\}_{n \geqq 0}$ if the output of α_n with inputs $w_1, \cdots, w_n$ is $1 \Leftrightarrow w_1 \cdots w_n \in L$ for all $w_1, \cdots, w_n \in \{0, 1\}$ and $n \geqq 0$.

DEFINITION 5. A *binary tree like circuit family* $\{\alpha_n\}_{n \geqq 0}$ is defined using three basic constant circuits in Fig. 3 which are independent of n. The lines $u_1, \cdots, u_k$ (resp. $v_1, \cdots, v_m$) are called the *left inputs* (resp. *right inputs*). The lines $x_1, \cdots, x_m$ (resp. $y_1, \cdots, y_k$; z) are called the *left outputs* (resp. *right outputs; center output*).

α_0 is either 1 or 0. For $n \geqq 1$, let m be the integer such that $2^{m-1} < n \leqq 2^m$. Consider the full binary tree T_m with 2^m leaves. Assume that the leaves of T_m are numbered from left to right as $1, \cdots, 2^m$. Then the tree $T[n]$ denotes the subtree of T_m with leaves $1, \cdots, n$. The binary tree like circuit α_n is defined as follows: First we assign the circuit C_2 (resp. C_1) to the internal vertices of $T[n]$ with two children (resp. one child). The circuit I is assigned to each leaf. Then we connect these circuits by identifying the left (resp. right) inputs of a circuit with the right (resp. left) outputs of the circuit left below (resp. right below), where the center output is ignored. For the circuit locating at the root, the center output is used while the left and right outputs are ignored (see Fig. 4).

DEFINITION 6. We define the class BC as follows:

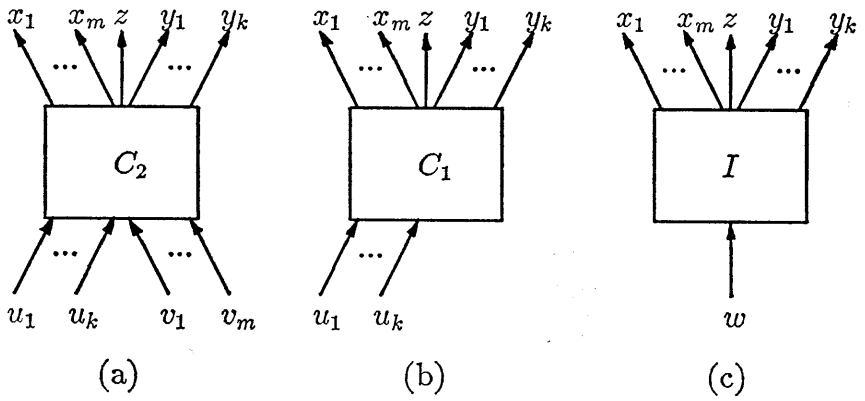


Fig. 3.

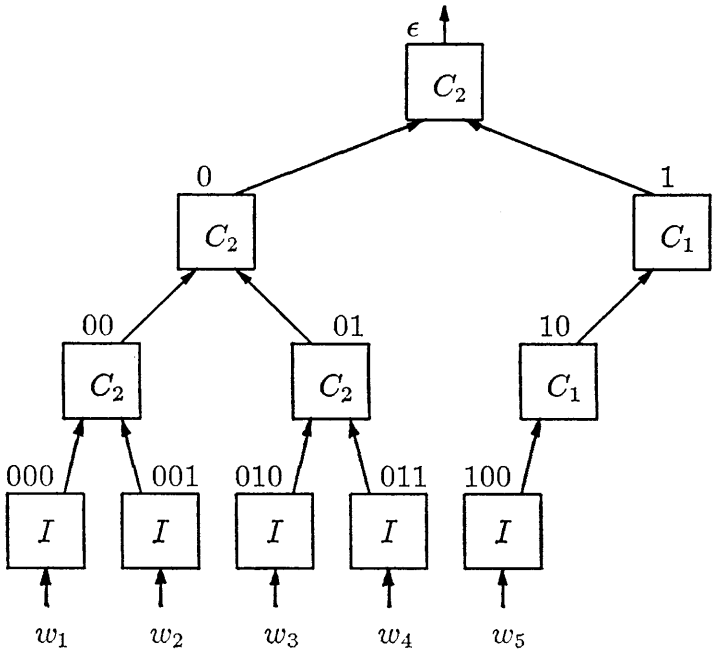


Fig. 4.

BC=$\{L \mid L \subseteq \{0, 1\}^*$ is accepted by a binary tree like circuit family$\}$.

3. Characterization of IAFA by Circuits

THEOREM 1. *BC=IAFA.*

PROOF. BC⊆IAFA: We give an informal description of an indexing afa which simulates a binary tree like circuit family with basic constant circuits C_2, C_1 and I. We explain how the circuit in Fig. 4 is simulated by an indexing afa M. Initially, it guesses the output bit z. Then it guesses which circuit of C_2, C_1, I locates at the position ε of Fig. 4. If C_2 is guessed, it also guesses the left and right input bit vectors U and V for C_2. Then it checks if z is the output of C_2 with inputs U and V. After the pair (U, V) is determined, it universally branches into two states corresponding to U and V executing the operations *write*(0) and *write*(1), respectively (Fig. 5(a)). Simultaneously, it passes to the state for V the information that it is, at present, in the right most position. For the position not right most, it must guess either C_2 or I.

When C_2 is guessed for the position 0, the left and right input bits are guessed and checked in the same way.

At the position 1, which is now right most, if C_1 is guessed, then the left input bit vector U' is guessed and checked in the same way. Then it universally branches into the state for U' and the special state q_S executing *write*(0) and *write*(1), respectively. Once the state q_S is reached, it universally branches into two parts with the same state q_S executing the operations *write*(0) and *write*(1), respectively (Fig. 5(b)). From

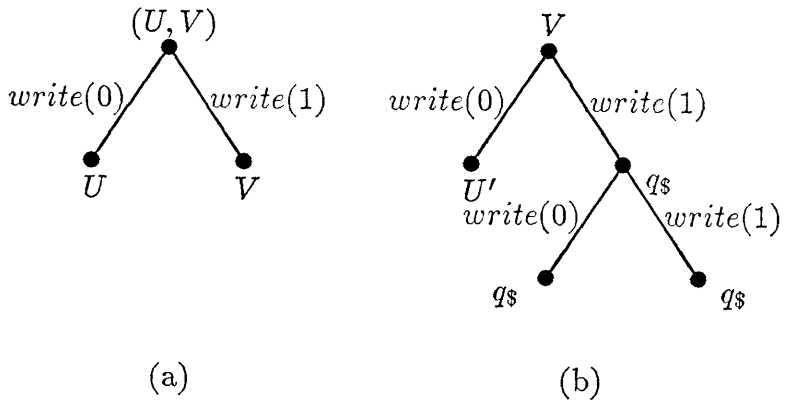


Fig. 5.

the state $q_{\$}$, it is also possible to enter a read state. When it reached a read state, it is allowed to enter the accepting state if and only if the position specified by the index tape contents is greater than the input length.

When the circuit I is guessed, it guesses the input symbol and checks the consistency with the bit vector determined before. Then it accesses the input tape using the index tape contents, if possible, and verifies that the guessed input symbol coincides with the accessed input symbol.

IAFA$\subseteq$BC: Given an indexing afa M, we define three basic circuits $C_2(M)$, $C_1(M)$ and $I(M)$ in the following way: Assume that M has states $q_1, \cdots, q_s, r_1, \cdots, r_t$, **accept, reject,** where $r_1, \cdots, r_t$ are read states.

We first define the circuit $C_2(M)$ (see Fig. 6). The circuit $C_2(M)$ has gates $q_i^{(k)}$ and $p_i^{(k)}$ for $i, k=1, \cdots, s$, where $q_i^{(k)}$ is an AND-gate (resp. OR-gate) if q_i is a universal (resp. existential) state, and all $p_i^{(k)}$ are OR-gates. For $k=2, \cdots, s$, the OR-gate $p_i^{(k)}$ has the inputs from $q_i^{(k)}$ and $p_i^{(k-1)}$ and $p_i^{(1)}$ has an input from $q_i^{(1)}$ for $i=1, \cdots, s$. The gate $q_i^{(k)}$ has an input from the gate $p_j^{(k-1)}$ for each $q_j \in \{q_j | (q_i, \varepsilon, q_j) \in \delta, 1 \leqq j \leqq s\}$. By Remark (1), such s stage construction suffices to simulate contiguous ε-moves. Remark (2) guarantees that the fan-in of the circuit is at most 2. The inputs and outputs of $C_2(M)$ are defined as follows. For each transition $(q_i, write(0), q_j) \in \delta$, a left input comes into $q_i^{(1)}$ from the gate $p_j^{(s)}$ of the circuit left below and a right output goes out from $p_j^{(s)}$ to the gate $q_i^{(1)}$ of the circuit right above. Similarly, for each transition $(q_j, write(1), q_i) \in \delta$, a right input comes into $q_j^{(1)}$ from the gate $p_i^{(s)}$ of the circuit right below and a left output goes out from $p_i^{(s)}$ to the gate $q_j^{(1)}$ of the circuit left above. By Remark (3), the correspondence between inputs and outputs is unique. The center output goes out from $p_1^{(s)}$.

To define $C_1(M)$, we use a constant circuit $C_{\$}(M)$ with no input. $C_{\$}(M)$ has the same structure as that of $C_2(M)$ except the inputs. By Remark (4), we can associate each read state r_k with the unique state q_{j_k} satisfying $(q_{j_k}, \varepsilon, r_k) \in \delta$. If $(r_k, \$, \textbf{accept}) \in \delta$, then $q_{j_k}^{(1)}$ gets the value 1. Similarly, if $(r_k, \$, \textbf{reject}) \in \delta$, then $q_{j_k}^{(1)}$ gets the value 0. By Remark (5), these are well defined. The construction of $C_1(M)$ is similar to $C_2(M)$. It does not have the right inputs but instead of them the circuit $C_{\$}(M)$ is

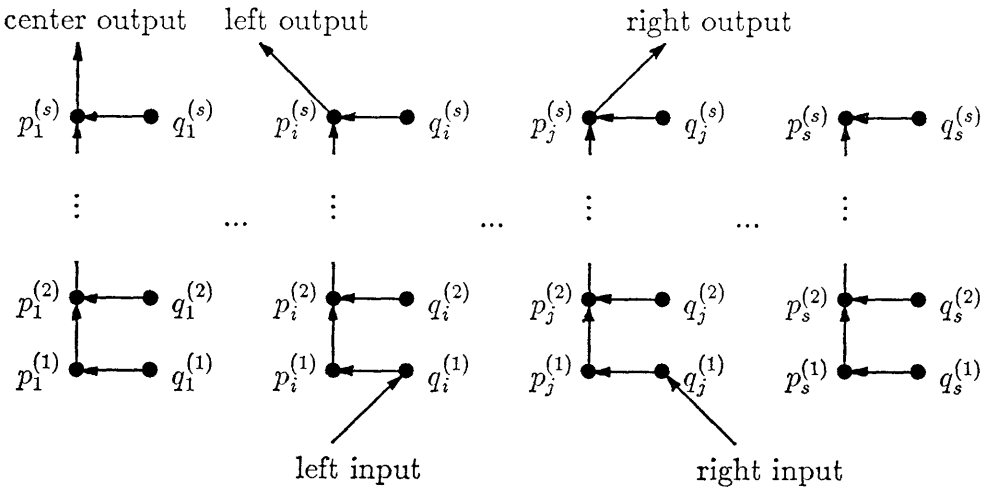


Fig. 6.

attached so that $q_i^{(1)}$ has an input from $p_j^{(s)}$ of $C_{\$}(M)$ for each transition $(q_i, write(1), q_j) \in \delta$.

Finally, we define the circuit $I(M)$. It also has the same structure as that of $C_2(M)$ except the inputs. $I(M)$ has the unique input w. Recall that a read state r_k can be accessible only via an ε-move $(q_{j_k}, \varepsilon, r_k)$. Then we connect the input w to q_{j_k} $(k=1, \cdots, t)$ in the following way: If $(r_k, 1, \textbf{accept}) \in \delta$ and $(r_k, 0, \textbf{reject}) \in \delta$, then w directly goes to $q_{j_k}^{(1)}$. If $(r_k, 1, \textbf{reject}) \in \delta$ and $(r_k, 0, \textbf{accept}) \in \delta$, then w is first negated and goes to $q_{j_k}^{(1)}$. If $(r_k, x, \textbf{reject}) \in \delta$ (resp. $(x_k, x, \textbf{reject}) \in \delta$) for all $x \in \{0, 1\}$, then the value 1 (resp. 0) goes to $q_{j_k}^{(1)}$. The well-definedness follows from Remark (3). This completes the proof. □

In [5] it is proved that any regular set can be accepted by an indexing alternating Turing machine in time $O(\log n)$. But it is not obvious that all regular sets over $\{0, 1\}$ are in IAFA. By the following proposition combined with Theorem 1, we can see that every regular set over $\{0, 1\}$ is accepted by an indexing afa and therefore in time $O(\log n)$ (Remark (1)).

PROPOSITION 2. *Let $L \subseteq \{0, 1\}^*$ be a regular set. Then L is accepted by a binary tree like cirduit family.*

PROOF. We may assume that L is accepted by a nondeterministic finite automaton $M=(Q, \{0, 1\}, q_1, \delta, \{q_s\})$ with a single accepting state q_s, where q_1 is the initial state and δ is the transition relation. Let $q_1, \cdots, q_s$ be the states of M. We construct the basic circuits C_2, C_1 and I as follows:

(1) The circuit C_2 is shown in Fig. 7(a). The pairs $(q_1, q_1)^x, \cdots, (q_i, q_j)^x, \cdots, (q_s, q_s)^x$ are boolean variables, where $x \in \{l, r\}$. It means that $(q_i, q_j)=1 \Leftrightarrow$ there is a sequence of transitions from q_i to q_j. The left and right outputs are the same. The center output is (q_1, q_s). In the circuit D_2, $(q_i, q_j) \leftarrow \Sigma_{k=1}^{s}(q_i, q_k)^l \cdot (q_k, q_j)^r$ is computed using the left and right inputs.

(2) The circuit I is illustrated in Fig. 7(b). If $(q_i, 1, q_j) \in \delta$, then the input w directly goes to the output (q_i, q_j). If $(q_k, 0, q_l) \in \delta$, then the input is first negated and

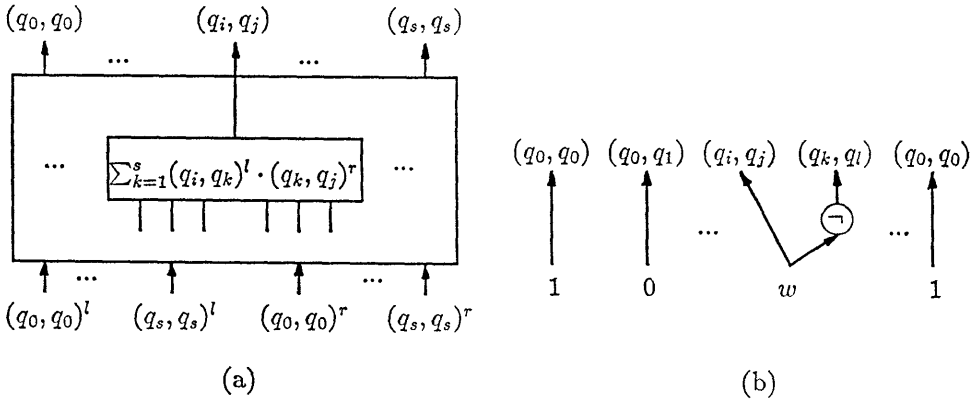


Fig. 7.

goes to the output (q_k, q_l). The output (q_i, q_i) gets the value 1. The other outputs get the value 0.

(3) The circuit C_1 is the identity circuit that just transfers the inputs to the outputs.

Then it is easy to see that the binary tree like circuit family with these basic circuits computes the transitive closure of the state transitions under given inputs. □

PROPOSITION 3. *The set* $L=\{0^n1^n \mid n\geqq 1\}$ *is not in IAFA.*

PROOF. Assume that L_0 is accepted by a binary tree like circuit family $\{\alpha_n\}_{n\geqq 0}$ with basic circuits C_2, C_1 and I. Let $m\geqq 2$. We consider the words 0^n1^n for n with $2^{m-2}+1\leqq n\leqq 2^{m-1}$ that must be accepted by α_{2n}. Notice that the height of $T[2n]$ is the same as that of $T[2^m]$ for all $2^{m-2}+1\leqq n\leqq 2^{m-1}$. We concentrate on the basic circuit C_2 locating at the root. Let l_n and r_n be the left and right input bit vectors of the circuit C_2 locating at the root of $T[2n]$ when the input for the circuit α_{2n} is 0^n1^n. Since the lengths of the bit vectors l_n and r_n are constants independent of the length of the input, $(l_i, r_i)=(l_j, r_j)$ for some $2^{m-2}+1\leqq i<j\leqq 2^{m-1}$ if m is sufficiently large. Then by the structure of the circuits α_{2n} for $2^{m-2}+1\leqq n\leqq 2^{m-1}$ we see that the word 0^i1^j is also accepted by the circuit α_{i+j}. This is a contradiction. □

In contrast with Proposition 3, the following result can be easily shown. Hence IAFA contains a non-regular set.

PROPOSITION 4. *The set* $L_1=\{0^n1^n \mid n=2^m$ *for some* $m\geqq 0\}$ *is in IAFA.*

Acknowledgement

The author would like to thank Alexander von Humboldt-Stiftung for the support during this work.

References

[1] CHANDRA, A.K., KOZEN, D.C. and STOCKMEYER, L.J.: *Alternation*, J. Assoc. Comput. Mach., **28** (1981), 114-133.

[2] COOK, S.A.: *A taxonomy of problems with fast parallel algorithms*, Inform. Contr., **64** (1985), 2-22.

[3] PIPPENGER, N.: *On simultaneous resource bounds (preliminary version)*, Proc. 20th IEEE Symposium on Foundations of Computer Science, (1979), 307-311.
[4] RUZZO, W.L.: *On uniform circuit complexity*, J. Comput. System Sci., **22** (1981), 365-383.
[5] STOCKMEYER, L.J. and VISHKIN, U.: *Simulation of parallel random access machines by circuits*, SIAM J. Comput., **13** (1984), 409-422.
[6] TOMPA, M.: *An extension of Savitch's theorem to small space bounds*, Inf. Process. Lett., **12** (1981), 106-108.

Received October 27, 1987