数理解析研究所講究録
353巻 1979年 204-213

Some properties of bottom-up cellular pyramids

(extended abstract)

Charles R. DYER[1)]
Akira NAKAMURA[2)]
Azriel ROSENFELD[1)]

[1)] Computer Science Center
University of Maryland
[2)] Department of Applied Mathematics
Hiroshima University

1. Introduction

Cellular pyramids have been introduced by Rosenfeld and Dyer [1,2] as interesting parallel pattern recognition devices. Bottom-up pyramid acceptors have been defined as cellular pyramids restricted in information transmission, and their properties and capabilities were extensively studied in [1-5]. In these papers, it is shown that many basic image processing and analysis tasks can be implemented very efficiently using these cellular acceptors. However, besides problems involving image-specific tasks, there are also interesting guestions about pyramid acceptors from the point of view of formal language theory. We have already established an important result [4] along this line, namely that the class of sets accepted by nondeterministic bottom-up pyramid acceptors is equivalent to that accepted by nondeterministic bounded cellular acceptors.

In this paper, we examine the formal language recognition capabilities of bottom-up cellular pyramids. The accepting powers of bottom-up pyramid acceptors and bounded cellular acceptors are compared. The main results are as follows:

(1) 2-dimensional deterministic bounded cellular acceptors are stronger than deterministic bottom-up pyramid acceptors (DBPA < 2-DBCA).

(2) 1-dimensional deterministic bounded cellular acceptors are stronger than deterministic bottom-up triangle acceptors (DBTA < 1-DBCA).

As corollaries of (1) and (2), we show that

(3) Nondeterministic bottom-up pyramid acceptors are stronger than deterministic ones (DBPA < NBPA).

(4) Nondeterministic bottom-up triangle acceptors are stronger than deterministic ones (DBTA < NBTA).

2. DBPA < 2-DBCA

In this section, we prove that the class of languages accepted by 2-DBCA's properly contains that accepted by DBPA's. (In brief, we say that 2-DBCA's are stronger that DBPA's or that DBPA < 2-DBCA.)

Theorem 2.1 2-DBCA's are stronger than DBPA's.

Proof:

Let $\not\emptyset$ be a special blank symbol. Given an m^2-bounded deterministic Turing machine (DTM) M which has input of the form: $\#x_1x_2\ldots x_m\underbrace{\not\emptyset\not\emptyset\ \ldots\ \not\emptyset}_{m(m-1)}\#$ where $m=2^i$ for some positive integer i, we can easily define a 2-DBCA C which simulates M, since the input can be "folded" into an m-by-m block.

Let L_{DLBA} be the class of sets accepted by deterministic linear bounded acceptors. It has been shown [6] that

$$L_{DLBA} \subsetneq L_{n^2\text{-bounded DTM}} .$$

It is provable by the same considerations as in [6] that

$$L_{m\text{-bounded DTM}} \subsetneq L_{m^2\text{-bounded DTM}}$$

where m is the same as before.

Since a triangle cellular acceptor can be simulated by a 1-DBCA [2], clearly $L_{DBTA} \subseteq L_{1\text{-DBCA}}$. We also know that $L_{1\text{-DBCA}} = L_{DLBA}$ [7]. Therefore, if we can show that a DBPA with input of the form:

$$\begin{array}{c} x_1 x_2 \ \cdots \ x_m \\ \not\emptyset\ \not\emptyset \ \ \cdots \ \not\emptyset \\ \vdots \qquad \vdots \\ \not\emptyset\ \not\emptyset \ \ \cdots \ \not\emptyset \end{array}$$

can be simulated by a DBTA with input $x_1 x_2 \ldots x_m$, then for this kind of input

$$L_{DBPA} \subseteq L_{DLBA}$$

This will prove that

$$L_{DBPA} \subseteq L_{DLBA} \subsetneq L_{m^2\text{-bounded DTM}} \subseteq L_{2\text{-DBCA}} .$$

We will now show that a DBPA with input of the form

$$\begin{array}{c} x_1 x_2 \ \cdots \ x_m \\ \not\emptyset\ \not\emptyset \ \ \cdots \ \not\emptyset \\ \vdots \qquad \vdots \\ \not\emptyset\ \not\emptyset \ \ \cdots \ \not\emptyset \end{array}$$

can be simulated by a DBTA with input $x_1 x_2 \ldots x_m$.

Given DBPA A with transition funtion δ, construct a DBTA B which acts as follows:

1. At time t = 1 all cells compute a state pair (q_1, q_2) such that

   if cell c is at level 0 in state s, at t = 0

   then define $\begin{cases} q_1 = s \\ q_2 = \not\beta \end{cases}$

   else cell c is at level k > 0

   in the quiescent state $\natural$, so

   define $\begin{cases} q_1 = \natural \\ q_2 = \natural \end{cases}$

   fi

2. At time t > 1

   if cell c is at level 0

   then $\begin{cases} q_1' = \delta(q_1,\#,\#,\#,\#) \\ q_2' = \delta(q_2,\#,\#,\#,\#) \end{cases}$

   else

   begin

   let (r_1, r_2) be the state pair of c's left son at time t-1

   (s_1, s_2) be the state pair of c's right son at time t-1

   $q_1' = \delta(q_1, r_1, s_1, r_2, r_2)$

   $q_1' = \delta(q_2, r_2, r_2, r_2, r_2)$

   end

   fi

Intuitively, A has cells of only two types: either a cell has a descendant in the top row of the input (call this cell type 1), or it doesn't (call this cell type 2). Cell type 1 applies transitions of the form $\delta(p_1,p_2,p_3,p_4,p_4)$ only, since the cell's southwest and southeast sons have identical inputs (all blanks). Cell type 2 applies transitions of the form $\delta(p_1,p_2,p_2,p_2,p_2)$ only, since all the cell's sons have identical inputs (all blanks).

Thus a cell at level k in B can simulate (with its q_2-state) the sequence of states that a type 2 cell at level k in A would be making. In addition, a cell at level k in B can simulate (with its q_1-state) the sequence of states that a type 1 cell at level k in B would be making, since its north sons are type 1 cells and have the necessary state information in their q_1-states (which are identical). [This can be proved more rigorously by induction, if desired.] Thus a DBTA can simulate a DBPA with input of the given form. //

Corollary 2.2 NBPA's are stronger than DBPA's.

Proof:

In [4], we proved that an NBPA can simulate a 2-DBCA. Therefore, we get this corollary from Theorem 2.1 . //

3. DBTA < 1-DBCA

In this section, we prove that DBTA < 1-DBCA.

Let S be the set $\{2^n \mid n \text{ is a nonnegative integer}\}$, let f be a function from S into S such that $f(2^n) \geq 2^n$, and let M be a DBTA with a special blank state symbol $\not\emptyset$ where $\not\emptyset$ is as before. Also, let L be a set of strings over the input state $Q_T-\{\#,\not\emptyset\}$. If M accepts the language $L'=\{\sigma\not\emptyset^m \mid \sigma \in L,$

$m=f(|\sigma|)-|\sigma|\}$ where $|\sigma|$ means the length of the string σ, then we say that M accepts L with $f(2^n)$ cells or that M is a $f(2^n)$-bounded DBTA. In particular, if f is the identity function, i.e. m=0, then M is called a linear-bounded DBTA. Up to now we have exclusively studied linear-bounded (D or N) BTA's. We now investigate the power of $(2^n)^2$-bounded DBTA's, where 2^n is the length of the input string.

Lemma 3.1 A detherministic linear-bounded automaton can simulate a $(2^n)^2$-bounded DBTA.

Proof:

The initial configuration in a $(2^n)^2$-bounded DBTA — i.e., 2^{2n}-bounded DBTA —— has the form shown in Figure 1.

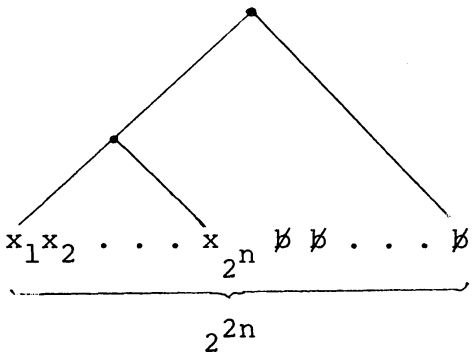

Figure 1

While there are $2\cdot 2^{2n}-1$ cells in a 2^{2n}-bounded DBTA, many of these cells' subtrees have identical initial configurations since most of the base is initially in the blank state. Clearly any two cells whose initial subtree configurations are identical must have identical state sequences. For example, in level 0 there are $m=2^{2n}-2^n$ cells with the initial subtree configuration  . Since all of these cells have identical state sequences, there are at most 2^n+1 distinct level 0 cells, namely 2^n non-ø-cells and one ø-cell. (Actually, there

are at most $|Q|$ distinct non-$\emptyset$-cells, but because ancestor cells may have distinct subtrees, we will save these redundant cells in order to simplify the computation of higher level cells' state sequences.) In general, at level k, $0 \leq k \leq n$, there are $2^{2n}/2^k$ cells. Of these $2^n/2^k$ cells have non-blank bases, and the rest have identical all-blank base segments. Hence there are at most $(2^n/2^k)+1$ distinct cells in level k. At each of levels n+1 through 2n there is exactly one cell per level with non-blank input. Hence there are only two distinct state sequences for cells at any one of these levels. Summing, we find that there are at most $2 \cdot 2^n+3 \cdot \log 2^n - 1$ distinct cell types in any 2^{2n}-bounded DBTA. Thus, while there are $O(2^{2n})$ cells in a 2^{2n}-bounded DBTA, there are only at most $O(2^n)$ which have distinct state sequences.

We now show how a deterministic linear-bounded automaton can simulate a 2^{2n}-bounded DBTA, M, by making use of the fact that many cells have identical state sequences and hence all of them need not be explicitly stored. In [2] it was shown how a 1-DBCA can simulate a DBTA after using a breadth-first ordering to linearize the DBTA cells. These cells' states were then stored two per cell in the 1-DBCA, and cell indexing functions were defined for accessing the states of a cell's father, brothers, and sons. In a similar manner, we can order the $2 \cdot 2^n-1$ cells of M with non-blank bases breadth-first and then map them into the 2^n squares of a DLBA's input tape, two states per tape square. Since a DLBA can simulate a 1-DBCA [7], the cell indexing functions for accessing both sons' state (described in [2]) can be used here. The remainder of

the DBTA cells to be saved are of two types: $2\log 2^n$ "blank cells", i.e., cells with all-blank input, one cell from each level 0 through $2 \cdot \log 2^n - 1$ in the DBTA; and $\log 2^n$ "non-blank cells" with non-blank input, one cell in each of levels $\log(2^n+1)$ through $2\log 2^n$. Let us map the $2\log 2^n$ blank cells into the leftmost $2\log 2^n$ squares of the tape so that the cell representing level k is stored in the kth square from the left end. Similarly, let us store the $\log 2^n$ non-blank cells in squares $(\log 2^n)+1$ through $2\log 2^n$ of the tape.

A DLBA can easily mark off blocks of tape of size $\log 2^n$ by counting the number of input squares. Hence it can easily find those sections of tape which contain these extra $3\log 2^n$ cells' states. At any time step the states of both sons of a blank cell are the same and are stored in the square to the left of the square associated with the given blank cell. The states of the sons of non-blank cells are also located in the squares immediately to the left of the given square. (There is one exception —— the cell in square $(\log 2^n)+1$ has as its left son the root of the subtree containing the input string $x_1 x_2 \ldots x_{2^n}$. This state is located in cell 1 as described earlier.) Thus a DLBA simulates a single step of M by sequentially accessing the states of the two sons for each of the $2 \cdot 2^n + 3 \cdot \log 2^n - 1$ cells stored therein. In particular, the state of the root of M is stored in tape square $2\log 2^n$, which the DLBA can find at the completion of each simulation step. If that cell is in an accepting state, then the DLBA also enters an accepting state. Hence a DLBA can can simulate a $(2^n)^2$-bounded DBTA. //

Theorem 3.2 1-DBCA's are stronger than DBTA's.

Proof:

In [2] we proved that a 1-DBCA can simulate a DBTA. We now show that the converse is not true. Trivially, a 1-DBCA with f(n) cells can simluate a f(n)-bounded Turing acceptor by keeping track of the movement of the input and storage tape heads. From [6] it is known that there exists a language accepted by an n^2-bounded Turing acceptor, but not by any n-bounded Turing acceptor. Thus by these results and the previous lemma we can immediately conclude that there exists a language accepted by a 1-DBCA with n^2 cells, but not by any n^2-bounded DBTA. //

Corollary 3.3 NBTA's are stronger than DBTA's.

Proof:

In [4] we showed how an NBTA can simulate a 1-DBCA. In particular, this means that an n^2-bounded NBTA can simulate a 1-DBCA with n^2 cells. This corollary now immediately follows from Theorem 3.2. //

Acknowledgment

A. Nakamura, one of the authors, would like to thank his research assistant Dr. T. Watanabe for valuable discussions.

References

[1] A Rosenfeld: Picture languages, forthcoming book.

[2] C. R. Dyer and A. Rosenfeld: Cellular pyramids for image analysis, TR-544, Computer Science Center, University of Maryland, College Park, Md., May, 1977.

[3] C. R. Dyer: Cellular pyramids for image analysis, 2, TR-596, Computer Science Center, University of Maryland, College Park, Md., Nov., 1977.

[4] A. Nakamura and C. R. Dyer: Nondeterministic bottom-up pyramid acceptors, TR-616, Computer Science Center, University of Maryland, College Park, Md., Dec. 1977.

[5] A. Nakamura and C. R. Dyer: Bottom-up cellular pyramids for image analysis, Proc.of 4th Inter. Joint Conference on Pattern Recognition, Nov. 7-10, 1978, Kyoto, 494-496.

[6] R. E. Stearns, J. Hartmanis and P. M. Lewis II: Hierarchies of memory limited computations, Proc. 6th SWAT, 1965, 179-190.

[7] A. R. Smith III: Cellular automata and formal languages, Proc. 11th SWAT, 1970, 216-224.