Minimum Broadcast Trees

ANDRZEJ PROSKUROWSKI

Abstract—**"Broadcasting" is an information dissemination process in which a member of a system generates a message communicated to all other members. We model this process by ordered rooted trees and investigate a special class of rooted trees allowing broadcasting from the root to all other vertices of the tree in the minimum time (over all rooted trees with *n* vertices). We characterize trees from this class ("mbt") and give an algorithm deciding membership in the class. We also present an algorithm to construct all *mbt*'s with a given number of vertices and give a recursive formula to count these trees.**

Index Terms—**Broadcasting, information dissemination, tree generation.**

I. Introduction

We investigate some ways in which information processing systems (like computer networks, business, or military organizations) may disseminate messages among its members. We model the structure underlying such systems by graphs, with members of a system represented by vertices and communication channels–by edges of the graph. In particular, we examine an information dissemination process called "broadcasting." In this process one member of a system generates a message which is communicated through the existing channels to all other members. To achieve that, each "informed" member makes a number of "calls" to some of its neighbors (in the sense of adjacency through a communication line). We choose to limit our model to synchronous calls where a vertex may be involved in, at most, one call at a time. The rationale for the model follows from recent results which indicate that: 1) disseminating a message to all the neighbors at one time unit does not (asymptotically) speed up the global dissemination process [4], and 2) the information dissemination is best done by the coordinated use of local interconnections, rather than dependence on a global broadcasting facility [8]. An "uninformed" member of the system may be called more than once in a redundant broadcast, which is likely to create an unnecessarily large number of calls (although measures exist to prevent network flooding [2]). A broadcasting in which each member of the system is called exactly once defines a spanning tree of the system's graph. This motivates our investigation of broadcasting in trees: what is the minimum time to complete broadcasting in a given tree, and which trees allow a broadcasting of minimum time over all trees with a given number of vertices. For general graphs, establishing the existence of a minimum time broadcast is computationally a very difficult task. It has been shown that the problem belongs to the class of NP-complete problems [7], which gives little hope for an efficient (polynomial time) solution algorithm. We show that limited to trees, the broadcast problem is solvable in linear time.

Until recently, not much has been published on the subject of broadcasting. Slater, Cockayne, and Hedetniemi [16] have designed an algorithm producing the set of vertices from which an optimal broadcast is possible in a given tree. Farley [3], Hedetniemi and Mitchell [10], and Farley, Hedetniemi, Mitchell, and Proskurowski [5] have studied broadcasting in graphs and discussed uniformly minimum broadcast time graphs (where the originator may be any of the graphs's vertices). More attention, although not in much depth, has been devoted to the process of "gossiping," which can be viewed as simultaneous broadcasting: each vertex of a graph is a source of a message which has to be communicated to all other vertices (Baker and Shostak [1], Hajnal *et al.* [9], Knödel [12], and Farley and Proskurowski [6]). Even here, the results on broadcast in trees imply lower bounds on the minimum time of completion of the information dissemination process.

Manuscript received March 3, 1979; revised October 19, 1979 and December 11, 1980. This work was supported in part by the National Science Foundation under Grant NSF-ENG 790296.

The author is with the Department of Computer and Information Science, University of Oregon, Eugene, OR 97403.

II. Definitions

A *free tree* $T = (V, E)$, is a connected, undirected graph with the set of n vertices V and $n - 1$ edges E. Often a free tree is represented by a rooted tree T_r, obtained by specifying a vertex $r \in V$ as its root. The choice of r induces direction on the edges of T_r: each edge connects a *father* and a *son* vertices, where father is the first vertex on the unique path from the son vertex to the root. For any vertex $v \neq r$, the subtree T_v^r of tree T_r is rooted at v and consists of v and all its descendants. All vertices without descendants are called *leaves* of T (the set of leaves of T is identical with the set of *end-vertices* of T, with the possible exception of the root r). The *height* of a tree T_r is defined as the maximum distance from the root r to any leaf of T_r. For algorithmic purposes, it is often very convenient to represent T_r by its *recursive representation* (Mitchell [14]). The vertices of T_r are labeled with integers $1, \cdots, n$ in such a way that the root r is labeled 1 and every set of vertices labeled $1, \cdots, i$ $(2 \leq i \leq n)$ is connected. Then the recursive representation of T_r, TREE, is an array of integers such that TREE$[i]$ is the label of the father of the vertex labeled i $(2 \leq i \leq n)$ and TREE$[i] < i$.

Broadcasting in a rooted tree T_r will denote a process of communicating a message from the root r to all the vertices of the tree. This is accomplished by some sequence of discrete and synchronous *calls* from each vertex to its sons. A vertex may be involved in, at most, one call at a time. Vertex v is *informed* if there exists a path from r to v of calls made in chronologically ordered time instances. The minimum time in which all vertices of T_r may become informed is called the *minimum broadcast time* $t(T_r)$. For a vertex v, time instance in which it must become informed in order to inform all its descendants within $t(T_r)$ is called the *minimum level requirement* of v, $l(v)$. We have immediately $l(r) = 0$. The minimum value of $t(T_r)$ over all rooted trees T_r of n vertices is bound from below by $\lceil \log n \rceil$. This follows from the fact that the set of informed vertices may, at most, double in a time unit. We call this limiting value $t(n)$, the minimum broadcast time for a tree with n vertices. All trees T_r for which $t(T_r) = t(n)$ are called *minimum broadcast trees* (mbt's). A broadcast in a rooted tree T_r is completely defined by *call sequences* for all vertices of T_r: lists of sons in the order they are called from a given vertex, their common father. We can thus equate the notion of an ordered rooted tree $\overline{T}_r$, and that of a broadcast in the rooted tree T_r. If $r_1, \cdots, r_k$ are the sons of r ordered by r's call sequence and n is the size (number of vertices) of $\overline{T}_r$, the broadcast may be represented recursively by $n\overline{T}_{r_1}^r \cdots \overline{T}_{r_k}^r$. We will discuss a construction process of sets of (ordered) rooted trees. Given a family (sequence) of sets of (ordered) rooted trees $\{S_1, S_2, \cdots S_k\}$, we will denote by $S_1 S_2 \cdots S_k$ the set of trees constructed as follows. Pick one tree from each set (in the order indicated) and join their roots to a new vertex–the root of a new (ordered) rooted tree. Through this construction process we will be able to produce all mbt's for a given number of vertices.

Our goal is to characterize minimum broadcast trees, determine whether a given tree is an mbt, and give a procedure of generating mbt's for a given value of n.

III. Minimum Broadcast Time Algorithm

Given a free tree T and a vertex r originating a message, there are two parameters of a successful broadcasting: the minimum time of completion and call sequences realizing that time. Additional questions may be asked: would a shorter time broadcasting be possible if the message originated at another vertex? Would another tree on n vertices, rooted at r, allow a broadcasting within a shorter time (with other words, is T_r an mbt)? The problem of optimal rooting of a tree was answered by Slater *et al.* [15]; if r is in the "broadcast center" of T, then the minimum broadcast time for T may be realized by call sequences that can be deduced from the algorithm. If this time is greater than $t(n) = \lceil \log n \rceil$, then T is not an mbt.

We present below an algorithm giving the minimum broadcast time and the corresponding call sequences for a rooted tree T_r. The algorithm is based on processing the vertices of T_r in decreasing order of their minimum level requirements, establishing their father's calling sequence and level requirements.

Algorithm 1: Minimum time broadcast in a rooted tree.

Input: Rooted tree T_r of n vertices.

Output: Call sequences and $t(T_r)$.

Method: For each vertex v of T_r we establish its level requirement $l(v)$ and its call sequence, call (v). Vertices whose all sons have been processed are inserted into the appropriate level set $S(l)$. Vertices in the set of the highest level, hilev, are processed.

```
/*0) Initialize */ for each vertex $v$ do
                      begin $l(v) := t(n)$; call($v$) := nil end;
                      for level := $t(n)$ down to $t(n) - n$ do
                          $S$(level) := empty;
                      for each leaf $v$ of $T_r$ do
                          $S(t(n)) := S(t(n)) \cup \{v\}$;
                      hilev := $t(n)$;
                      $T := T_r$; /*$T$ is the tree of not yet
                      processed vertices */
/*1) Process vertices */ while $|T| > 1$ do
                          begin if $S$(hilev) = empty
                          then hilev := hilev − 1;
    /*1.1) choose vertex */       $v$ := member of $S$(hilev);
    /*1.2) update father */       $w$ := father of $v$;
                                  call ($w$) := add prefix $v$
                                  to call ($w$);
                                  $l(w)$ := if $l(w) \leq$ hilev
                                  then $l(w) - 1$ else hilev − 1;
    /*1.3) remove v*/             $T := T - \{v\}$;
                                  $S$(hilev) := $S$(hilev) − $\{v\}$;
    /*1.4) update level sets */ if $w$ leaf of $T$
                                  then $S(l(w)) := S(l(w))$
                                  $\cup \{w\}$
                                  end; /* of processing $v$*/
/*2) Output */ $t(T_r) := t(n) - l(r)$
               end; /* of Algorithm 1*/
```

Before we prove the correctness and linearity of the algorithm let us state the following observations.

Observation 1: The final level requirement of a vertex w, as computed by Algorithm 1, differs from the level requirements of its son v by not less than the ordinal of v's position on w's call sequence.

Observation 2: Level requirements of vertices in any call sequence constructed during the execution of Algorithm 1 are monotonically nondecreasing.

Theorem 1: Algorithm 1 correctly computes $t(T_r)$ in $O(n)$ time.

Proof: The vertices removed from T_r in the course of execution of loop 1 constitute a forest of rooted trees T_u^r, such that ordering of T_u^r by the call sequences of its vertices defines a minimum time broadcast, and $t(T_u^r) = t(n) - l(u)$. We will prove it by mathematical induction on the number of removed vertices.

This assertion is true for the initial phase of the algorithm, when the forest $T_r - T$ is empty. Let us assume that for some T, $1 < |T| \leq n$, the assertion holds and let v be the vertex chosen in step 1.1. v is a leaf of T and therefore, when removed from T it becomes a root of a subtree T_v^r of T_r whose all vertices have been removed. By Observation 1, the calling sequence of v enables it to call each son r_i in the ith time unit such that

$$i + (t(n) - l(r_i)) \leq t(n) - l(v).$$

As $t(n) - l(r_i) = t(T_{ri}^r)$ by the inductive assumption, this defines a broadcast in T_v^r of time $t(n) - l(v)$. Among v's sons there exists at least one, r_j, such that $l(r_j) - j = l(v)$: the last son of v for which else-clause was executed in step 1.2. Any broadcast requiring less time than $t(n) - l(v)$ would have to place r_j on an earlier position in v's calling sequence. This would cause some other son v_i, $i < j$, to be called at time j. By Observation 2, $l(r_i) < l(r_j)$ and hence, $j + t(n) - l(r_i) \geq t(n) - l(v)$ not yielding any better broadcast time. Thus, $t(T_v^r) = t(n) - l(v)$. At the termination of the algorithm, the only vertex remaining is r. Its removal gives $t(T_r) = t(n) - l(r)$ ($l(r) = 0$ iff T_r is an mbt).

The linearity of Algorithm 1 is assured by constant time implementation of operations "member," "father," "add," and set difference "−" in loop 1. Linked lists are suitable data structures for sets S, call, and T allowing such operators. □

The call sequences resulting from the application of Algorithm 1 to a given rooted tree T_r impose an ordering on sibling vertices. Thus, the corresponding broadcast may be viewed as an ordered rooted tree $\overline{T}_r$. It is well known that such trees are in one-to-one correspondence with binary trees ("natural correspondence," Knuth [11, p. 332]); the following observation was suggested by a referee.

Observation 3: The time of a given broadcast $\overline{T}_r$ is equal to the depth of the binary tree associated with $\overline{T}_r$ by the natural correspondence.

IV. Construction of MBT's

The results of the preceding section gives the following condition for existence of a broadcast of time t in a given rooted tree.

Lemma 1: Given a rooted tree T_r, there exists a broadcast with completion time t if and only if the sons r_i ($i = 1, \cdots, k \leq t$) of the root r can be ordered in such a way that the corresponding subtrees T_{rj}^r allow broadcast times $t-1, t-2, \cdots, t-k$, respectively (i.e., $t(T_{ri}^r) \leq t - i$).

Proof: ($\Leftarrow$) Let us assume that the subscripts of r_i reflect such an order. Then $(r_i, \cdots, r_k)$ may serve as a calling scheme r; this, together with the assumed broadcasts in T_{ri}^r, define a broadcast in T_r requiring, at most, t time units. ($\Rightarrow$) If T_r allows a broadcast of time t, a result of applying to it Algorithm 1 will be a calling sequence for r which will order r's sons. By Observation 1, $i + t(T_{ri}^r) = i + t(n) - l(r_i) \leq t(n) - l(r) = t$ for each son r_i. Thus, $t(T_{ri}^r) \leq t - i$. □

Lemma 1 suggests a method of constructing mbt's on n vertices. The idea is to partition the integer $n - 1$ (the number of descendants of the root) into at most $t(n)$ components $n_1, \cdots, n_k$ ($k \leq t(n)$, $n_1 + \cdots + n_k = n - 1$) such that $t(n_i) \cdots t(n) - i$. Such partition defines possible subtree sizes for an mbt of n vertices.

We will now present a theorem, a corollary to which states that not only does the above construction process yield exclusively mbt's, but also all mbt's on n vertices may be generated by it. Let M_n^t denotes the set of all rooted trees on n vertices allowing broadcast time t.

Theorem 2: For a given t and n, $M_n^t = \cup M_{n1}^{t-1} M_{n2}^{t-2} \cdots M_{nk}^{t-k}$, where the union is taken over all values of $k \leq t$ and all partitions $n_1, n_2, \cdots, n_k$ of $n - 1$.

Proof: By definition of M_n^t and by Lemma 1, $M_{n1}^{t-1} \cdots M_{nk}^{t-k} \subseteq M_n^t$ for any k and $n_1, n_2, \cdots n_k$ ($k \leq t$, $t(n_i) \leq t - i$). On the other hand, in any tree T_r allowing a broadcast in time t, the subtrees T_{ri}^r of the root allow broadcast times $t - i$, respectively (by Lemma 1). Thus, $T_{ri}^r \in M_{ni}^{t-i}$ for some $n_1, \cdots, n_k$ such that $k \leq t$ and the sum of n_i gives the number of descendants of r in T_r: $\Sigma_i n_i = n - 1$. □

Corollary 1: For the set of all mbt trees on n vertices we have $M_n^{t(n)} = \cup M_{n1}^{t(n)-1} \cdots M_{nk}^{t(n)-k}$, where the union is taken over all values of $k \leq t(n)$ and all partitions $n_1, \cdots, m_k$ of $n - 1$.

An analogical construction outline may be stated for all broadcasts (ordered rooted trees) on n vertices. Let $\overline{M}_n^t$ denote the set of ordered rooted trees on n vertices defining broadcasts of completion time at most t.

Corollary 2: $\overline{M}_n^t = \cup \overline{M}_{n1}^{t-1} \cdots \overline{M}_{nk}^{t-k}$ for all choices of $k \leq t$ and all partitions $n_1, \cdots, n_k$ of $n - 1$.

Each choice of $k, n_1, \cdots, n_k$ and a tree from each $\overline{M}_{ni}^{t-i}$ ($1 \leq i \leq k$) during the construction process described by Corollary 2 yields a distinct (nonisomorphic) ordered rooted tree. A simple algorithm based upon this fact lists all the ordered rooted trees on n vertices and broadcast time, at most, t. It follows from Observation 3 that all broadcasts with n vertices and completion time t are represented by binary trees with the same number of vertices and height t. Clearly, roots of all such trees have only one (left) son. The construction algorithm essentially enumerates lexicographically all binary trees with $n - 1$ vertices and height $t - 1$. We will state it in terms of the subtree size representation of the broadcast (or, equivalently, left subtree size representation of the binary tree). Let each vertex v of an ordered rooted tree $\overline{T}_r$ be labeled by the size of (number of vertices in) the subtree $\overline{T}^r_v$. These labels, as listed during the preorder traversal of $\overline{T}_r$ (Knuth [11]), represent uniquely the tree.

Algorithm 2: Listing ordered rooted trees.

Input: Number of vertices n and broadcast time t.

Output: List of all nonisomorphic members of $\overline{M}^t_n$.

Method: Generation of all admissible partition of $n - 1$, and partitions of $n_i - 1$, etc. The size representation of a tree (partitions within partitions) is assembled in fields "size" of an array of records [1. .n]. Other fields of a record contain the broadcast time for the subtree considered and the total number of vertices in the subtree not yet assigned a position.

```
/* 0) Initialize*/ tree[1]. size := $n$;
                   tree[2].time := $t - 1$;
                   tree[2].remain := $n - 1$;
                   current := 2; down := true;
/* 1) Scan    */ while current > 1 do with tree [current] do
                   begin if down then size := min (remain, 2 ↑ time)
  /*1.1) set the size*/ else size := size − 1;
  /*1.2) if admissible*/ if (size > 0) & (remain-size
                            < 2 ↑ time)
                          then begin down := true;
                                  if remain > size
  /*1.2.1) if more vertices*/ then begin tree [current + size]
            .time := time − 1;
  /*        remain prepare */
  /*        next subtree */ tree [current + size].remain := remain
                             − size end;
                           if  size > 1 then begin
  /*1.2.2) if any descendants*/     tree[current + 1]
                                     .time := time − 1;
  /*        prepare their partition*/  tree[current + 1]
                                     .remain := size − 1 end;
                                  if current = n then list__sizes
                                  else current := current + 1 end
  /*1.3) not admissible size*/ else begin current := current − 1;
                               down = false end
                              end;
```

Algorithm 2 constructs all broadcasts $\overline{M}^t_n$ in lexicographical order. We omit a straightforward but tedious proof of that fact.

When the mbt's are looked upon as rooted but unordered trees, two different members of $\overline{M}^{t(n)}_n$, $\overline{T}_u$, and $\overline{T}_v$, may be isomorphic in the sense that the sets of the subtrees in the two trees are identical $\{T^u_{ri}\} = \{T^v_{ri}\}$. An illustration is given in Fig. 1 where the operation of joining subtrees for the given sets M^3_3, M^2_3, M^1_2 in order to construct M^4_9 yields isomorphic trees for two different choices of subtrees.

By considering only partitions $n_1, \cdots, n_k$ of $n - 1$, which have the monotonic property $n_1 \geq n_2 \geq \cdots \geq n_k$, we can easily eliminate a large part of isomorphic duplicates in the construction of mbt's analogical to Algorithm 2. However, this will not eliminate cases represented by Fig. 1, where the component sets M^3_3 and M^2_3 have some identical members. The fact that Algorithm 2 constructs the trees in lexicographical order suggests a modification to be applied in construction of mbt's: in cases of sibling subtrees of equal sizes, omit all younger subtrees of lexicographically greater representation. Thus, in our example, $T_4 = 9\ \underline{3\ 1\ 1}\ \underline{3\ 2\ 1}\ 2\ 1$ would not be constructed, since $3\ 1\ 1 < 3\ 2\ 1$. This situation is represented in corresponding binary trees as such, where a right son has the same label (size of the left subtree plus one) as its father. Here again, the son's left subtree must not be lexicographically smaller than the father's left subtree.

The following algorithm constructs all mbt's on n vertices.

Algorithm 3: Listing all nonisomorphic mbt's.

Input: Number of vertices n.

Output: List of mbt's with n vertices.

Method: Modification of Algorithm 2 as explained in the text. A field "forced" is added to a record of a vertex, indicating restricted subtrees.

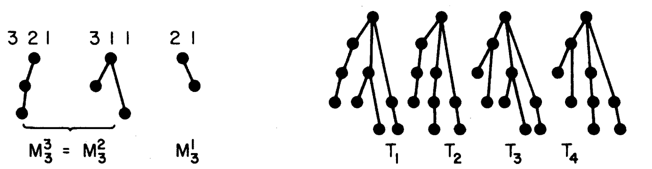

Fig. 1. Two members of $\overline{M}^4_9$ are isomorphic as unordered trees: $T_1 \cong T_4$.

```
/* 0)   Initialize*/ for i := 2 to n do tree[i].forced := 0;
                     tree[1].size := n; tree[2].remain := n − 1;
                     tree[2].time := t(n);
                     current := 2; down := true;
/* 1)   Scan */ while current > 1 do
/*1.1)  propose the current size */
          /*1.1.1) decrease if backing up, */
          /*1.1.2) copy the preceding subtree if forced,*/
          /*1.1.3) or simply set the admissible value */
/*1.2)  if the size is admissible */
          /*1.2.1) prepare next subtree */
          /*1.2.2) prepare descendants */
          /*1.2.3) advance the pointer */
/*1.3)  if size not admissible back up */
```

The only essential difference from Algorithm 2 is in the step 1.1), which we expand below.

```
/*1.1) propose the current size */
     if down then begin if tree [current].forced > 0
           then same__size;
           tree [current].size := min (tree[current].remain,
                    2 ↑ tree[current].time) end
        else tree [current].size := tree[current]
              .size − 1;
```

The procedure "same__size" copies the preceding subtree (if it fits on the current level), advances the pointer, and prepares the next subtree. Algorithm 3 produces all mbt's in lexicographical order with all different orderings of an mbt being represented by the lexicographically largest one.

Different mbt's may be isomorphic when the rooting is neglected. If the broadcast center of a tree T consists of more than one vertex (Slater *et al.* [16]), then other different rooted trees obtained by rooting T in all vertices of the center may be counted in M^t_n but represent the same free tree T. For instance, the three mbt's in Fig. 2 are obtained by differently rooting the same free tree T.

An immediate consequence of Observation 3 is the uniqueness of mbt's on $n = 2^k$ vertices. Indeed, the only binary tree with $2^k - 1$ vertices and height $k - 1$ is the complete binary tree. The mbt with 2^k vertices corresponds to the binary tree whose left subtree is such a complete tree. These unique mbt's on 2^k vertices are also known as "binomial trees" (Vuillemin [17]), named so because of the property that there are $\binom{k}{d}$ vertices of the distance d from the root in such tree of size 2^k.

Corollary 2 and the fact that each choice of a partition (in ordered sense) and a subtree gives a different broadcast lead to the following observation.

Observation 4: For given t and n

$$|\overline{M}^t_n| = {}_k\sum_{n_1\cdots n_k} {}_i|\overline{M}^{t-i}_{n_i}|$$

for all choices of $k \leq t$ and all partitions $n_1, \cdots, n_k$ of $n - 1$.

The reader is referred to [15] for elaboration of the pertinent counting results.

V. Conclusions and Further Research

In this correspondence we have described rooted trees in which a certain one-to-all communication process requires a minimum amount of time over all trees of the same number of vertices. We have presented a linear algorithm to recognize such trees. Another algorithm presented in this paper constructs all minimum broadcast trees for a given number of vertices.

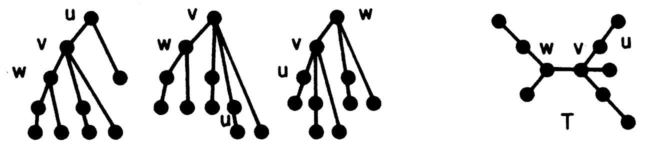

Fig. 2. (a) Three nonisomorphic mbt's, (b) resulting from different optimal rooting of a free tree.

Several continuations of this work have been initiated. One would lead to an efficient construction algorithm (without a need for isomorphism check) of free trees with n vertices which may be rooted as to allow minimum time broadcast. Another avenue of extension of the current results is the investigation of classes of trees yielding minimum broadcast time for a larger number of messages. Still another area of consideration opens when the problems for trees with unit weight of edges are translated into the nonuniformly weighted case.

Acknowledgment

The author gratefully acknowledges Dr. T. Beyer's contributions to the design of data structures of Algorithm 1.

References

[1] B. Baker and R. Shostak, "Gossips and telephones," *Discrete Math.*, vol. 2, pp. 191-193, 1972.
[2] Y. K. Dalal and R. M. Metcalfe, "Reverse path forwarding of broadcast packets," *Commun. Ass. Comput. Mach.*, vol. 21, no. 12, pp. 1040-1048, 1978.
[3] A. Farley, "Minimal broadcast graphs," *Networks*, vol. 9, no. 4, pp. 313-332, 1979.
[4] A. Farley and S. Hedetniemi, "Broadcasting in grid graphs," in *Proc. 9th SE Conf. Combin., Graph Theory, Comput.*, 1978, pp. 275-288.
[5] A. Farley, S. Hedetniemi, S. Mitchell, and A. Proskurowski, "Minimum broadcast graphs," *Discrete Math.*, vol. 25, pp. 189-193, 1979.
[6] A. Farley and A. Proskurowski, "Gossiping in grid graphs," *J. Combinatorics, Inform., Syst. Sci.*, vol. 5, no. 3, pp. 321-328, 1980.
[7] M. R. Garey and D. S. Johnson, *Computers and Intractability*. San Francisco: Freeman,1979.
[8] W. M. Gentelman, "Some complexity results for matrix computations on parallel processors," *J. Ass. Comput. Mach.*, vol. 25, pp. 112-115, 1978.
[9] A. Hajnal, E. C. Milner, and E. Szemeredi, "A cure for the telephone disease," *Can. Math. Bull.*, vol. 5, no. 3, pp. 447-450, 1972.
[10] S. Hedetniemi and S. Mitchell, "Census of minimum broadcast graphs," Dep. Comput. Sci., Univ. of Oregon, Eugene, Tech. Rep. 78-7.
[11] D. E. Knuth, *Art of Computer Programming*, vol. I. Reading, MA: Addison-Wesley, 1973, 2nd ed.
[12] W. Knödel, "New gossips and telephones," *Discrete Math.*, vol. 13, p. 95, 1975.
[13] C. L. Liu, *Introduction to Combinatorial Mathematics*. New York: McGraw-Hill, 1968, pp. 2 and 19.
[14] S. Mitchell, "Algorithms on trees and maximal outer planar graphs: Design, complexity analysis, and data structure studies," Ph.D. dissertation, Univ. of Virginia, Charlottesville, 1976.
[15] A. Proskurowski, "Minimum broadcast trees," Dep. Comput. Sci., Univ. of Oregon, Eugene, Tech. Rep. 78-18.
[16] P. Slater, E. Cockayne, and S. Hedetniemi, "Information dissemination in trees," Dep. Comput. Sci., Univ. of Oregon, Eugene, Tech. Rep. 78-11; also to appear in *SIAM J. Comput.*
[17] J. Vuillemin, "A data structure for manipulating priority queues," *Commun. Ass. Comput. Mach.*, vol. 21, pp. 309-315, Apr. 1978.

Finite-Turn Repetitive Checking Automata and Sequential/Parallel Matrix Languages

PATRICK SHEN-PEI WANG

***Abstract*—A class of machines capable of recognizing two-dimensional sequential/parallel matrix languages is introduced. It is called "finite-turn repetitive checking automata" (FTRCA). This checking automaton is provided with three READ-WRITE heads, one READ only head, and a control register in its memory to keep track of the internal transition states and the stack symbols. The value of the register ranges from 0 to a finite positive integer k and can repetitively appear as many times as necessary. A one-to-one correspondence is established between the class of FTRCA and $(R{:}R)ML$-the smallest class of the sequential/parallel matrix languages. Several closure properties are also investigated.**

***Index Terms*—Automata, instantaneous description, matrix languages, row(column) catenation, row(column) Kleene's closure.**

I. Introduction

In recent years the study of two-dimensional languages has become more and more interesting since it has significant applications in two-dimensional pattern processing [1], [7], [9], [15]. There have also been several studies of parallelism in two-dimensional pattern processing [5], [7], [8]. While parallel generation reduces computer time, it increases the hardware required in contrast to sequential processing, which requires less hardware but more time. As a compromise, sequential parallel methods have been proposed [9], [16], [17].

In [16] the author extended the models of [5], [6], and [12] by allowing higher generative rules for the vertical derivations and placing matrix restriction on the applicability of the rules. We note that all the advantages of the models [5], [6], [12] such as natural hierarchies and simplicity of basic picture operation descriptions carry over to these models as well, and furthermore, these models have higher generative capacity than the earlier ones [5], [6], [12]. Also some of the classes in [16] have generative power greater than that of [5], [6], and [12], while one class is equivalent to the model discussed in [14].

In this correspondence we are going to construct machines that capture $(R{:}R)ML$-the smallest class of sequential/parallel matrix languages [16]. The readers are presumed to have familiarity with the formal language theory [4].

II. Finite-Turn Repetitive Checking Automaton

In this section we define finite-turn repetitive checking automaton and provide the notation necessary to establish the various results in this paper. We adapt the notation and definition of one-way stack automaton and checking automaton and k-turn checking automaton, as found in [3] and [10].

Definition 1: A k-turn repetitive checking automaton (k-turn RCA) is a 16-tuple

$$M = (K, I, T, \delta, \delta_b, R, q_0, Z_0, F, \notin, I', K', \delta', q_0', F', k)$$

where

K: a finite set of states, $q_0 \in K$ initial state,

I: a finite set of input symbols,

Manuscript received March 6, 1979; revised July 7, 1980. This work was supported by the Directorate of Mathematical and Computer Sciences, National Science Foundation under Grant MCS-8003290.

The author is with GTE Research Laboratories, Waltham, MA 02154.