ENGINEERING RESEARCH INSTITUTE
UNIVERSITY OF MICHIGAN
ANN ARBOR

THE FOLDED TREE

ARTHUR W. BURKS
ROBERT McNAUGHTON
CARL H. POLLMAR
DON W. WARREN
JESSE B. WRIGHT

Project 1828

BURROUGHS CORPORATION
RESEARCH CENTER
PAOLI, PENNSYLVANIA

September 30, 1954

engn
UMR 1019

TABLE OF CONTENTS

PREFACE

This report is, in part, a result of rewriting material contained in our two previous reports in the series having the general title, Language Conversion for Digital Computers. Those two reports are (1) General Introduction and Volume I, The Logical Realization of Transliterative Functions, by Arthur W. Burks, Carl H. Pollmar, Don W. Warren, and Jesse B. Wright; and (2) Volume III, Minimal Switch Theory and the Folded Tree, by Arthur W. Burks, Carl H. Pollmar, Don W. Warren, and Jesse B. Wright.

The old material is presented in a new and improved form (Sections 1 through 5) and is augmented by novel results (Sections 6 and 7). The two earlier volumes contained discussions of both folded trees and minimality; the minimality results have been extended and presented anew in a report, Complete Decoding Nets: General Theory and Minimality, and it seemed worthwhile to perform a similar task for the results concerning folded trees.

THE FOLDED TREE*

1. INTRODUCTION

The problem of constructing circuits which perform a certain function and have some formal properties contributing to engineering efficiency was considered by Shannon in [I]. (Roman numerals in brackets refer to the bibliography at the end of this paper.) Part of Shannon's problem (Part II of [I], pp. 81-89) was to formulate an arithmetic condition such that for any sequence of positive integers satisfying that condition a tree circuit can be constructed whose load distribution is given by that sequence. That condition is identical with our notion of "admissibility," defined in Section 3 of this paper.

The present paper takes up this same problem, but is entirely self-contained. No reference to Shannon's paper is necessary either for definitions of concepts or for proofs of theorems. Our results go beyond those of Shannon, both in that we prove the necessity of the admissibility condition and also in that we give a constructive technique, helpful to a practical engineer, for constructing a folded tree with any given load distribution satisfying the condition of admissibility. To obtain these results we must formulate a precise definition of the term "folded tree." Although Shannon does not give a precise definition of any corresponding concept, it is clear that our precisely defined concept applies to the circuits which he considers.

We shall use some of the concepts of our previous paper [II], but we define them anew here, so no acquaintance with that paper is presupposed. Our method, here as in [II], is to discuss diagrams rather than circuits directly. The diagrams may be realized by circuits of various different kinds, e.g., relay transfer contact nets discussed in [I], and electronic digital computing circuits. The advantages of this approach are (1) that the range of application of results is wider, and (2) that the problems, being abstract, can be solved within pure mathematics. Our methods and

* The writing of this paper and the research which it reports were done under the sponsorship of the Burroughs Corporation, Research Center, Paoli, Pennsylvania.

The authors wish to thank Dr. Irving M. Copi for many helpful suggestions and also Dr. Frank Harary for his suggestions.

results are intimately connected with the field of mathematics known as the theory of linear graphs; but they are not an application of any previously known materials of that field, and consequently we presuppose no knowledge of it. Although our definitions and theorems concern diagrams, we shall frequently comment on their significance for physical circuits.

2. DEFINITION OF FOLDED TREE

In this section we formulate a precise definition of the term "folded tree" and show that the diagrams covered by this term are suitable for representing some familiar circuits constructed out of standard devices.

We shall be concerned with *vertex diagrams* which are arrangements of small circles and straight lines, such that (1) each circle has just three lines touching its circumference, one on its left and two on its right, (2) each end of each line may touch either a circle (as in Fig. 1) or the ends of any number of other lines (as in Fig. 10) or it may touch nothing (as in Fig. 1), and (3) no circles touch each other. (In this paper if a word is italicized in a sentence, then the word is defined in that sentence.) The circles are called *vertices* and the lines *wires*; the left-hand wire is called the *vertex-input*, the upper right-hand wire is called the *upper vertex-output*, and the lower right-hand wire, the *lower vertex-output*. The usage here of "output" and "input" is nearer to that of [I] than to [II].

The first vertex diagram we introduce is an *n-bay tree*, for which the following recursive definition is provided:

1. A *1-bay tree* consists of just one vertex with its input and output wires; this vertex is the *first* (and only) *bay* of the 1-bay tree;

2. An *$(i + 1)$-bay tree* results from an i-bay tree when, to each vertex-output u of a vertex in the ith bay, a new vertex is joined so that u is the vertex-input of the new vertex; the new vertices constitute the $(i + 1)$th bay;

3. An n-bay tree has only the vertices and wires provided for in 1 and 2; no diagrams other than those so provided for are trees.

The *input* of an n-bay tree is the vertex-input of the vertex of the first bay. An *output* of an n-bay tree is a vertex-output of a vertex in the nth bay. Note that the tree input is the only wire in the tree which is not a vertex-output, and that the tree outputs are the only wires which are not vertex-inputs. Note also that there are 2^n outputs of an $\underline{n}$-bay tree. The result of deleting the crosses (but no lines or circles) from Fig. 1 is a 4-bay tree. (Note that each bay in this figure is simply a vertical column of vertices. This will be true of all figures of trees in this paper, although the definition of "tree" does not require this.)

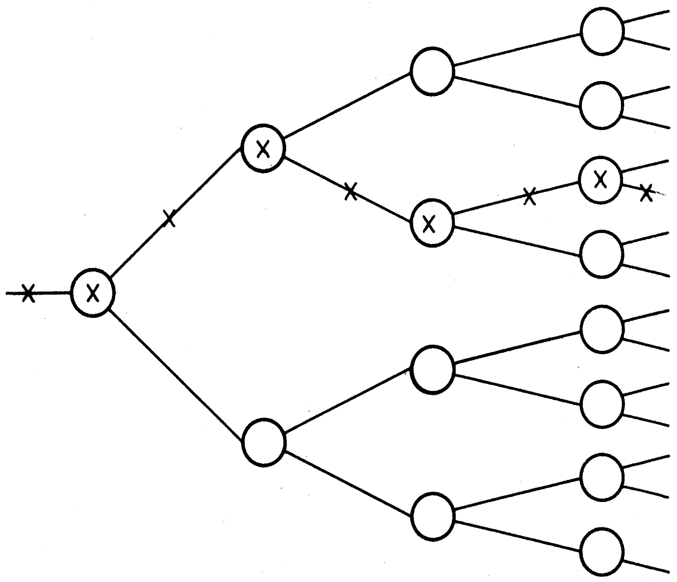

Fig. 1

A wire is to be understood as being at any one moment in either of two states, state 1 or state 0, and a vertex as being in just one of two settings, an upper setting or a lower setting. The state of a vertex-output is determined by the setting of the vertex and the state of the vertex-input: if the vertex-input is in state 1 and if the vertex is in the upper setting, then the upper vertex-output is in state 1 and the lower vertex-output in state 0; if the vertex-input is in state 1 and the vertex in the lower setting, then the upper vertex-output is in state 0 and the lower vertex-output in state 1; if the vertex-input is in state 0, then regardless of the setting of the vertex the state of both vertex-outputs is 0.

Before proceeding we will relate trees to the net diagrams of [II] and described two physical realizations of trees. In the net diagram of Fig. 2 each square is a conjunction ("and") element whose output is in state 1 if and only if both its inputs are in state 1. A vertex, together with its vertex-input and vertex-outputs, and the net of Fig. 2 represent the same type of circuit if exactly one of the input pair $\underline{b}, \underline{b}'$ is in state 1 at any one time. (The use of the word "input" in this connection is similar to the use in [II]; it differs significantly from the use of the word in the major part of this paper.)

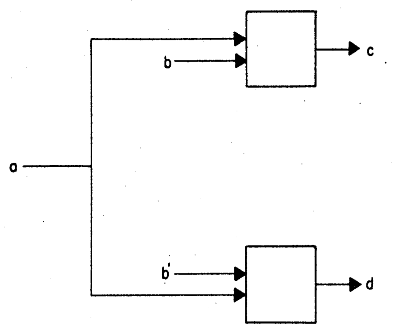

Fig. 2

Wire $\underline{a}$ of Fig. 2 is the vertex-input, and wires $\underline{c}$ and $\underline{d}$ are the vertex-outputs, and the vertex is in the upper setting just in case $\underline{b}$ is in state 1. Nets like the one of Fig. 2 may clearly be combined to form trees.

A conjunction element may be realized by a crystal diode or vacuum tube circuit whose output voltage is high (state 1) whenever both circuit inputs are at high voltage, otherwise the circuit output voltage is low (state 0). Two such circuits, interconnected as in Fig. 2, constitute a physical interpretation of a vertex with its vertex input and two vertex-outputs. Wire $\underline{c}$ is in state 1 whenever $\underline{a}$ and $\underline{b}$ are both in state 1, and $\underline{d}$ is in state 1 whenever $\underline{a}$ and $\underline{b}'$ are both in state 1.

A vertex may also be realized by a relay transfer element. A relay transfer element is a (mechanical) single-pole double-throw switch with the wire of the pole realizing the vertex-input and the wires from the output contacts realizing the vertex-outputs. The transfer element is controlled by a magnet and spring working in opposition, the two positions being represented by the two vertex settings. Thus, if current is flowing (state 1) in the vertex-input, it flows in the upper vertex-output whenever the relay is not activated and in the lower vertex-output whenever the relay is activated.

It should be noted that our vertex diagrams (trees and the diagrams of Section 6) are particularly suited to represent relay contact nets composed of transfer elements, namely, single-pole double-throw switches, electro-magnetically controlled. In such electrical nets the wires to the device (magnet) which controls the settingsof the transfer contacts are not connected into the contact net itself. That our vertex diagrams do not show the wires which determine the vertex settings is intended to represent that feature. In contrast the wires $\underline{b},\underline{b}'$ of an electronic circuit of the sort represented by Fig. 2 are, in general, like $\underline{a}$, $\underline{c}$, and $\underline{d}$, connected to other components of the circuit. Electronic circuits can thereby realize nets which are more complex than vertex diagrams. Hence, the class of circuits represented by vertex diagrams (in the broad sense of Section 6) is narrower than the class of circuits represented by diagrams constructed from conjunction elements. It turns out, for example, that in a certain sense of "cost" the tree is probably a minimal diagram in the first class, while it is not at all minimal in the second class (see Section 6). To put the point alternatively, a generally efficient way to do complete decoding with relays is by means of a tree, but that is not generally an efficient way to do complete decoding with vacuum tubes or crystal diodes.

A chain of an n-bay tree is a sequence $X_1, \ldots, X_{2n+1}$, where X_1 is the tree input and where, for $i \leqq 2n$, if X_i is a vertex-input of some vertex, then X_{i+1} is that vertex and, if X_i is a vertex, then X_{i+1} is one of its vertex-outputs. It follows that X_{2n+1} is a tree output. In Fig. 1 the vertices and wires marked with crosses constitute a chain. It is obvious that each tree output is a member of one and only one chain.

In an n-bay tree the settings of the vertices and the state of the tree input vary independently of each other, but once these are determined the state of each wire is determined. If the tree input is in state 0, then every wire is in state 0, regardless of the settings of the vertices. If the tree input is in state 1, then there will be a chain whose every wire is in state 1 and such that every wire not on the chain is in state 0. What chain it is will depend on the settings of the vertices, since the state of each vertex-output is determined by the setting of the vertex and the state of its vertex-input.

We are interested in trees in which the settings of the vertices do not vary completely independently of each other, but whose vertices are partioned into a number of classes, all the vertices of each class being in the same setting at any one time. In such trees we indicate the class to which each vertex belongs by affixing the same label to every vertex of a given class, vertices which belong to different classes being given different labels. We call such a tree a "labeled-tree," and define it as follows. An n-bay labeled-tree is formed from an n-bay tree by marking each vertex with exactly one label from a set of m labels, and using each of the m labels to mark at least one vertex; m can be less than, equal to, or greater than n. We require, of course, that all vertices having the same label be in the same setting. Fig. 3 is a 3-bay labeled-tree in which $m = n$.

An n-bay folded tree is an n-bay labeled-tree in which there are exactly n labels (i.e., $m = n$) and, for every chain and for each label, there is at least one vertex with that label in the chain. It follows that there is exactly one vertex in each chain with a given label, since there are n labels and n vertices in a chain. A remark on the appropriateness of the term "folded" in this connection will be made in the Appendix. Note that Fig. 4 is a folded tree, but Fig. 3 is not.

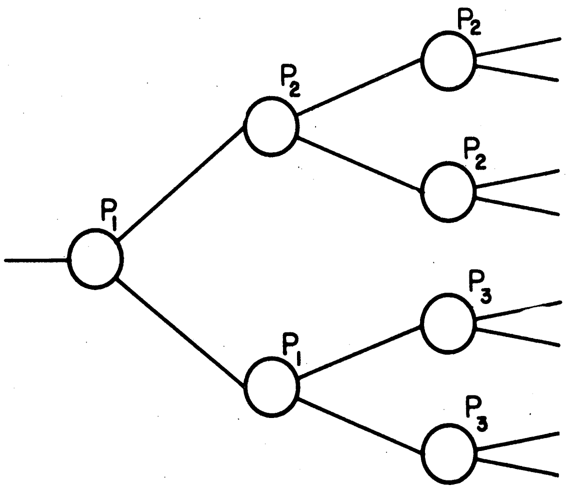

Fig. 3

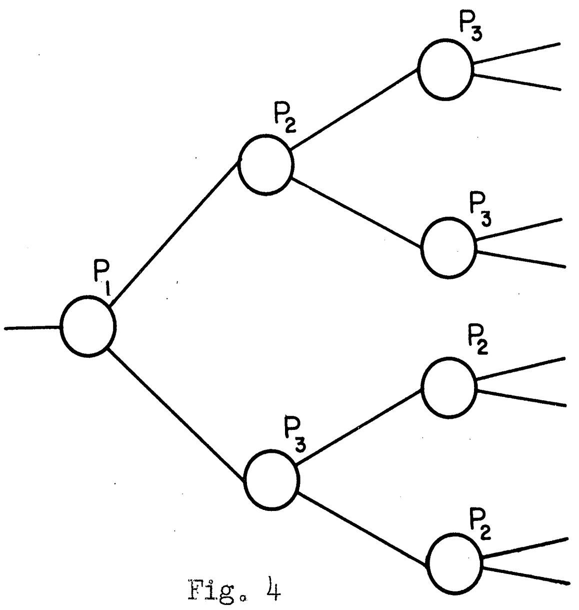

Fig. 4

Folded trees are important among labeled-trees because they can function as "complete decoding nets" in the sense given in [II]. We must explain what this means and why it is so. A circuit represented by a labeled-tree performs its function when the tree input is in state 1 (e.g., after all the transfer contacts of a relay contact net are set, an electrical signal introduced at the tree input will emerge at the desired output). We define a <u>tree state</u> of a labeled-tree to be a definite assignment of its vertex settings such that all vertices with the same label are set the same, and such that the tree input is in state 1. It is clear that there are 2^m different tree states for a labeled-tree having $\underline{m}$ distinct labels.

We say that an $\underline{n}$-bay labeled-tree is <u>complete decoding</u> if and only if the number of labels $\underline{m}$ is equal to $\underline{n}$, and for each tree state there is at least one tree output which is in state 1 in that tree state and state 0 in every other tree state. It follows that there is only one tree output in state 1 in any tree state of a complete decoding $\underline{n}$-bay labeled-tree, since it has exactly 2^n tree states and 2^n tree outputs. (A nearly identical concept of complete decoding is discussed in Section 3 of [II].)

<u>Theorem 1</u>: A necessary and sufficient condition for a labeled-tree to be complete decoding is that it be a folded tree.

Proof of necessity: Consider an $\underline{n}$-bay labeled-tree which is complete decoding. Consider any tree state $\underline{S}$ and the tree output $\underline{Q}$ which is in state 1 only in $\underline{S}$. There is only one chain $\underline{C}$ which includes $\underline{Q}$. Suppose that, for some label $\underline{P_i}$, there is no vertex in $\underline{C}$ labeled $\underline{P_i}$. Now let $\underline{S'}$ be a different tree state which coincides with $\underline{S}$ except for the settings of the vertices labeled $\underline{P_i}$. $\underline{Q}$ would be in state 1 in $\underline{S'}$ as well as in $\underline{S}$, contrary to the assumption that the tree is complete decoding. Hence, every tree state determines one chain such that for each label there is at least one vertex with that label in the chain. Different tree states determine different chains; and since there are 2^n different tree states, there must be 2^n different chains, each of which has the property that for each label there is at least one vertex with that label in the chain. But these are all the chains that there are, since an $\underline{n}$-bay labeled-tree contains exactly 2^n chains. Hence, any complete decoding labeled-tree is a folded tree.

Proof of sufficiency: Consider any $\underline{n}$-bay folded tree. Every tree output $\underline{Q}$ is on a chain $\underline{C}$ containing, for each label $\underline{P_i}$, one and only one vertex labeled $\underline{P_i}$. Hence, there is a tree state $\underline{S}$ in which all the wires of $\underline{C}$ (including $\underline{Q}$) are in state 1. Also, for any different tree state $\underline{S}'$, there will be at least one vertex of $\underline{C}$ in a different setting and so $\underline{Q}$ will be in state 0 in $\underline{S}'$. Therefore, (1) each tree output will be in state 1 in one and only one tree state. Furthermore, (2) any two tree outputs will be in state 1 in different tree states, which can be proved as follows. If $\underline{Q_1}$ and $\underline{Q_2}$ are different tree outputs on chains $\underline{C_1}$ and $\underline{C_2}$, respectively, then let $\underline{i}$ be the number of the latest bay in which $\underline{C_1}$ and $\underline{C_2}$ have a vertex $\underline{V}$ in common. Suppose that $\underline{V}$ is labeled $\underline{P_k}$, and suppose (without loss of generality) that $\underline{C_1}$ includes the upper vertex-output of $\underline{V}$. $\underline{C_2}$ must then include the lower vertex-output of $\underline{V}$. In order for $\underline{Q_1}$ to be in state 1, $\underline{V}$ will have to be in the upper setting, and, in order for $\underline{Q_2}$ to be in state 1, $\underline{V}$ will have to be in the lower setting. Hence, $\underline{Q_1}$ and $\underline{Q_2}$ are in state 1 in different tree states.

Since there are as many tree states as tree outputs it follows from (1) and (2) that the tree is complete decoding. This completes our proof of Theorem 1.

An $\underline{n}$-bay standard tree is an $\underline{n}$-bay labeled-tree in which any two vertices have the same label if and only if they are in the same bay. Fig. 5. is a 3-bay standard tree.

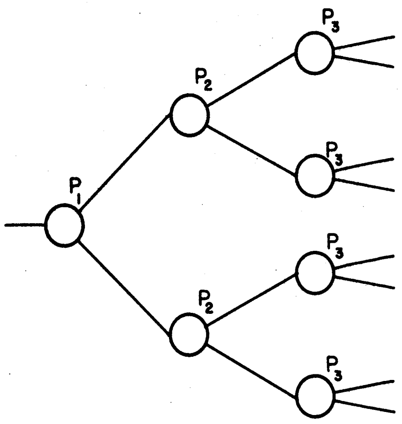

Fig. 5.

(If we interpret the vertices as in Fig. 2, an $\underline{n}$-bay standard tree is essentially the same as the 2-$\underline{n}$ tree defined in Section 4 of [II].)

Consider now two electrical devices, $\underline{T}_1$ realizing a 6-bay standard tree and $\underline{T}_2$ realizing a 6-bay folded tree in which the number of vertices labeled $\underline{P}_1, \ldots, \underline{P}_6$, respectively, is 1, 12, 12, 12, 13, 13. Both perform the same decoding function but the latter has a more nearly equal distribution of labels. Since all of the vertices with the same label are set in the same way at any one time, this means that the load distribution on the wires controlling the settings of the vertices is more nearly equal in $\underline{T}_2$ than in $\underline{T}_1$. (The sequence 1, 12, 12, 12, 13, 13 will later be referred to as the "loading sequence" of the tree realized by $\underline{T}_2$.) If the devices are constructed of relays (and for reasons indicated earlier in this section the tree is more important for relay contact nets than for electronic digital computing circuits), a further factor needs to be considered. This factor is the number of contacts that each available coil controls. One transfer contact can realize one vertex, but two contacts of the same relay cannot realize vertices with different labels. If one had available relays with eight transfer contacts each, $\underline{T}_2$ with load distribution 1, 12, 12, 12, 13, 13 would require eleven relays, while the (standard tree) circuit $\underline{T}_1$ would require only ten relays. On the other hand, a (folded tree) circuit with load distribution 1, 8, 8, 15, 15, 16 would require only nine relays. Depending upon what physical equipment is available, different load distributions for a folded tree offer greater or less advantage. It is therefore of practical interest to know what different load distributions are possible for folded trees.

We are thus led to consider the interesting problem of determining how the $\underline{P}_i$'s can, in general, be distributed among the 2^n-1 vertices of a folded tree. More precisely, given a sequence of $\underline{n}$ integers $\underline{a}_1, \ldots, \underline{a}_n$, is there an $\underline{n}$-bay folded tree having, for each $\underline{i}$, $\underline{a}_i$ vertices labeled $\underline{P}_i$? A condition for a sequence of positive integers to have a folded tree corresponding to it was stated in [I] and there proved to be sufficient. That condition is the property of "admissibility" as defined in the following section. All the sequences discussed in the preceding examples have this property; 1, 2, 5, 6, 20, 29 is a sequence that does not. In Section 3 of this paper we prove admissibility to be a necessary condition also; and in Sections 4 and 5 we present and justify a method of construction which when applied to any admissible sequence of positive integers results in a folded tree corresponding to that sequence, thus providing a proof of sufficiency different from Shannon's earlier proof which was not constructive.

3. LOADING SEQUENCES; ADMISSIBILITY

In this section we define an arithmetic condition on sequences of non-negative integers, which we shall call the condition of "admissibility," and show it to be a necessary condition for a sequence to represent a distribution of labels among the vertices of a folded tree. To define "admissibility" we must first introduce some additional notions.

Let $\underline{V}(\underline{i},\underline{j})$, $1 \leqq \underline{j} \leqq 2^{i-1}$, be the $\underline{j}$th vertex in the $\underline{i}$th bay. Inasmuch as the bays are vertical in our figures, $\underline{V}(\underline{i},1)$ is the top vertex of the $\underline{i}$th bay, $\underline{V}(\underline{i},2)$ the vertex next to the top, etc. It is clear that the vertex-outputs of $\underline{V}(\underline{i},\underline{j})$ are the vertex-inputs of $\underline{V}(\underline{i}+1,2\underline{j}-1)$ and $\underline{V}(\underline{i}+1,2\underline{j})$. Now let us consider all the chains which include $\underline{V}(\underline{i},\underline{j})$. These all have a common vertex in each of the first $\underline{i}$ bays. That part of the labeled-tree which includes $\underline{V}(\underline{i},\underline{j})$ and all vertices from the last $\underline{n}-\underline{i}$ bays which are on chains containing $\underline{V}(\underline{i},\underline{j})$, together with all vertex-inputs and vertex-outputs of those vertices, is itself a labeled-tree and is called a minor tree. We refer to it as $\underline{T}(\underline{i},\underline{j})$ since it is determined by $\underline{V}(\underline{i},\underline{j})$. $\underline{T}(1,1)$ is, of course, the original labeled-tree itself. Note Fig. 6 in which $\underline{T}(2,2)$ is circled.

We assume without any loss of generality that the $\underline{m}$ labels of $\underline{T}(1,1)$ are $\underline{P}_1, \underline{P}_2,\ldots,\underline{P}_m$. The index of $\underline{P}_k$ in $\underline{T}(\underline{i},\underline{j})$ is the number of vertices labeled $\underline{P}_k$ in $\underline{T}(\underline{i},\underline{j})$ which we denote by $\underline{a}_k(\underline{i},\underline{j})$. The loading sequence $\underline{S}(\underline{i},\underline{j})$ of $\underline{T}(\underline{i},\underline{j})$ is the sequence $\underline{a}_1(\underline{i},\underline{j}),\ldots,$ $\underline{a}_m(\underline{i},\underline{j})$. Note that $\underline{a}_k(\underline{i},\underline{j})$ is sometimes 0. $\underline{S}(1,1)$ is then the loading sequence of the original labeled-tree $\underline{T}(1,1)$.

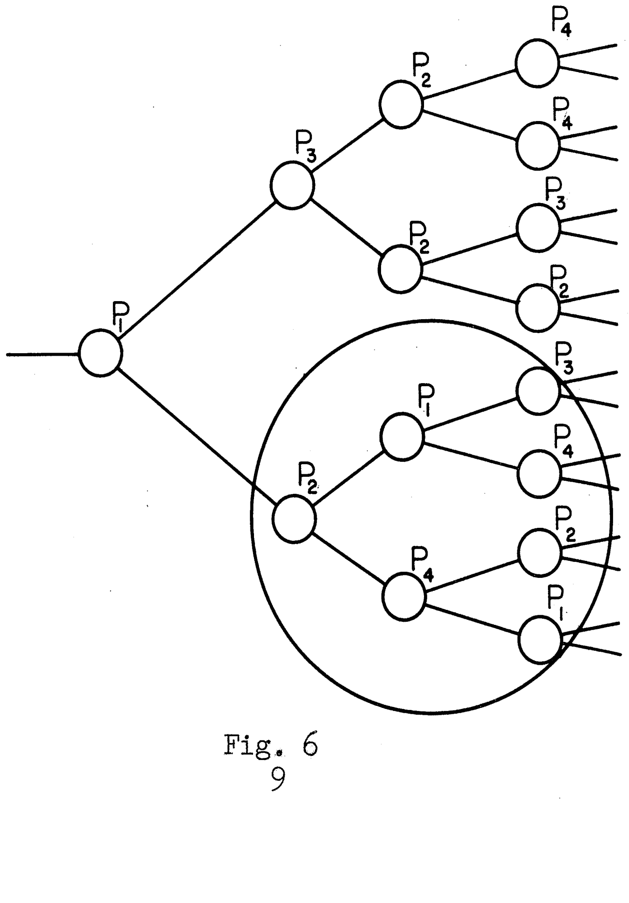

Fig. 6

In Fig. 6. $\underline{S}(2,2)$ is 2, 2, 1, 2, while $\underline{S}(1,1)$ is 3, 5, 3, 4. Note that $\underline{T}(\underline{i},\underline{j})$, $\underline{V}(\underline{i},\underline{j})$, $\underline{S}(\underline{i},\underline{j})$, and $\underline{a}_k(\underline{i},\underline{j})$ are functionally dependent upon the labeled-tree under consideration as well as upon $\underline{i}$ and $\underline{j}$, even though that functional dependence is not explicit in our notation.

Where $\underline{S}$ is a finite sequence of non-negative integers $\underline{a}_1$, ..., $\underline{a}_m$, we define $\underline{M}(\underline{S})$ to be the sequence rearranged in monotonic non-decreasing order, zeros deleted. Thus, if $\underline{S}$ is 3, 7, 5, 0, 2, 1, 0, 5, $\underline{M}(\underline{S})$ is 1, 2, 3, 5, 5, 7.

A sequence $\underline{S}$ of $\underline{m}$ integers satisfies the <u>unit condition</u> if there is a 1 in the sequence. $\underline{S}$ satisifes the <u>total sum condition</u> if the sum of the terms of $\underline{S}$ is $2^{\underline{p}}-1$, where $\underline{p}$ is the number of non-zero terms. $\underline{S}$ satisfies the <u>partial sum condition</u> if, for each $\underline{k} \leqq \underline{p}$, the sum of the first $\underline{k}$ terms of $\underline{M}(\underline{S})$ is $\geqq 2^{\underline{k}}-1$. A sequence $\underline{S}$ is <u>admissible</u> if it satisfies all of these three conditions. Thus, 0, 5, 0, 1, 0, 5, 4 is an admissible sequence in which $\underline{m} = 7$ and $\underline{p} = 4$; and 1, 9, 5, 10, 6 is an admissible sequence in which $\underline{m} = \underline{p} = 5$.

<u>Theorem 2</u>: A minor tree $\underline{T}(\underline{i},\underline{j})$ of an $\underline{n}$-bay folded tree is an $(\underline{n}-\underline{i}+1)$-bay folded tree.

Proof: Obviously $\underline{T}(\underline{i},\underline{j})$ has $\underline{n}-\underline{i}+1$ bays. All the chains of $\underline{T}(1,1)$ containing $\underline{V}(\underline{i},\underline{j})$ have the vertices of the first $\underline{i}$ bays in common. These are labeled with exactly $\underline{i}$ of the labels. Now each chain must contain each of the remaining $\underline{n}-\underline{i}$ labels in the last $\underline{n}-\underline{i}$ bays. But these chains (with the vertices of the first $\underline{i}-1$ bays deleted) are the chains of $\underline{T}(\underline{i},\underline{j})$. Hence, $\underline{T}(\underline{i},\underline{j})$ is a folded tree.

<u>Theorem 3</u>: In an $\underline{n}$-bay folded tree $\underline{S}(\underline{i},\underline{j})$ has $\underline{i}-1$ zeros and $\underline{n}-\underline{i}+1$ non-zero terms.

The proof is immediate from Theorem 2.

<u>Theorem 4</u>: In a folded tree, if $\underline{V}(\underline{i},\underline{j})$ is labeled $\underline{P}_k$, then $\underline{a}_k(\underline{i},\underline{j}) = 1$.

Proof: Since $\underline{V}(\underline{i},\underline{j})$ is labeled $\underline{P}_k$, no other vertex in any chain containing $\underline{V}(\underline{i},\underline{j})$ is labeled $\underline{P}_k$. For suppose that $\underline{V}(\underline{i}',\underline{j}')$, $\underline{i}' \neq \underline{i}$, on a chain $\underline{C}$ with $\underline{V}(\underline{i},\underline{j})$ is also labeled $\underline{P}_k$. Then $\underline{C}$ has $\underline{n}-2$ other vertices which must (in order to satisfy the folded tree condition) be labeled with $\underline{n}-1$ other labels which is impossible.

Theorem 5: In a folded tree $S(i,j)$ satisfies the unit conditon.

Proof: This follows directly from Theorem 4.

Theorem 6: In a folded tree $S(i,j)$ satisfies the total sum condition.

Proof: This follows from theorem 3 since there are $2^{n-i+1}-1$ vertices in $T(i,j)$.

Theorem 7: In an n-bay folded tree $a_k(i+1,2j-1) = 0$ if and only if $a_k(i+1,2j) = 0$.

Proof: $T(i,j)$ by Theorem 2 is a folded tree and, therefore, each P_k which labels any vertex of $T(i,j)$ has to label at least one vertex in each chain of $T(i,j)$. Every such chain contains $V(i,j)$. Beyond $V(i,j)$ each chain of $T(i,j)$ lies wholly within $T(i+1,2j-1)$ or $T(i+1,2j)$. Hence any label occurs in $T(i+1,2j-1)$ if and only if it occurs in $T(i+1,2j)$, from which Theorem 7 follows directly.

Theorem 8: In an n-bay folded tree $S(i,j)$ satisfies the partial sum condition.

Proof: The proof is by induction, beginning at the nth bay of the tree and going to the left. We first show (1) that, for every $j \leqq 2^{n-1}$, the theorem holds for $S(n,j)$. We then show (2) that, if it holds for $S(i+1,j)$ for every $j \leqq 2^{i}$, it holds for $S(i,j)$ for every $j \leqq 2^{i-1}$. (1) is true by Theorem 3 for $S(n,j)$ has only one non-zero term. We prove (2) by showing that, if it holds for $S(i+1,2j-1)$ and $S(i+1,2j)$, then it holds for $S(i,j)$. Suppose $V(i,j)$ is labeled P_q. Let $M(S(i,j))$ be $a_q(i,j)$, $a_{g_1}(i,j)$, $a_{g_2}(i,j)$, ..., $a_{g_{n-i}}(i,j)$, where, of course, $a_q(i,j) = 1$. By hypothesis $S(i+1,2j-1)$ and $S(i+1,2j)$ satisfy the partial sum condition; in other words, for each $k \leqq n-i$, the sum of the first k terms of $M(S(i+1,2j-1))$ or $M(S(i+1,2j)) \geqq 2^k-1$. Now, if such a condition holds for a monotonic non-decreasing sequence, it holds for the sequence in any order. Furthermore, since, for $k \neq q$, $a_k(i,j) = a_k(i+1,2j-1) + a_k(i+1,2j)$, with the aid of Theorem 7 it is easy to see that the non-zero terms of $S(i+1,2j-1)$ are $a_{g_1}(i+1,2j-1)$, ..., $a_{g_{n-i}}(i+1,2j-1)$ and the non-zero terms of $S(i+1,2j)$ are $a_{g_1}(i+1,2j)$, ..., $a_{g_{n-i}}(i+1,2j)$. Hence for each $\underline{k} \leqq \underline{n}-\underline{i}$, $\sum_{x=1}^{k} a_{g_x}(\underline{i}+1,2\underline{j}-1) \geqq 2^{k}-1$ and $\sum_{x=1}^{k} a_{g_x}(\underline{i}+1,2\underline{j}) \geqq 2^{k}-1$. Hence, $\sum_{x=1}^{k} (a_{g_x}(\underline{i}+1,2\underline{j}-1) + a_{g_x}(\underline{i}+1,2\underline{j})) \geqq 2^{k+1}-2$. Clearly,

$$a_q(\underline{i},\underline{j}) + \sum_{x=1}^{k} a_{g_x}(\underline{i},\underline{j}) = 1 + \sum_{x=1}^{k} (a_{g_x}(\underline{i}+1,2\underline{j}-1) + a_{g_x}(\underline{i}+1,2\underline{j}));$$

hence for any $\underline{k} \leqq \underline{n}-\underline{i}$, $a_q(\underline{i},\underline{j}) + \sum_{x=1}^{k} a_{g_x}(\underline{i},\underline{j}) \geqq 2^{k+1}-1$, which means that the sum of the first $\underline{k}+1$ terms of $\underline{M}(\underline{S}(\underline{i},\underline{j}))$ is $\geqq 2^{k+1}-1$, proving that $\underline{S}(\underline{i},\underline{j})$ satisfies the partial sum condition.

Theorem 9: The loading sequence $\underline{S}(1,1)$ of an $\underline{n}$-bay folded tree is an admissible sequence of $\underline{n}$ non-zero terms.

This result follows immediately from Theorems 3, 5, 6, and 8, putting $\underline{i} = \underline{j} = 1$.

4. THE SPLITTING TECHNIQUE

In this section we provide an effective method of constructing a folded tree with a given admissible loading sequence of positive integers. The proof that this method will produce a folded tree is given in the next section. The procedure consists in taking the loading sequence proposed for the folded tree $\underline{T}(1,1)$, selecting the unit term for $\underline{V}(1,1)$, and dividing the remaining terms so as to yield two admissible sequences, one for $\underline{T}(2,1)$ and the other for $\underline{T}(2,2)$. This procedure, called the "splitting technique", is then repeated for each of those two sequences to provide admissible sequences for $\underline{T}(3,1)$, $\underline{T}(3,2)$, $\underline{T}(3,3)$, and $\underline{T}(3,4)$. The process is iterated until $\underline{T}(\underline{n},2^{n-1})$ is reached. It will be proved in the next section that each vertex is thus provided with a label and that, for each $\underline{k}$, the proper number of vertices will be labeled $\underline{P}_k$.

It must be admitted that there are alternative splitting techniques which applied to the same admissible sequence give different folded trees. Our splitting technique, for example, applied to the admissible sequence 1, 4, 5, 5 gives a folded tree having three different labels in its last bay, but there is also a folded tree with the same loading sequence which has only two different labels in its last bay (see Fig. 7).

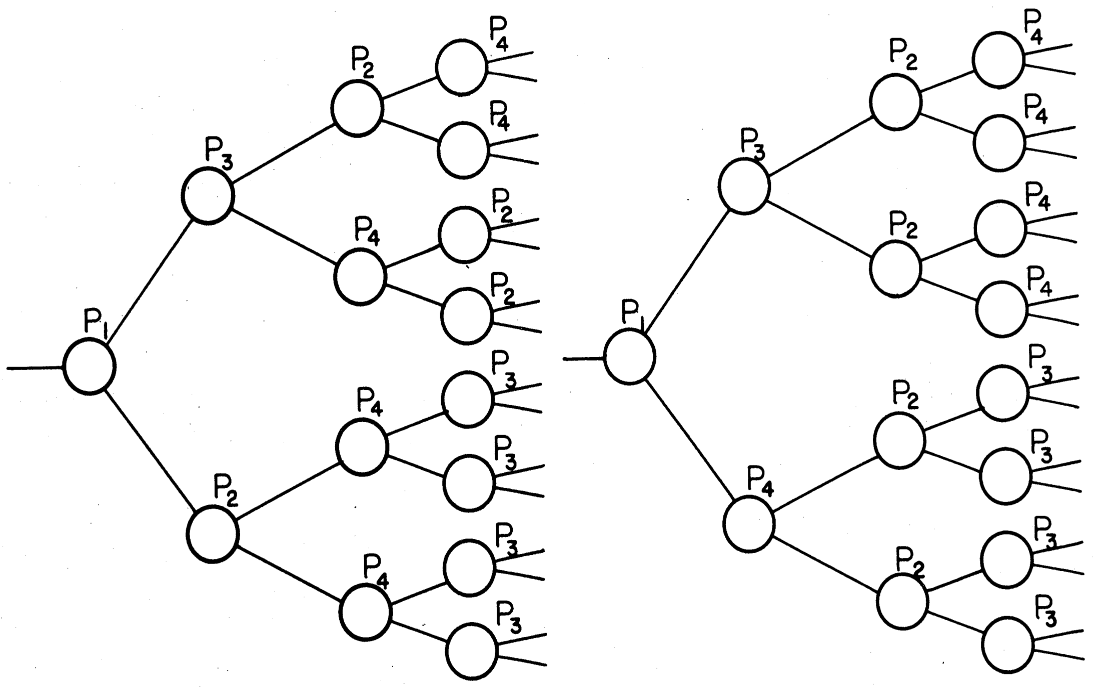

Fig. 7

In this section and in Section 5 we let $\underline{S}'(\underline{i},\underline{j})$, which is $\underline{a}_1'(\underline{i},\underline{j})$, $\underline{a}_2'(\underline{i},\underline{j})$, ..., $\underline{a}_n'(\underline{i},\underline{j})$, be the sequence obtained by repeated applications of the splitting technique from $\underline{S}'(1,1)$, which is an arbitrary admissible sequence without zero terms. The symbols $\underline{S}'(\underline{i},\underline{j})$ and $\underline{S}(\underline{i},\underline{j})$ are to be clearly distinguished even though in some cases they refer to the same sequence of numbers. $\underline{S}'(\underline{i},\underline{j})$ is the sequence derived by our splitting technique from a given admissible sequence $\underline{S}'(1,1)$, whereas $\underline{S}(\underline{i},\underline{j})$ is the loading sequence of $\underline{T}(\underline{i},\underline{j})$, a minor tree of a given labeled-tree.

Let us suppose that $\underline{S}'(\underline{i},\underline{j})$ is a given admissible sequence. We describe now the splitting technique which determines $\underline{S}'(\underline{i}+1,2\underline{j}-1)$ and $\underline{S}'(\underline{i}+1,2\underline{j})$. For every $\underline{k}$, if $\underline{a}_k'(\underline{i},\underline{j}) = 1$ or 0, then let $\underline{a}_k'(\underline{i}+1,2\underline{j}-1) = \underline{a}_k'(\underline{i}+1,2\underline{j}) = 0$. Let $\underline{M}(\underline{S}'(\underline{i},\underline{j}))$ be $1,\underline{d}_1, \ldots, \underline{d}_p$. (It will turn out that $\underline{p} = \underline{n}-\underline{i}$.) We then determine sequences $\underline{b}_1, \ldots, \underline{b}_p$ and $\underline{c}_1, \ldots, \underline{c}_p$, such that, if $\underline{d}_x$ is $\underline{a}_k'(\underline{i},\underline{j})$, $\underline{b}_x$ and $\underline{c}_x$ are to be $\underline{a}_k'(\underline{i}+1,2\underline{j}-1)$ and $\underline{a}_k'(\underline{i}+1,2\underline{j})$, respectively. The recursive procedure for determining $\underline{b}_x$ and $\underline{c}_x$ is as follows: (1) put $\underline{c}_1 = 1$ and $\underline{b}_1 = \underline{d}_1-1$; (2) if $\underline{b}_1 = 1$ (i.e., if $\underline{d}_1 = 2$), proceed to (3), otherwise first put $\underline{b}_2 = 1$ and $\underline{c}_2 = \underline{d}_2-1$; (3) (for each $\underline{x}$, $\underline{x} \geqq 3$, and, for $\underline{x} = 2$ if $\underline{d}_1 = 2$) assuming that $\underline{b}_1, \ldots, \underline{b}_{x-1}$ and $\underline{c}_1, \ldots, \underline{c}_{x-1}$ have been determined, put

$$\underline{b}_x = \left[\frac{d_x + \Delta_x}{2}\right]$$

and

$$\underline{c}_x = \left[\frac{\underline{d}_x + 1 - \Delta_x}{2} \right]$$

where

$$\Delta_x = \sum_{y=1}^{x-1} (\underline{c}_y - \underline{b}_y) .$$

Here $[\phi]$, for any real number ϕ, is the integral part of ϕ, i.e., the greatest integer not greater than ϕ. The reader can readily verify that for each $\underline{x}$, $\underline{b}_x + \underline{c}_x = \underline{d}_x$.

In actual calculation the work is even simpler than it appears from the formal statement of the procedure. The idea behind the splitting technique is simply to provide for the 1 in each sequence, and, for each $\underline{d}_x$, to find a $\underline{b}_x$ and $\underline{c}_x$ such that $\underline{b}_x + \underline{c}_x = \underline{d}_x$, in such a way that

$$\sum_{y=1}^{x} \underline{b}_y$$

is as nearly equal as possible to

$$\sum_{y=1}^{x} \underline{c}_y .$$

We also require that

$$\sum_{y=1}^{x} \underline{c}_y$$

be never less than

$$\sum_{y=1}^{x} \underline{b}_y ,$$

except where $\underline{d}_1 > 2$ and $\underline{x} = 2$. Thus, all terms except possibly the first three will be split as evenly as possible so that Δ_x for $\underline{x} > 3$ is either 0 or 1. For $\underline{x} > 3$, then, if $\underline{d}_x$ is even, $\underline{b}_x = \underline{c}_x = \underline{d}_x/2$; if $\underline{d}_x$ is odd and Δ_x is 1, then $\underline{b}_x = (\underline{d}_x+1)/2$ and $\underline{c}_x = (\underline{d}_x-1)/2$; if $\underline{d}_x$ is odd and Δ_x is 0, then $\underline{b}_x = (\underline{d}_x-1)/2$

and $c_x = (d_x+1)/2$. Furthermore, Δ_x does not have to be recalculated each time for $x > 3$; for if d_x is even, then $\Delta_{x+1} = \Delta_x$, and if d_x is odd, then Δ_{x+1} is 1 if Δ_x is 0 and 0 if Δ_x is 1.

The following table carries through a calculation from $S'(i,j)$ to $S'(i+1,2j-1)$ and $S'(i+1,2j)$.

$S'(i,j)$	0	3	0	1	8	14	5
d-sequence	3	5	8	14			
b-sequence	2	1	5	7			
c-sequence	1	4	3	7			
$S'(i+1,2j-1)$	0	2	0	0	5	7	1
$S'(i+1,2j)$	0	1	0	0	3	7	4

It is simple to rearrange the terms of the b and c sequences in forming the sequences $S'(i+1,2j-1)$ and $S'(i+1,2j)$ in accordance with the rule that where d_x is $a'_k(i,j)$, b_x and c_x are $a'_k(i+1, 2j-1)$ and $a'_k(i+1,2j)$, respectively.

To illustrate the case where $d_1 = 2$, consider the $M(S'(i,j))$ sequence 1, 2, 5, 9, 14. Here the b-sequence is 1, 2, 5, 7 and the c-sequence is 1, 3, 4, 7.

Since we have shown how an admissible $S'(i,j)$ is split into $S'(i+1,2j-1)$ and $S'(i+1,2j)$, it is easy to see that given $S'(1,1)$ all the $S'(i,j)$ can be calculated, provided all the applications of the splitting technique yield admissible sequences. Once all these have been calculated, a folded tree whose $S(i,j)$ is $S'(i,j)$ can easily be constructed. If there are n terms in $S'(1,1)$, then, of course, we want an n-bay folded tree. An n-bay tree is constructed and each vertex $V(i,j)$ is labeled P_k where $a'_k(i,j)$ is unity. (In the next section we prove that a vertex will be assigned one and only one label by this procedure.)

We will now provide an example showing the use of this procedure in constructing a 5-bay folded tree with the given admissible loading sequence, 1, 7, 7, 8, 8, which is one of the most evenly balanced loading sequences possible for a 5-bay folded tree. We omit the d-sequence, as well as the b-sequence and c-sequence, in each step.

Since $S'(1,1)$ is 1, 7, 7, 8, 8,

$S'(2,1)$ is 0, 6, 1, 4, 4, and $S'(2,2)$ is 0, 1, 6, 4, 4;
$S'(3,1)$ is 0, 3, 0, 3, 1, and $S'(3,2)$ is 0, 3, 0, 1, 3;
$S'(3,3)$ is 0, 0, 3, 3, 1, and $S'(3,4)$ is 0, 0, 3, 1, 3;
$S'(4,1)$ is 0, 2, 0, 1, 0, and $S'(4,2)$ is 0, 1, 0, 2, 0;
$S'(4,3)$ is 0, 2, 0, 0, 1, and $S'(4,4)$ is 0, 1, 0, 0, 2;
$S'(4,5)$ is 0, 0, 2, 1, 0, and $S'(4,6)$ is 0, 0, 1, 2, 0;
$S'(4,7)$ is 0, 0, 2, 0, 1, and $S'(4,8)$ is 0, 0, 1, 0, 2.

For each i,j we label $V(i,j)$ P_k where the kth term of $S'(i,j)$ is unity. It is not necessary to compute $S'(5,j)$ for any j, since every $V(5,j)$ can be labeled with that label which has not already appeared in the chain containing that $V(5,j)$. The result is Fig. 8.

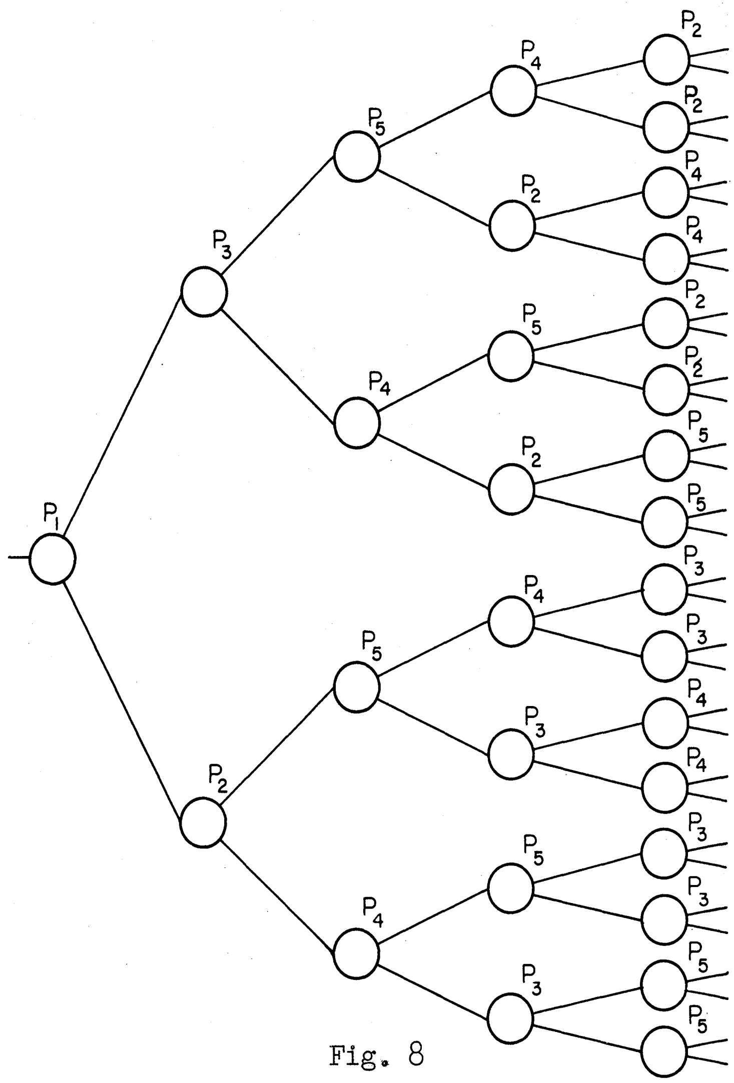

Fig. 8

5. THE VALIDITY OF THE SPLITTING TECHNIQUE

In this section we show that, given any admissible n-term sequence $S'(1,1)$ containing no zeros, the procedure specified in Section 4 always results in a folded tree whose loading sequence $S(1,1) = S'(1,1)$. Thus we have a constructive proof that the admissibility of any sequence of positive integers is a sufficient condition for it to be the loading sequence of a folded tree. The most difficult part of the proof is to show that the splitting technique is admissibility-preserving, i.e., to show that the sequences $S'(i+1,2j-1)$ and $S'(i+1,2j)$ which result from $S'(i,j)$ by the splitting technique are admissible if $S'(i,j)$ is admissible.

(We do not give a complete proof here but rely on the results obtained in [IV]. We shall, however, present enough of the proof to give an intuitive understanding of the argument.)

Theorem 10: If $S'(i,j)$ is an admissible sequence with at least two non-zero terms, then the two sequences which result from applying the splitting technique to it satisfy the unit condition.

Proof (here and in the following proofs we use the notation developed in Section 4): Since $S'(i,j)$ has at least two non-zero terms, d_1 must exist. $d_1 > 1$ since $M(S'(i,j))$ satisfies the partial sum condition. If $d_1 = 2$, then $b_1 = c_1 = 1$. If $d_1 > 2$, then $S'(i,j)$ cannot have exactly two non-zero terms, otherwise $S'(i,j)$ would not satisfy the total sum condition; hence, d_2 also exists and $c_1 = b_2 = 1$.

Theorem 11: Both of the sequences which result from applying the splitting technique to an admissible sequence satisfy the total sum condition.

Proof: We need two lemmas.

Lemma 1. If $4 \leqq x \leqq p + 1$, then $\Delta_x = 0$ or 1; if $d_1 = 2$, then this is so for $x = 1,2,3$ as well.

The proof is by induction, and can be found on p. 33 of [IV] (Lemma 1). (Note that "d_x" of this paper corresponds to "a_i" of [IV].)

Lemma 2. $\Delta_p + 1 = 0$.

Proof: We show first that $\Delta_p + 1 = 0$ or 1. By Lemma 1 the only possible exceptions to this would be where $d_1 > 2$ and $p = 1$ or 2. If $p = 1$, then $d_1 = 2$, in order that $M(S'(i,j))$ satisfy the total sum condition. If $p = 2$, then $d_1 + d_2 = 6$ by the total sum condition. If $d_1 > 2$, then $d_1 = d_2 = 3$, since the d-sequence is monotonic non-decreasing. Here $c_1 = b_2 = 1$ and $b_1 = c_2 = 2$ so that $\Delta_3 = 0$. Having shown that in every case $\Delta_p + 1 = 0$ or 1, we must now show that the value is actually 0. As mentioned in the preceding section, for each x, $c_x + b_x = d_x$; hence,

$$\sum_{x=1}^{p} c_x + \sum_{x=1}^{p} b_x = \sum_{x=1}^{p} d_x \quad .$$

But, since $M(S'(i,j))$ satisfies the total sum condition,

$$\sum_{x=1}^{p} d_x = 2^{p+1}-2,$$

which is even. But Δ_{p+1} is precisely the difference between

$$\sum_{x=1}^{p} c_x$$

and

$$\sum_{x=1}^{p} b_x \quad .$$

If Δ_{p+1} were 1, then their sum would have to be odd. Hence, $\Delta_{p+1} = 0$.

The theorem now follows directly. Since Lemma 2 entails that

$$\sum_{x=1}^{p} c_x$$

and

$$\sum_{x=1}^{p} b_x$$

are equal, it follows that

$$\sum_{x=1}^{p} d_x = 2\sum_{x=1}^{p} b_x = 2\sum_{x=1}^{p} c_x = 2^{p+1}-2 \; .$$

Hence,

$$\sum_{x=1}^{p} b_x = \sum_{x=1}^{p} c_x = 2^{p}-1$$

and Theorem 11 is proved.

Theorem 12: If $b_1, \ldots, b_p$ and $c_1, \ldots, c_p$ are the sequences which result from the application of the splitting technique to an admissible sequence $S'(i,j)$, then for every x such that $1 \leqq x \leqq p$,

$$\sum_{y=1}^{x} b_y \geqq 2^{x}-1$$

and

$$\sum_{y=1}^{x} c_y \geqq 2^{x}-1 \; .$$

This is proved as Lemma 3 on p. 34 of [IV]. (The "derived order", as the term is used in [IV], is the order $b_1, b_2, \ldots, b_p$ and $c_1, c_2, \ldots, c_p$ as opposed to the monotonic order.) Roughly, Theorem 12 follows because $M(S'(i,j))$ satisfies the partial sum condition and the xth term of the b-sequence and the xth term of the c-sequence are usually obtained from the $(x+1)$st term of $M(S'(i,j))$ in such a way that

$$\sum_{y=1}^{x} b_y$$

is nearly equal to

$$\sum_{y=1}^{x} \underline{c}_y \quad .$$

Thus,

$$\sum_{y=1}^{x} \underline{b}_y$$

is approximately

$$1/2\sum_{y=1}^{x} \underline{d}_y \quad .$$

Since 1, $\underline{d}_1$, ..., $\underline{d}_x$ are the first $\underline{x}+1$ terms of $\underline{M}(\underline{S}'(\underline{i},\underline{j}))$ and, since $\underline{S}'(\underline{i},\underline{j})$ satisfies the partial sum condition,

$$1 + \sum_{y=1}^{x} \underline{d}_y \quad \geqq \quad 2^{x+1}-1$$

and hence

$$1/2\sum_{y=1}^{x} \underline{d}_y \quad \geqq \quad 2^{x}-1 \quad .$$

Theorem 13: If $\underline{S}'(\underline{i},\underline{j})$ is admissible, then $\underline{S}'(\underline{i}+1,2\underline{j}-1)$ and $\underline{S}'(\underline{i}+1,2\underline{j})$ satisfy the partial sum condition.

This is proved as Theorem 3 on p. 40 of [IV]. In order to prove Theorem 13 one must extend the result of Theorem 12; for the $\underline{b}$-sequence and $\underline{c}$-sequence are not always in monotonic non-decreasing order, and a sequence satisfying the summation condition of Theorem 12 may no longer satisfy it when monotonized. Fortunately, it can be proved that after the third term the $\underline{b}$-sequence and $\underline{c}$-sequence are each, in the terminology of [IV], quasi-monotonic, i.e., for any $\underline{x}$ and $\underline{x}'$, if $4 \leqq \underline{x} < \underline{x}' \leqq \underline{p}$, then $\underline{b}_x \leqq \underline{b}_{x'} + 1$ and $\underline{c}_x \leqq \underline{c}_{x'} + 1$. It is rather easy to show by virtue of this fact and Theorem 12 that, for each $\underline{x}$,

$$\sum_{y=1}^{x} \underline{b}'_y \quad \geqq \quad 2^{x}-1 \quad ,$$

where $\underline{b}_1', \ldots, \underline{b}_p'$ is monotonic after the third term and results from $\underline{b}_1, \ldots, \underline{b}_p$ by interchanging the appropriate terms after $\underline{b}_3$; similarly for the $\underline{c}$-sequence. (Cf. the theorem on p. 18 on [IV].) The extension of this result to the case where the first three terms are rearranged to make the entire sequence monotonic is made in the proof in [IV].

Theorem 14: For every $\underline{i} < \underline{n}$, if $\underline{S}'(\underline{i},\underline{j})$ has $\underline{n}$ terms including exactly $\underline{i}$-1 zero terms and is admissible, and if $\underline{S}'(\underline{i}+1,2\underline{j}-1)$ and $\underline{S}'(\underline{i}+1,2\underline{j})$ are obtained from $\underline{S}'(\underline{i},\underline{j})$ by the splitting technique, then $\underline{S}'(\underline{i}+1,2\underline{j}-1)$ and $\underline{S}'(\underline{i}+1, 2\underline{j})$ are admissible and each contains exactly $\underline{i}$ zeros.

Proof: That these are admissible follows from Theorems 10, 11, and 13. $\underline{S}'(\underline{i}+1,2\underline{j}-1)$ differs from $\underline{b}_1,\ldots, \underline{b}_p$ in at most the presence of zeros and the order of terms, which does not affect admissibility; similarly for $\underline{S}'(\underline{i}+1,2\underline{j})$ and $\underline{c}_1, \ldots, \underline{c}_p$. To prove that they each have $\underline{i}$ zeros, it suffices to prove that there is one and only one unit term in $\underline{S}'(\underline{i},\underline{j})$. That there is one follows from the fact that $\underline{S}'(\underline{i},\underline{j})$ satisfies the unit condition. That there is only one follows from the fact that $\underline{S}'(\underline{i},\underline{j})$ satisifes the partial sum condition; for if there were at least two, then the sum of the first two terms of $\underline{M}(\underline{S}'(\underline{i},\underline{j}))$ would be $2 < 2^2-1$.

Theorem 15: Given $\underline{S}'(1,1)$ as an admissible $\underline{n}$-term sequence without zeros, the procedure specified in Section 4 results in an $\underline{n}$-bay folded tree whose $\underline{S}(1,1) = \underline{S}'(1,1)$.

Proof: From Theorem 14 it follows that for any $\underline{j} < 2^{n-1}$ the sequence $\underline{S}'(\underline{n},\underline{j})$ has $\underline{n}$-1 zeros and is admissible. Its one non-zero term, say, $\underline{a}_k$, must be unity by the total sum condition, and so $\underline{V}(\underline{n},\underline{j})$ is labeled $\underline{P}_k$. Therefore, (1) $\underline{S}'(\underline{n},\underline{j})$ is $\underline{S}(\underline{n},\underline{j})$ and $\underline{T}(\underline{n},\underline{j})$ is a folded tree.

We now go on to prove that (2) for any $\underline{i}$ and $\underline{j}$, if $\underline{S}'(\underline{i}+1,2\underline{j}-1)$ is $\underline{S}(\underline{i}+1,2\underline{j}-1)$ and $\underline{S}'(\underline{i}+1,2\underline{j})$ is $\underline{S}(\underline{i}+1,2\underline{j})$, and if $\underline{T}(\underline{i}+1,2\underline{j}-1)$ and $\underline{T}(\underline{i}+1,2\underline{j})$ are folded trees, then $\underline{S}'(\underline{i},\underline{j})$ is $\underline{S}(\underline{i},\underline{j})$ and $\underline{T}(\underline{i},\underline{j})$ is a folded tree. There is a set

$$\left\{ \underline{P}_{m_1}, \ldots, \underline{P}_{m_{n-i}} \right\}$$

of $n-i$ distinct labels such that each chain of $T(i+1,2j-1)$ contains exactly one vertex labeled with each member of the set. The m_1th, ..., m_{n-i}th terms of $S(i+1,2j-1)$, which is $S'(i+1,2j-1)$, are exactly those terms which are non-zero terms. There is a similar set for $T(i+1,2j)$. Now this set is identical to the set for $T(i+1,2j-1)$, since for any q the qth term of $S'(i+1,2j-1)$ will be made zero by the splitting technique if and only if the qth term of $S'(i+1,2j)$ is made zero. If $V(i,j)$ is labeled P_k, then k is not one of the $m_1, \ldots, m_{n-i}$; this is so because the kth term of $S'(i,j)$ is one and, therefore, the kth term of $S'(i+1,2j-1)$ and the kth term of $S'(i+1,2j)$ are each zero. From this it follows that every chain of $T(i,j)$ has exactly one vertex labeled with each member of the set

$$\left\{P_k, P_{m_1}, \ldots, P_{m_i}\right\} ,$$

since a chain of $T(i,j)$ is either a chain of $T(i+1,2j-1)$ or a chain of $T(i+1,2j)$ with $V(i,j)$ and its vertex input added. Hence $T(i,j)$ is a folded tree. Now each term of $S'(i,j)$, except $a'_k(i,j) = 1$, is equal to the sum of the corresponding terms of $S'(i+1,2j-1)$ and $S'(i+1,2j)$; $a'_k(i+1,2j-1) = a'_k(i+1,2j) = 0$. (This follows from the specification of the splitting technique.) Since the vertices of $T(i,j)$ are those of $T(i+1,2j-1)$ and $T(i+1,2j)$ together with $V(i,j)$, it follows, from the fact that $S'(i+1,2j-1)$ is $S(i+1,2j-1)$ and $S'(i+1,2j)$ is $S(i+1,2j)$, that $S'(i,j)$ is $S(i,j)$.

From (1) and (2) it follows immediately by induction that $S'(1,1)$ is $S(1,1)$ and $T(1,1)$ is a folded tree.

6. THE ECONOMY OF THE FOLDED TREE

The question arises as to whether circuits represented by folded trees are the most "economical" ones which can function as complete decoding circuits. As we have indicated earlier (Section 2) the answer is in the negative so far as electronic digital computing circuits are concerned (see also Section 5 of [II]). However, folded tree relay transfer contact nets are probably the most economical (in a sense to be defined) of all complete decoding relay transfer contact nets. To formulate this proposition precisely we must delimit the class of diagrams whose realizations are all such complete decoding relay transfer contact nets.

Our definition of vertex diagram at the beginning of Section 2 was motivated by two considerations: (1) that a relay transfer contact is well represented by a vertex with a single vertex-input and two vertex-outputs, and (2) that in a relay transfer contact net the relay transfer contacts can be arbitrarily connected. Hence any relay transfer contact net can be represented by some vertex diagram. In this section we go on to define an <u>n</u>-label complete decoding vertex diagram in such a way that every transfer contact net which performs a complete decoding function is represented by a diagram of this kind. Our belief that folded tree relay transfer contact nets are probably the most economical of all complete decoding relay transfer contact nets can now be more precisely stated as the conjecture: The <u>n</u>-bay folded tree has the minimal number of vertices of any <u>n</u>-label complete decoding vertex diagram.

The objection may be made that the minimality of the number of vertices of a complete decoding vertex diagram is not a sufficient condition for its realization by relays to be minimal in cost, for the cost of a relay transfer contact net depends not only on the number of transfer contacts in it but also on the number of relay coils required to operate it. To particularize this objection, consider the problem of constructing a complete decoding net using only relays with eight transfer contacts each; here the number of relays required is a better indication of the cost of the net than the number of transfer contacts it contains.

There is a certain force to this objection. However, it is to a large extent mitigated by our previous folding results. For if an <u>n</u>-bay standard tree circuit has a minimal number of contacts, we can in practice use that folded tree circuit which of all <u>n</u>-bay folded trees has the least number of relay coils. For an example see the last part of Section 2. When relays with different numbers of contacts are available at costs which are not directly proportional to the number of contacts they contain, then a different folding can be employed to minimize the total cost of the circuit.

We have not proved our minimality conjecture, but in this section we present a partial result in that direction. To this end we will introduce a certain subclass of the class of <u>n</u>-label complete decoding vertex diagrams, namely, the subclass of all <u>n</u>-label progressive diagrams. We shall prove that the <u>n</u>-bay folded tree has the minimal number of vertices of an <u>n</u>-label progressive diagram. It seems, intuitively, that an <u>n</u>-label complete decoding vertex diagram which is not an <u>n</u>-label progressive diagram should have at least as many vertices as an <u>n</u>-bay folded tree, and it is on this ground that we make our conjecture.

We now proceed to carry out the program sketched above. Since all the vertex diagrams considered in previous sections were trees, we first present an example of a vertex diagram that is not a tree (Fig. 9). (Note the use of the loop in the wire <u>W</u> of Fig. 9 to indicate that the wire <u>W</u> does not touch the wire <u>V</u>.)

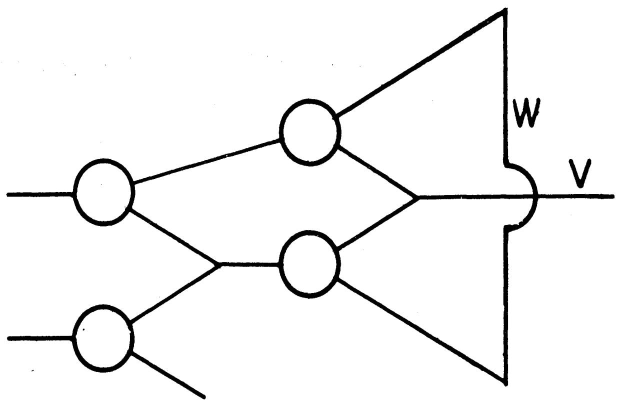

Fig. 9

Some of the concepts already introduced in connection with trees must be generalized to apply to arbitrary vertex diagrams. First, we require a more general method of describing the way states of wires are determined. To accomplish this we define the notion of the connection of two wires.

We say that two wires are directly connected when they touch each other. A vertex is directly connected to a wire whenever the wire is either its vertex-input, or its upper (lower) vertex-output when the vertex is in the upper (lower) setting. Two wires W and W' are connected if there is a sequence of wires and vertices $X_1, \ldots, X_n$ ($n \geq 1$) such that W is X_1, W' is X_n, and such that, for each $i < n$, X_i is directly connected to X_{i+1}. By taking $n = 1$ we see that any wire is always connected to itself. The sequence $X_1, \ldots, X_n$ is a connection of W and W'. Thus, whether two wires in a diagram are connected usually depends upon the settings of some of the vertices of the diagram.

An n-label complete decoding vertex diagram (with designated diagram input and diagram outputs) is a vertex diagram in which the following hold.

1. Each vertex V has exactly one label from a set of n distinct labels.

2. For each label there is at least one vertex with that label.

3. There is exactly one wire K designated as the diagram input.

4. A diagram state is a definite assignment of the vertex settings such that (a) all the vertices with the same label are set the same, and (b) a wire is in state 1 if and only if it is connected to the diagram input.

5. For each diagram state, there is at least one wire which is in state 1 in that diagram state and in state 0 in every other diagram state, and for each diagram state one such wire is designated (arbitrarily, if there is more than one) as a diagram output of the diagram.

The diagram input is in state 1 in any diagram state since it is always connected to itself. Since there are n labels for the vertices, there are 2^n diagram states, and, therefore, 2^n diagram outputs. For any diagram output, (Q), let $S(Q)$ be the diagram state in which Q is in state 1.

Sometimes in these diagrams a vertex may have the state of its vertex-input depend on the state of one of its vertex-outputs. For example, consider the uppermost vertex of Fig. 10. When it is in its lower setting, the state of its vertex input Q_2 depends upon the state of its lower vertex output. For this reason the terms "vertex-input" and "vertex-output" are not as appropriate for the more general class of vertex diagrams as they are for the class of trees.

Obviously, trees are vertex diagrams, and by Theorem 1 n-bay folded trees are complete decoding n-label vertex diagrams.

For an arbitrary connection $X_1,\ldots,X_p$, if X_k is directly connected to X_m for $m > k + 1$, then $X_{k+1},\ldots,X_{m-1}$ are superfluous; the sequence with them deleted is still a connection. It is easy to see, therefore, that in a diagram state, S, if there is a connection between W and W', then there is a connection between them in S without any superfluous vertices or wires. A chain of a diagram output Q is a sequence X_1, $\ldots,X_p$ of wires and vertices which in $S(Q)$ is a connection of the input K and Q without any superfluous elements. Obviously, the chain as defined in Section 2 is a chain in this sense.

If $X_1,\ldots,X_m$ is a chain C (where X_1 is the diagram input K and X_m is a diagram output), then for any i,j such that $i < j$ we say that X_i is earlier than X_j in C. If X_i is a vertex, then X_{i-1} and X_{i+1} are wires of the vertex, one being the vertex-input of X_i and the other being one of the vertex-outputs of X_i; we say that X_{i-1} is the early wire and X_{i+1} the late wire of the vertex X_i in the chain C. If the early wire of a vertex in a chain is the vertex-input, then we say the vertex is oriented forward in the chain; if the early wire is a vertex-output, then the vertex is oriented backward. For example, the vertices labeled P_2 in the chains of Q_1 and Q_2 in Fig. 10 are backward, whereas the vertex labeled P_2 in the chains of Q_3 and Q_4 is forward in both those chains. The vertex labeled P_1 is forward in all chains.

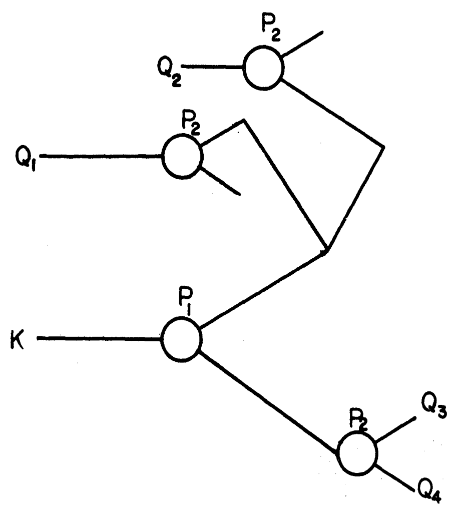

Fig. 10

A progressive diagram is a complete decoding vertex diagram in which each output $\underline{Q}$ has at least one chain in which all vertices are forward. The folded tree is a progressive diagram while Fig. 10 is a complete decoding vertex diagram which is not progressive.

An output $\underline{Q}$ of a progressive diagram may have more than one chain in which all vertices are forward. It is convenient to pick out for each output $\underline{Q}$ of a progressive diagram one such chain and refer to it as $\underline{C}(\underline{Q})$, or the selected chain of the output.

Where $\underline{Q}$ and $\underline{Q}'$ are two distinct diagram outputs of a complete decoding vertex diagram, we define $\underline{V}(\underline{Q},\underline{Q}')$ to be the latest vertex $\underline{V}$ in $\underline{C}(\underline{Q})$ whose early wire is in state 1 in $\underline{S}(\underline{Q}')$ and whose late wire is in state 0 in $\underline{S}(\underline{Q}')$. Such a vertex must exist since the diagram input $\underline{K}$ is in state 1 in $S(Q')$ but $\underline{Q}$ is in state 0 in $\underline{S}(\underline{Q}')$; the latest wire of $\underline{C}(\underline{Q})$ which is in state 1 in $\underline{S}(\underline{Q}')$ must be the early wire of a vertex in $\underline{C}(\underline{Q})$. For example, in Fig. 11 $\underline{V}(\underline{Q}_1,\underline{Q}_4)$ is the vertex labeled $\underline{P}_1$ in $\underline{C}(\underline{Q}_1)$ and $\underline{V}(\underline{Q}_4,\underline{Q}_1)$ is the vertex labeled $\underline{P}_2$ in $\underline{C}(\underline{Q}_4)$. This example shows, incidentally, that $\underline{V}(\underline{Q},\underline{Q}')$ and $\underline{V}(\underline{Q}',\underline{Q})$ are not always identical.

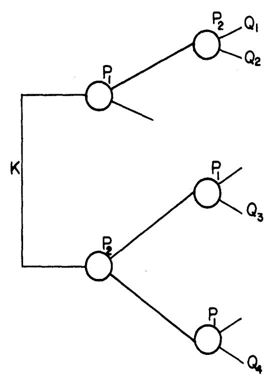

Fig. 11

Whenever A is any set of outputs of a progressive diagram we let $D(A)$ be the set of just those vertices $V(Q,Q')$ where Q and Q' are both in A. If A has only one output Q, then $D(A)$ is the null set, since there is no $V(Q,Q)$. For any set A and any output Q not in A, $A+\{Q\}$ is the set whose members are Q and all the members of A.

Theorem 16: If A is a set of one or more outputs of a progressive diagram, and if Q is an output not in A, then there is a vertex in $D(A+\{Q\})$ which is not in $D(A)$.

Proof: Let V be the latest vertex $V(Q,Q')$ in $C(Q)$ where Q' is in A. We shall prove that V is not in $D(A)$, from which Theorem 16 follows directly since V must be in $D(A+\{Q\})$.

We shall use the reductio ad absurdum method. Suppose that V is in $D(A)$. Then there are diagram outputs Q'' and Q''' in A such that V is $V(Q'',Q''')$. Let x be the early wire of V in $C(Q)$ and y the late wire. Since the diagram is progressive and since V is in $C(Q)$, x is the vertex-input and y is a vertex-output of V. Let z be the other vertex-output of V. Since V is $V(Q'',Q''')$, x is in $C(Q'')$ and either y or z is in $C(Q'')$

Case I. y is in $C(Q'')$. Then y is in state 1 in $S(Q'')$. The latest wire in $C(Q)$ which is in state 1 in $S(Q'')$, then, can be no earlier than y, and hence $V(Q,Q'')$ is later than V in $C(Q)$, contrary to the original stipulation that V be the latest such vertex in $C(Q)$.

Case II. z is in $C(Q'')$. Since V is $V(Q'',Q''')$, x must be in state 1 in $S(Q''')$, and z in state 0. This means that the setting of the vertex V in $S(Q''')$ must be such that y is in state 1 in $S(Q''')$. Reasoning as in Case I, then we can show that $V(Q,Q''')$ is later than V in $C(Q)$, which is likewise contradictory. This completes the proof of Theorem 16.

Theorem 17: In a progressive diagram, if A is a set of k outputs then there are at least k-1 vertices in $D(A)$.

Proof: Let $A_2,\dots,A_k$ be a sequence of subsets of A such that A_k is A, A_i has exactly i members, and, if $2 \leqq i \leqq k-1$, A_i is a proper subset of A_{i+1}. Thus, A_{i+1} has just one member besides those of A_i. There is at least one vertex in $D(A_2)$; and by Theorem 16 $D(A_{i+1})$ has at least one more vertex than $D(A_i)$. Hence, $D(A_k)$ has at least k-1 vertices.

Theorem 18: In the class of n-label progressive diagrams the n-bay folded tree has a minimal number of vertices.

Proof: In an n-label complete decoding vertex diagram there are 2^n outputs. If A is the set of all these outputs, the D(A) must have at least 2^n-1 vertices. Hence, the diagram must have at least 2^n-1 vertices which is the number of vertices in the n-bay folded tree.

7. A GENERALIZATION OF THE FOLDED TREE

In this section we shall consider generalized folded trees containing vertices, all of which have the same arbitrary number of vertex outputs and possible settings. A vertex with m vertex-outputs is called an m-order vertex and its m settings are the first setting, the second setting, ..., the mth setting. The ith vertex output is the ith right-hand wire from the top. (See Fig. 12.)

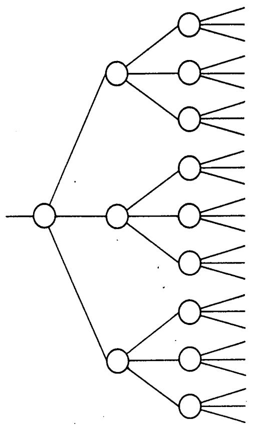

Fig. 12

A generalized n-bay folded tree containing vertices of order m, called an n-bay m-order folded tree, obviously contains

$$\sum_{x=1}^{n} m^{x-1} = (m^n-1)/(m-1)$$

vertices. Thus an n-bay folded tree as previously defined has order 2. The definition of "complete decoding" is the same as that given in Section 2, and the generalization of Theorem 1 goes through quite easily.

It is not difficult to see what sort of physical realization an $\underline{m}$-order vertex can have, either in relay circuits or in electronic digital computing circuits. In relay circuits the $\underline{m}$-order vertex represents a single-pole $\underline{m}$-throw switch, e.g., a stepping switch. In electronic digital computing circuits the $\underline{m}$-order vertex can represent an arrangement of $\underline{m}$ conjunction elements generalized from the arrangement of Fig. 2.

As in Section 2 we ask the question, for a given loading sequence, $\underline{a}_1, \ldots, \underline{a}_n$, is there an $\underline{n}$-bay $\underline{m}$-order folded tree having $\underline{a}_1$ vertices labeled $\underline{P}_1, \ldots,$ and $\underline{a}_n$ vertices labeled $\underline{P}_n$? We have a generalized condition of "admissibility" which we can prove to be necessary and which we conjecture to be sufficient if the sequence contains no zeros. Our generalized condition of admissibility involves four conditions, as compared with only three in Section 3.

A sequence satisfies the unit condition if there is a 1 somewhere in the sequence. It satisfies the total sum condition if the sum of all the terms is equal to

$$(\underline{m}^{p}-1)/(\underline{m}-1) \quad = \quad \sum_{x=1}^{p} \underline{m}^{x-1}$$

where $\underline{p}$ is the number of non-zero terms. A sequence $\underline{S}$ satisfies the partial sum condition if, for each $\underline{k} \leq \underline{p}$, the sum of the first $\underline{k}$ terms of $\underline{M}(\underline{S})$ is greater than or equal to

$$(\underline{m}^{k}-1)/(\underline{m}-1) \quad = \quad \sum_{x=1}^{k} \underline{m}^{x-1} \quad .$$

It satisfies the congruence condition if, for each term $\underline{a}_k$, $\underline{a}_k \equiv 1(\text{mod } \underline{m}-1)$, i.e., $\underline{a}_k-1$ is divisible by $\underline{m}-1$. A sequence is admissible if it satisfies all four conditions. Note that admissibility as defined in Section 3 is a special case ($\underline{m} = 2$) of this more general notion of admissibility, for any sequence of integers satisfies the congruence condition when $\underline{m} = 2$.

Theorem 19: The loading sequence $\underline{S}(1,1)$ of an $\underline{n}$-bay $\underline{m}$-order folded tree is an admissible sequence of $\underline{n}$ non-zero terms.

Proof: Obviously, $\underline{S}(1,1)$ must satisfy the unit condition and the total sum condition. The reader can reread Theorem 8 and its proof and see that it very easily generalizes from the case ($\underline{m} = 2$) to prove that $\underline{S}(1,1)$ satisifes the partial sum condition for any $\underline{m}$. To demonstrate that $\underline{S}(1,1)$ satisfies the congruence condition we consider any $\underline{n}$-bay $\underline{m}$-order folded tree having $\underline{a}_k(1,1)$ vertices labeled $\underline{P}_k$. Suppose that for each $\underline{h} \leqq \underline{n}$ there are $\underline{k}_h$ vertices labeled $\underline{P}_k$ in the $\underline{h}$th bay. There are $\underline{m}^n$ chains and each chain must have exactly one vertex labeled $\underline{P}_k$.

But a vertex in the $\underline{h}$th bay is on exactly $\underline{m}^{n-h+1}$ chains. Hence (1) the number of chains is equal to

$$\sum_{h=1}^{n} \underline{k}_h \underline{m}^{n-h+1} = \underline{m}^n .$$

By elementary number theory we know that for any non-negative integer $\underline{x}$, $\underline{m}^x \equiv 1(\text{mod } \underline{m}-1)$. Thus, for each $\underline{h}$, $\underline{k}_h\underline{m}^{n-h+1} \equiv \underline{k}_h(\text{mod } \underline{m}-1)$. Hence (2)

$$\sum_{h=1}^{n} \underline{k}_h\underline{m}^{n-h+1} \equiv \sum_{h=1}^{n} \underline{k}_h(\text{mod } \underline{m}-1).$$

From (1) and (2), since $\underline{m}^n \equiv 1(\text{mod } \underline{m}-1)$, it follows that

$$\underline{a}_k(1,1) = \sum_{h=1}^{n} \underline{k}_h \equiv 1(\text{mod } \underline{m}-1) .$$

This completes our proof of Theorem 19.

APPENDIX

Interchange and the Folded Tree

The phrase "folded tree" is appropriate because a folded tree can be obtained from the standard tree of Section 2 by the technique of "folding." That technique, perhaps better described as "interchange," can be defined as follows. An n-bay labeled-tree $T'(1,1)$ results from an n-bay labeled-tree $T(1,1)$ by an interchange of P_h and P_k in the minor tree $T(i,j)$ when all the vertices labeled P_h in $T(i,j)$ are labeled P_k in $T'(i,j)$ and vice versa, all other vertices of $T'(1,1)$ being labeled the same as in $T(1,1)$.

By referring to the definition of "folded tree", it is not difficult to see (1) that if $T'(1,1)$ is obtained from $T(1,1)$ by interchange, then $T'(1,1)$ is a folded tree if and only if $T(1,1)$ is.

It can also be shown (2) that for any n-bay folded trees $T(1,1)$ and $T'(1,1)$ having the same set of labels, $T'(1,1)$ can be obtained from $T(1,1)$ by a sequence of interchanges. For suppose $T_1(1,1)$, $T_2(1,1)$, ..., is a sequence of distinct n-bay labeled-trees where $T_1(1,1)$ is $T(1,1)$ and where $T_{x+1}(1,1)$ is obtained from $T_x(1,1)$ by the following process. Having ordered all the vertices of $T_x(1,1)$ in the sequence $V_x(1,1)$, $V_x(2,1)$, $V_x(2,2)$, $V_x(3,1)$, $V_x(3,2)$, ..., we consider the first vertex $V_x(i,j)$ which has a label different from the corresponding vertex $V'(i,j)$ of $T'(1,1)$. Suppose that P_h and P_k are the labels of $V_x(i,j)$ and $V'(i,j)$, respectively. Then $T_{x+1}(1,1)$ results from $T_x(1,1)$ by interchange of P_h and P_k in $T_x(i,j)$. It is easily seen that P_k must label at least two vertices in $T_x(i,j)$, so the interchange can always be made if $T_x(1,1)$ is not identical with $T'(1,1)$. Obviously, then the label of $V_{x+1}(i,j)$ and the labels of all vertices of $T_{x+1}(1,1)$ which precede $V_{x+1}(i,j)$ in the ordering of vertices mentioned above are the same as the labels of the corresponding vertices of $T'(1,1)$. It is not difficult to see that the sequence $T_1(1,1)$, $T_2(1,1)$, ... has a last member $T_q(1,1)$ which must be $T'(1,1)$.

Since a standard tree is a folded tree, it follows from (1) and (2) that a necessary and sufficient condition that an n-bay labeled-tree with labels P_1, ..., P_n be a folded tree is that it be obtainable from the n-bay standard tree by a sequence of interchanges.

Everything asserted in this appendix is true also of trees all of whose vertices are of order $m > 2$.

BIBLIOGRAPHY

[I] Shannon, C. E., "The Synthesis of Two-Terminal Switching Circuits," Bell System Technical Journal, Vol. 28, pp. 59-98, January, 1949.

[II] Burks, A. W., R. McNaughton, C. H. Pollmar, D. W. Warren, and J. B. Wright, Complete Decoding Nets: General Theory and Minimality, ERI-1828-1-T, Ann Arbor, July 15, 1954 (Burroughs Corporation, Research Center, Paoli, Pennsylvania). To be published.

[III] Keister, W., A. E. Ritchie, and S. H. Washburn, The Design of Switching Circuits, New York, 1951.

[IV] Burks, A. W., C. H. Pollmar, D. W. Warren, and J. B. Wright, "Minimal Switch Theory and the Folded Tree," Vol. III, Language Conversion For Digital Computers, ERI Project M828, Ann Arbor, 1 March 1953. (Copies may be obtained by writing Mr. George W. Patterson, Burroughs Corporation, Research Center, Paoli, Pennsylvania.)