COMMUNICATION SCIENCES AND ENGINEERING

X. PROCESSING AND TRANSMISSION OF INFORMATION

Academic and Research Staff

Prof. Peter Elias
Prof. Robert G. Gallager
Prof. Robert S. Kennedy
Prof. Jeffrey H. Shapiro
Dr. Estil V. Hoversten
Dr. Horace P. H. Yuen

Graduate Students

Vincent Chan
Samuel J. Dolinar, Jr.
Richard A. Flower
Robert J. Fontana
Robert P. Gallant
John G. Himes
Pierre A. Humblet
Thomas L. Marzetta
Gerard P. Massa
Robert J. Metzger
Andrew Z. Muszynsky
Nam-soo Myung
Howard F. Okrent
John J. Poklemba
Stanley R. Robinson
Alexander S. Theoduru
Victor Vilnrotter
Kohei Yamada

A. OPTICAL CHANNELS

1. UNIQUENESS OF THE OPTIMUM RECEIVER FOR THE M-ARY PURE STATE PROBLEM

National Aeronautics and Space Administration (Grant NGL 22-009-013)

Robert S. Kennedy

We have seen[1] that the necessary and sufficient conditions for a set of operators π_i

$$\pi_i \geq 0$$
$$\Sigma \pi_i = I \tag{1}$$

to be the optimum decision operators are that the operator λ defined by $\lambda = \sum_i p_i \rho_i \pi_i$ satisfy

$$\lambda = \lambda^+ \tag{2}$$

and

$$\lambda - p_i \rho_i \geq 0 \qquad \text{all } i.$$

We have further shown that, on the space spanned by the m linearly independent message vectors $\langle u_i$, the optimum π_i are orthogonal projectors.[1] That is,

$$\pi_i \pi_j = \delta_{ij} \pi_i. \tag{3}$$

We now claim that the optimum π_i on that space are unique.

PROOF: Let us suppose to the contrary that there are two optimum measurements $\{\tilde{\pi}_i\}$ and $\{\hat{\pi}_i\}$ each satisfying Eqs. 1 and 2. Then any measurement $\{\pi'_i\}$ with

$$\pi'_i \equiv \mu\tilde{\pi}_i + (1-\mu)\hat{\pi}_i \qquad 0 \leq \mu \leq 1 \tag{4}$$

also satisfies Eqs. 1 and 2 and hence is also an optimum measurement.

We have seen that the optimum measurement operators are orthogonal projectors (Eq. 3). Thus it must be that for all values of μ in the interval $[0, 1]$ the operators defined by Eq. 4 must satisfy $\pi'_i\pi'_j = \delta_{ij}\pi'_i$ or equivalently

$$\mu^2\tilde{\pi}_i\tilde{\pi}_j + (1-\mu)^2\hat{\pi}_i\hat{\pi}_j + \mu(1-\mu)[\tilde{\pi}_i\hat{\pi}_j + \hat{\pi}_i\tilde{\pi}_j] = \delta_{ij}[\mu\tilde{\pi}_i + (1-\mu)\hat{\pi}_i].$$

Since the $\{\hat{\pi}_i\}$ and the $\{\tilde{\pi}_i\}$ satisfy Eq. 3, this may also be stated as

$$\mu(1-\mu)[\tilde{\pi}_i\hat{\pi}_j + \hat{\pi}_i\tilde{\pi}_j] = \delta_{ij}\mu(1-\mu)[\tilde{\pi}_i + \hat{\pi}_i]$$

or

$$\tilde{\pi}_i\hat{\pi}_j + \hat{\pi}_i\tilde{\pi}_j = \delta_{ij}(\hat{\pi}_i + \tilde{\pi}_i). \tag{5}$$

Next, we show that Eq. 5 implies that the set of $\{\hat{\pi}_i\}$ and the set of $\{\tilde{\pi}_i\}$ are identical. To this end, we recall that the $\{\hat{\pi}_i\}$ are a set of m commuting projection operators on an m-dimensional space. This implies that they possess a common set of eigenvectors, that each eigenvector is in the null space of all but one projection operator, and that for that one operator its eigenvalue is one. Hence we must have

$$\tilde{\pi}_i = |\phi_i\rangle\langle\phi_i|, \tag{6}$$

where the $\langle\phi_i|$ are an orthonormal set. Similarly,

$$\hat{\pi}_i = |\psi_i\rangle\langle\psi_i|, \tag{7}$$

where the $\langle\psi_i|$ are also an orthonormal set.

Introducing Eqs. 6 and 7 in Eq. 5 yields

$$|\phi_i\rangle\langle\phi_i|\psi_j\rangle\langle\psi_j| + |\psi_i\rangle\langle\psi_i|\phi_j\rangle\langle\phi_j| = 0 \qquad \text{for } i \neq j.$$

Operating on $|\phi_i\rangle$ with this identity yields

$$|\phi_i\rangle\langle\phi_i|\psi_j\rangle\langle\psi_j|\phi_i\rangle = 0.$$

Hence $\langle\phi_i|\psi_j\rangle = 0$ for $i \neq j$. That is, each of the $|\psi_j\rangle$ are orthogonal to all but one of the $\{|\phi_i\rangle\}$ and that one is $|\phi_j\rangle$. Consequently the $|\psi_j\rangle$ may be expressed as $|\psi_j\rangle = c_j|\phi_j\rangle$. Moreover, since the $|\psi_j\rangle$, as well as the $|\phi_j\rangle$, are normalized, the magnitude of c_j must be one. Thus $|\psi_j\rangle\langle\psi_j| = |\phi_j\rangle\langle\phi_j|$, and the $\hat{\pi}_j$ and $\tilde{\pi}_j$ are identical, as was claimed.

References

1. R. S. Kennedy, "On the Optimum Quantum Receiver for the M-ary Linearly Independent Pure State Problem," Quarterly Progress Report No. 110, Research Laboratory of Electronics, M. I. T., July 15, 1973, pp. 142-146.

B. COMPLEXITY OF NETWORKS AND ALGORITHMS

1. COMPLEXITY OF ACCEPTORS FOR PREFIX CODES

U. S. Army Research Office – Durham (Contract DAHC04-71-C-0039)

Donna J. Brown

Introduction

In the field of coding, a code is simply a (one-to-one) correspondence between some sort of message "words" and a set of "code words" (strings of characters chosen from some alphabet). For instance, the message that is transmitted might be

0 0 0 1 0 0 1 0 1 0 1 1 0 1 1 1 0 0 1 0 1 1 1,

with the encoding

a: 00	e: 1100
b: 010	f: 1101
c: 011	g: 111
d: 10	

This message would be decoded as

0 0 | 0 1 0 | 0 1 0 | 1 0 | 1 1 0 1 | 1 1 0 0 | 1 0 | 1 1 1,

which gives

a b b d f e d g.

There are several desirable properties for a code. One is that the code be uniquely decodable; i. e., no string of code words may represent more than one message. Furthermore, we would in general like to allow code words of different lengths, and to avoid the extra work involved in recognizing start and stop bits that indicate code word boundaries. To do this, the code must have the property that the code word boundaries are obvious. A code for which it is possible to detect the end of a code word without "looking ahead" to succeeding symbols is called an instantaneous code. Obviously, in such a case no code word can be a prefix of any other code word; hence, such a code is called a prefix code. Such codes have been studied in some detail, and the Kraft inequality[1] states that a necessary and sufficient condition for the existence of an instantaneous code with word lengths $\ell_1, \ell_2, \ldots, \ell_r$ is

$$\sum_{i=1}^{r} s^{-\ell_i} \leq 1,$$

where s is the size of the code alphabet. Observe that any prefix code can be represented by a tree, as illustrated in Fig. X-1.

Given the prefix code C_1, containing 8 code words:

a:	0	e:	11001
b:	100	f:	11010
c:	101	g:	11011
d:	11000	h:	111

This code can be represented by the tree T_1.

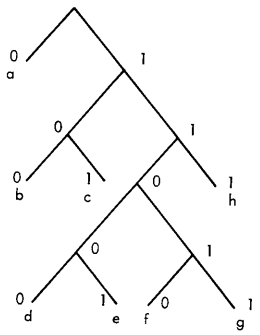

Fig. X-1. Tree representation of code C_1.

Another desirable property, for speed of transmittal, is that the code have the shortest possible word lengths. Thus, if the i^{th} message word (and its corresponding code word) has independent probability p_i of occurring in any given r-word transmission, we want a prefix code whose vector of word lengths $(\ell_1, \ell_2, \ldots, \ell_r)$ for any such transmission will minimize the average word length $\sum_{i=1}^{r} p_i \ell_i$. In other words, we want to consider a vector of word lengths having the minimum average word length among all vectors that satisfy the Kraft inequality. Such a code is called a compact prefix code. In particular, we shall look at binary codes (s = 2) for which the Kraft condition says $\sum_{i=1}^{r} 2^{-\ell_i} \leq 1$. In 1952, Huffman[2] described a procedure for the construction of a compact prefix code, given the (independent) probabilities of the individual words. It is possible, however, to construct fundamentally different codes for which the set of code-word lengths is the same and hence the code is still compact prefix. For example, in Fig. X-2, even if labeling is ignored, trees T_2 and T_3 are not isomorphic, and thus the codes that they represent are fundamentally different even though the set of word lengths is the same.

Set of word lengths: 2, 2, 3, 3, 3, 3

Trees T_2 and T_3 represent codes C_2 and C_3 as indicated:

Code C_2	Code C_3
a: 00	a: 01
b: 01	b: 10
c: 100	c: 000
d: 101	d: 001
e: 110	e: 110
f: 111	f: 111

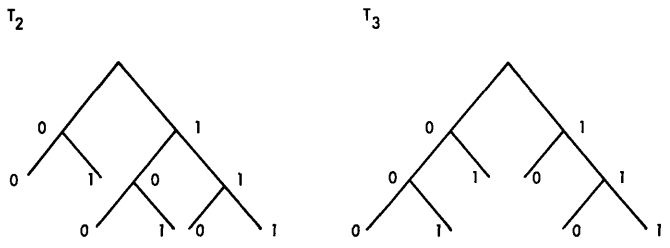

Fig. X-2.
Tree representations of codes C_2 and C_3.

We shall consider the problem of detecting the ends of legal code words in a string of code symbols (rather than actually decoding the message). A simple model for such a detector is a finite-state machine with one accepting state, as shown in Fig. X-3.

Given the prefix code C_4:

a: 00	g: 11010
b: 010	h: 11011
c: 011	i: 11100
d: 100	j: 11101
e: 101	k: 11110
f: 1100	l: 11111

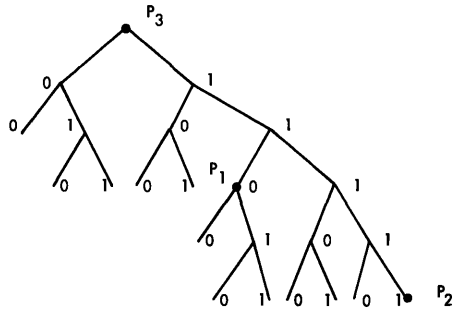

Fig. X-3.
Tree representation of code C_4.

This code can be represented by the tree T_4.
Code C_4 can then be "recognized" by the finite-state machine shown in Fig. X-4.

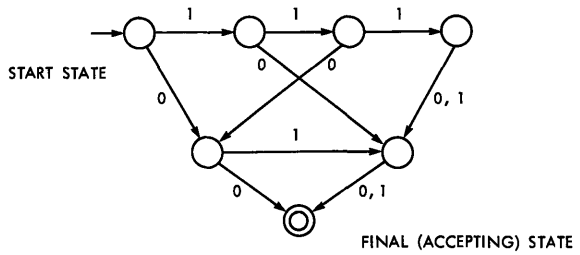

Fig. X-4. Machine that "recognizes" code C_4.

For a given (arbitrary) set of messages and an arbitrary maximum code-word length, we shall develop and analyze a method of code construction in which the word lengths

are those of a compact prefix code but the decoding will be optimal, in that a finite-state end-of-word detector will have a minimal number of states as a function of n. We refer to such a detector as an acceptor.

Because we analyze a code through its tree representation, we shall state some preliminary definitions and make some observations. First, we use the term "tree" to refer to a full binary tree; i.e., a tree in which each node has 0 or 2 sons.

Definition: The depth of a node i, represented d(i), is the number of nodes on the path from the root to node i, not counting the root node.

For example, in Fig. X-3 node P_1 has depth 3, node P_2 has depth 5, and node P_3 has depth 0.

Definition: An R-set is a "set" of elements in which repetitions are allowed. The notation $[i_1, i_2, \ldots, i_n]$ is used to represent the n (not necessarily distinct) elements of an R-set, and the abbreviated notation $\left[i_1^{a_1}, i_2^{a_2}, \ldots, i_k^{a_k}\right]$ indicates that among the $n = a_1 + a_2 + \ldots + a_k$ elements in the R-set, the element i_j appears a_j times.

A simple inductive argument leads to the following theorem.

Theorem 1. For any (full binary) tree T, $\sum_{\substack{i \text{ a terminal} \\ \text{node of T}}} 2^{-d(i)} = 1$.

For example, in Fig. X-1 we have

$$\sum_{\substack{i \text{ a terminal} \\ \text{node of } T_1}} 2^{-d(i)} = 2^{-d(a)} + 2^{-d(b)} + \ldots + 2^{-d(h)}$$

$$= 2^{-1} + 2^{-3} + 2^{-3} + 2^{-5} + 2^{-5} + 2^{-5} + 2^{-5} + 2^{-3}$$

$$= 1.$$

Definition: A length set (of size k) is an R-set of (k) positive integers that meets the Kraft inequality with equality.

Now another inductive argument leads us to the following result:

Theorem 2. To any length set of size k there corresponds at least one tree with k leaves.

Recalling the example in Fig. X-2, we observe that more than one tree may correspond to a given length set. Thus, applying Theorem 2, we conclude:

Theorem 3. The number of unlabeled trees with k leaves is greater than the number of length sets of size k (for $k \geq 6$).

Theorems 1, 2, and 3 are well known and are given here for ease of reference (for example, see Abramson[3] and Gallager[4]).

Problem Statement

For a given set of messages and their assigned probabilities, Huffman's procedure[2] gives a method of computing a length set that is optimal in the sense that the average word length is minimized. Corresponding to a particular length set, however, there may be more than one code. Equivalently, as we have seen in Theorem 3, corresponding to a particular length set

$$\left[n^{a_n}, (n-1)^{a_{n-1}}, (n-2)^{a_{n-2}}, \ldots, 2^{a_2}, 1^{a_1}\right]$$

of maximum depth n, there may be more than one full binary tree. Let L(n) be the set of length sets of maximum depth n. Then, for any $\ell \in L(n)$, we are interested in the value of $S(\ell)$, where $S(\ell)$ is the smallest number of states in any finite-state acceptor that accepts a set of code words with length set ℓ. We shall show that

$$O\left(\frac{n^2}{\log n}\right) \leq \max_{\ell \in L(n)} S(\ell) \leq O(n^2).$$

Upper Bound

By brute force, simply allowing a finite-state machine to model the tree, we can recognize a length set $\left[1^{a_1}, 2^{a_2}, \ldots, n^{a_n}\right]$ in $(a_1+a_2+\ldots+a_n-1) + 1 = a_1 + a_2 + \ldots + a_n$ states, since $a_1 + \ldots + a_n - 1$ is the number of internal nodes in any corresponding tree and the one additional state is the final accepting state. To see how this works, consider the following machine that recognizes code C_4 and models tree T_4.

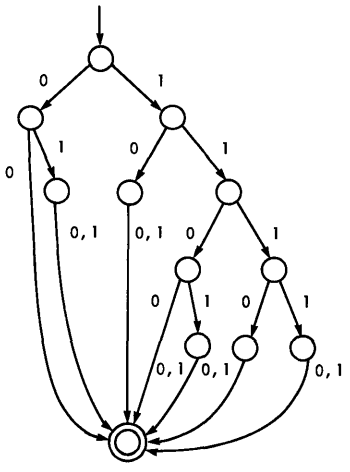

Fig. X-5. Acceptor for C_4 that models T_4.

(Notice that the machine in Fig. X-5 is not the simplest possible acceptor for C_4; the one in Fig. X-4 is simpler.) This method of machine construction causes the number

of states to grow exponentially with the depth of the tree. And, given an arbitrary code, the growth in minimal number of states in an acceptor may indeed be exponential. But in general we are free to design codes, and we shall present a procedure to construct, from any length set, a code for which the number of states in a recognizing machine grows only as the square of the length of the longest code word. We then see that this is "almost" the best possible result.

In order to obtain the n^2 upper bound, we define what we mean by a tree in "block form." We use the term block to refer to a (full binary) subtree of uniform depth. Figure X-6 shows examples of blocks. Thus each leaf of an arbitrary tree is a leaf in

Fig. X-6. Examples of blocks.

some block. And for each tree T there is a corresponding trimmed tree T', which is obtained from T by removing all blocks and leaving only the roots of the blocks (see Fig. X-7). A tree is in block form if there are no two blocks of the same size

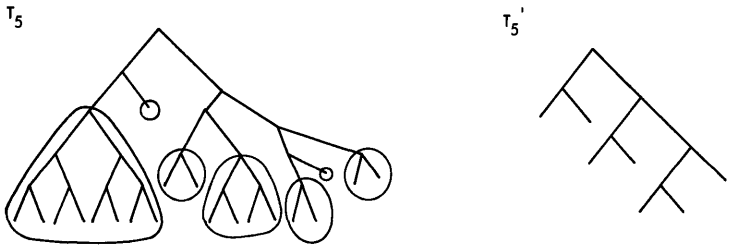

Fig. X-7. Tree T_5 and its corresponding trimmed tree T_5'.

at the same depth. In other words, a tree in block form has its leaves grouped so that it forms maximum size blocks. In Fig. X-8, T_6 is not in block form because blocks B_1 and B_2 are of the same size and at the same depth. In tree T_7, however, B_1 and B_2 have been combined, and T_7 is in block form. A code is in block form if its corresponding tree is in block form.

Perhaps the best way to understand what is meant by block form is, given a length set, to see how a corresponding block-form tree can be constructed. Suppose there are a_i leaves at depth i. Let the binary expansion of a_i be

$$a_i = b_{i0} \cdot 2^0 + b_{i1} \cdot 2^1 + b_{i2} \cdot 2^2 + \ldots + b_{ip} \cdot 2^p,$$

where $b_{ij} = 0$ or 1. This tells us immediately that for $b_{ij} = 1$ there is a block with 2^j leaves at depth i, and also that the trimmed tree will have a leaf at

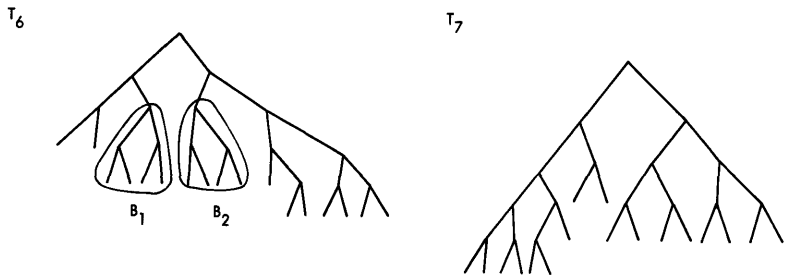

Fig. X-8. Example of a tree T_6 not in block form and a tree T_7 in block form.

depth i - j. Consider as an example the length set $[2, 4^3, 5^8, 6^{11}, 7^{12}, 8^{12}]$, and suppose that T_8 is a tree representing this length set.

Leaves in T_8		Corresponding Leaves in T_8'	Leaves in T_8		Corresponding Leaves in T_8'
2^1	→	2	6^8	→	3
4^1	→	4	7^4	→	5
4^2	→	3	7^8	→	4
5^8	→	2	8^4	→	6
6^1	→	6	8^8	→	5
6^2	→	5			

We may now construct, in any way desired, a (trimmed) tree T_8' with length set $[2^2, 3^2, 4^2, 5^3, 6^2]$. The block-form tree can now be obtained from T_8' by adding on to the appropriate leaves of T_8' the blocks removed in the trimming process (see Figs. X-9 and X-10).

We now deduce an upper bound on the number of states required to recognize a given length set. We have already seen, for a length set $\left[1^{a_1}, 2^{a_2}, \ldots, n^{a_n}\right]$, how we can

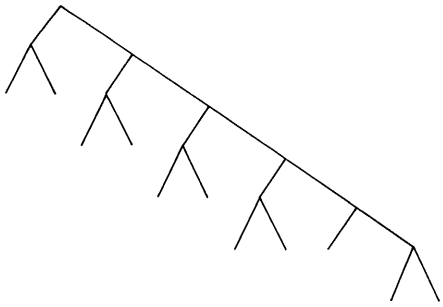

Fig. X-9. Trimmed tree T'_8.

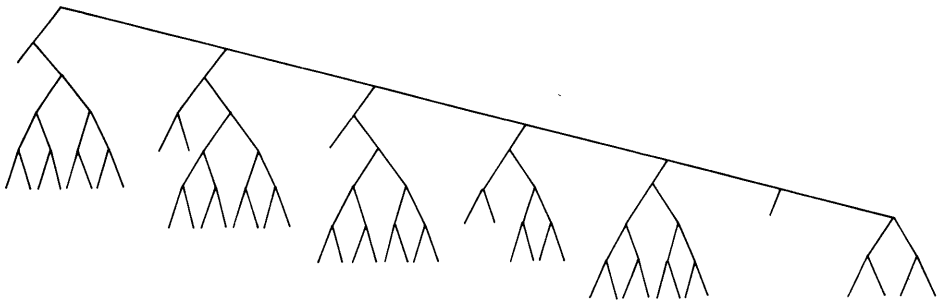

Fig. X-10. Block-form tree corresponding to T'_8.

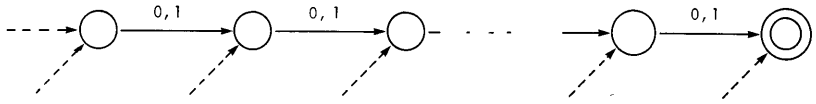

Fig. X-11. Block-recognizing machine.

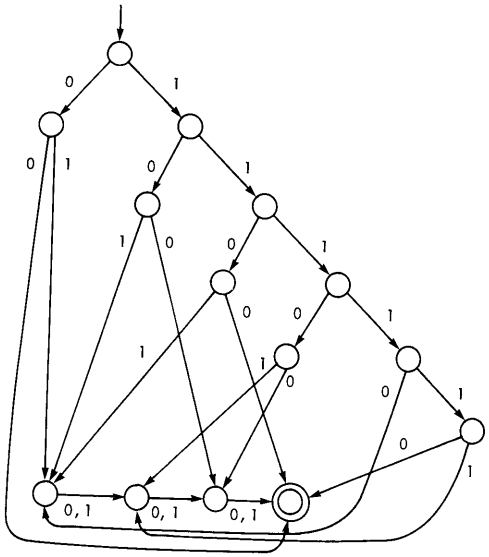

Fig. X-12. Acceptor for the code represented by the block-form tree corresponding to T_8.

construct a corresponding tree T in block form. Since T has $a_1 + a_2 + \ldots + a_n - 1$ internal nodes, its trimmed tree T' has $(a_1+a_2+\ldots+a_n-1) - r$ internal nodes, where r is the total number of nonleaf nodes contained in blocks of T. But notice that a machine of the form shown in Fig. X-11 can recognize any block of depth less than or equal to p. Thus, if we combine this block-recognizing machine with the machine that models T', then we shall have a machine that will recognize T and has only

$$a_1 + a_2 + \ldots + a_n - r + p \quad \text{states,}$$

where p, the depth of the biggest block, is $\lfloor(\log_2(\max(a_1, a_2, \ldots, a_n))\rfloor$, and r is the number of nonleaf nodes in blocks of T. As an example of this construction, Fig. X-12 illustrates an acceptor for the code represented by the block-form tree in Fig. X-10.

Having obtained an upper bound on the number of states required to recognize a given length set, we now want to analyze how good (or bad) this upper bound actually is. An inductive proof gives our main result:

Theorem 4: To any length set with maximum length n ($n \geq 2$) there corresponds a tree of depth n that can be recognized by a finite-state machine with at most $n(n+1)/2 = O(n^2)$ states.

Although the details of the proof of this theorem are a bit tedious, we can give an intuitive argument that shows that block form does actually lead to an upper bound of $O(n^2)$ states. Given a length set $\left[1^{a_1}, 2^{a_2}, 3^{a_3}, \ldots, n^{a_n}\right]$, a corresponding block-form tree has $2(a_1+a_2+\ldots+a_n) - 1$ nodes. Consider again a_i written in binary expansion,

$$a_i = b_{i0} \cdot 2^0 + b_{i1} \cdot 2^1 + b_{i2} \cdot 2^2 + \ldots + b_{ip} \cdot 2^p.$$

We see that the trimmed tree has $\sum_{\substack{1\leq i\leq n\\ 0\leq j\leq p}} b_{ij}$ leaves and therefore $\sum_{\substack{1\leq i\leq n\\ 0\leq j\leq p}} b_{ij} - 1$ internal nodes. Thus T can be recognized in

$$\leq \sum_{\substack{1\leq i\leq n\\ 0\leq j\leq p}} b_{ij} - 1 + p + 1 \quad \text{states}$$

$$\leq \sum_{\substack{1\leq i\leq n\\ 0\leq j\leq p}} b_{ij} + p \quad \text{states}$$

$$\leq n(p+1) + p \quad \text{states}$$

$$\leq n^2 + 2n \quad \text{states,}$$

since $p \leq n$. We conclude, therefore, that T can be recognized by a finite-state machine with at most $O(n^2)$ states, thereby giving the upper bound $\max_{\ell \in L(n)} S(\ell) \leq O(n^2)$.

Lower Bound

We have succeeded in obtaining an upper bound of $O(n^2)$ states, and a counting argument now gives a lower bound of $O(n^2/\log n)$. We obtain this lower bound by comparing the number of length sets of depth n and the number of acceptors with at most k states. If there are not enough acceptors to recognize all of the length sets, then k must be a lower bound on the number of states required.

To obtain an upper bound on the number of acceptors with k states, suppose we have a sequence of k numbered states and want to determine how many machines can be constructed. Without loss of generality, let state number 1 be the start state and let state number k be the (final) accepting state. For each of the states 1, ..., k-1, there are k states to which we might move upon receiving a 1, and k to which we might move on receiving a 0. Thus there are k^2 possibilities for each of the k-1 states, which gives an upper bound of $k^{2(k-1)}$ possible machines.

Lemma. There are at most $k^{2(k-1)}$ finite-state machines with two input symbols and k states.

What we are really concerned with, however, is the number of acceptors with at most k states and the following theorem gives that.

Theorem 5. There are at most k^{2k} finite-state machines with two input symbols and at most k states.

Proof. (Number of machines with $\leq$k states) $= \sum_{i=1}^{k}$ (number of machines with i states)

$$\leq \sum_{i=1}^{k} i^{2(i-1)}$$

$$\leq k \cdot k^{2(k-1)}$$

$$< k^{2k}$$

The following theorem gives the number of length sets of depth n.

Theorem 6. The number of length sets of depth n is $2^{O(n^2)}$.

Proof. We show first that the number of depth n length sets $\leq 2^{n^2}$. Each length set

of depth n can be written in the form

$$\left[n^{a_n}, (n-1)^{a_{n-1}}, (n-2)^{a_{n-2}}, \ldots, 2^{a_2}, 1^{a_1} \right], \qquad 0 \leq a_i \leq 2^i.$$

We want to know in how many ways the $a_1, \ldots, a_n$ can be chosen. Certainly $a_i \leq 2^n$ for all i, and so in binary notation a_i has at most n digits. Thus $a_1 a_2 \ldots a_n$ has at most n^2 digits, and each of these n^2 digits can be chosen in two ways. So there can be at most 2^{n^2} sequences $a_1 a_2 \ldots a_n$. A detailed counting argument will show that the number of length sets of depth $n \geq 2^{O(n^2)}$, and this gives us our result.

We have shown that there are at most k^{2k} acceptors with at most k states and that the number of length sets of depth n is $2^{O(n^2)}$. Thus, if we can find a k such that $k^{2k} \leq 2^{O(n^2)}$, then k must be a lower bound on the number of states required. It can be shown that $n^2/\log n$ is such a value for k; thus we obtain our lower bound,

$$\max_{\ell \in L(n)} S(\ell) \geq O\left(\frac{n^2}{\log n}\right).$$

Conclusion

For any length set ℓ of maximum depth n, we have succeeded in obtaining the fairly tight bound on the maximum over $\ell \in L(n)$ of $S(\ell)$, the minimum number of states in any finite-state acceptor that accepts the worst set of code words with maximum length n:

$$O\left(\frac{n^2}{\log n}\right) \leq \max_{\ell \in L(n)} S(\ell) \leq O(n^2).$$

The block-form procedure gives a method of code construction in which we are guaranteed that the number of states in an acceptor will grow at most as fast as the square of the length of the longest code word. It can also be shown that there actually are block-form codes for which the number of states in the acceptor does grow as fast as this; thus the $O(n^2)$ upper bound cannot be improved by using block-form codes. In the computation of the lower bound, we determined that the number of length sets of depth n is $2^{O(n^2)}$. Our count of the number of finite-state machines with at most k states, however, gave us only the gross upper bound of k^{2k}; a more careful count might improve our lower bound result of $O(n^2/\log n)$.

Another approach to problems of this kind is to find a length set that is "close" to the given length set and for which it is easy to build an acceptor. Such approximation questions are considered by Elias.[5]

References

1. L. G. Kraft, "A Device for Quantizing, Grouping and Coding Amplitude Modulated Pulses," S. M. Thesis, Department of Electrical Engineering, M. I. T., 1949.

2. D. A. Huffman, "A Method for the Construction of Minimum-Redundancy Codes," Proc. IRE 40, 1098-1101 (1952).

3. N. Abramson, Information Theory and Coding (McGraw-Hill Publishing Co., New York, 1963).

4. R. G. Gallager, Information Theory and Reliable Communication (John Wiley and Sons, Inc., New York, 1968).

5. P. Elias, "Minimum Times and Memories Needed to Compute the Values of a Function" (to appear in the Journal of Computer and System Sciences).