Universal Coding of Integers and Unbounded Search Trees

Rudolf Ahlswede, Te Sun Han, *Fellow, IEEE*, and Kingo Kobayashi, *Senior Member, IEEE*

***Abstract*—In this paper we study universal coding problems for the integers, in particular, establish rather tight lower and upper bounds for the Elias omega code and other codes. In these bounds, the so-called log-star function plays a central role. Furthermore, we invesigate unbounded search trees induced by these codes, including the Bentley–Yao search tree. We will reveal beautiful recursion structures latent in these search trees as well as in these codes. Finally, we introduce the modified log-star function to reveal the existance of better prefix codes than Elias omega code and other known codes.**

***Index Terms*—Universal coding, search trees, log-star function, Elias code, Stout code, Bentley–Yao tree.**

I. Elias Omega Code and Related Codes for the Positive Integers

LET us introduce some notation to express binary sequences. We denote the unary representation of positive integer j by $(j)_1$. For example, $(1)_1 = 0, (2)_1 = 10, \cdots, (5)_1 = 11110$, etc. The standard binary expression of positive integer $j \in \mathcal{N}^+ = \{1, 2, \cdots\}$ is denoted by $(j)_2$, the most significant bit (MSB) of which is 1. For example, $(13)_2 = 1101$. The binary expression of integer j to the base 2^k is denoted by $(j)_{2,k}$. Thus $(7)_{2,2} = 01\,11, (43)_{2,4} = 0010\,1011$. Furthermore, by $[j]_2$ we mean the standard binary expression of j in which the MSB is removed. For example, $[13]_2 = 101$.

Next, we express the floor function of $\log_2$ by

$$\lambda_2(j) = \lfloor \log_2 j \rfloor. \tag{1.1}$$

Finally, we introduce the integer-valued function $\Lambda_2(j)$ on the set $\mathcal{N}^+$ of positive integers as the one specifying the length of the standard binary expression of positive integer j. That is

$$\Lambda_2(j) = \lfloor \log_2 j \rfloor + 1. \tag{1.2}$$

Let λ_2^k and Λ_2^k denote the k-fold compositions of functions λ_2 and Λ_2, respectively. We will omit the suffix 2 if there is no fear of confusion from the context.

Manuscript received February 1, 1995; revised May 23, 1996.

R. Ahlswede is with Fakultät für Mathematik, Universität Bielefeld, POB 100131, 33501 Bielefeld, Germany.

T. S. Han is with the Graduate School of Information Systems, University of Electro-Communications, Chofugaoka 1-5-1, Chofu, Tokyo, 182, Japan.

K. Kobayashi is with the Department of Computer Science and Information Mathematics, University of Electro-Communications, Chofugaoka 1-5-1, Chofu, Tokyo, 182, Japan.

Publisher Item Identifier S 0018-9448(97)00626-3.

Elias [1] has introduced a universal prefix code $\omega : \mathcal{N}^+ \to \{0,1\}^*$, called the ω code

$$\omega(j) = \begin{cases} 0, & \text{if } j = 1 \\ (\lambda^{k-1}(j))_2 \cdots (\lambda^2(j))_2 (\lambda(j))_2 (j)_2 0, & \text{if } j \geq 2 \end{cases} \tag{1.3}$$

where $k = k(j)$ is the positive integer satisfying $\lambda^k(j) = 1$ (which uniquely exists for any $j \geq 2$). Then the codeword length of this prefix code ω is given by

$$|\omega(j)| = \sum_{i \geq 1 : \lambda^i(j) \geq 0} (\lambda^i(j) + 1) \quad (j = 1, 2, \cdots). \tag{1.4}$$

The codeword $\omega(j)$ is recursively decoded from left to right as follows. If the top bit is 0 then we have 1 as the decoded integer. Otherwise, we first look at the leftmost two bits. Let it represent an integer k_1 in binary expression, then we next look at $k_1 + 1$ bits to the right. Let it represent an integer k_2, then we look at $k_2 + 1$ bits to the right, and so on. This process is repeated until we have bits with 0 at the top, and then the last bits (with 1 at the top) represents the integer j to be decoded.

Furthermore, we may slightly modify the ω code to get another prefix ω' code

$$\omega'(j) = \begin{cases} (j-1)_{2,2}, & \text{if } 1 \leq j \leq 3 \\ 1(\Lambda^{k-1}(j))_2 \cdots (\Lambda(j))_2 (j)_2 0 & \text{if } j \geq 4, \end{cases} \tag{1.5}$$

where $k = k(j)$ is the positive integer satisfying $\Lambda^k(j) = 3$ (which uniquely exists for any $j \geq 4$). The codeword length of this prefix code ω' is given by

$$|\omega'(j)| = 2 + \sum_{i \geq 1 : \Lambda^i(j) \geq 3} \Lambda^i(j). \tag{1.6}$$

The codeword $\omega'(j)$ is decoded in a similar (but a bit different) manner to decoding the above $\omega(j)$. If the leftmost two bits are not equal to 11, then $\omega'(j)$ is decoded as the integer $j = 1, 2$, or 3 according to the value $(j-1)_{2,2}$ of those two bits. Otherwise, delete the top bit 1 and look at the three bits to the right. If it represents an integer k_1, then we look at k_1 bits to the right. If it represents an integer k_2, then we look at k_2 bits to the right, and so on. This process is stopped when we have bits with 0 at the top. Then the last bits (with 1 at the top) in this process represent the integer j to be decoded.

The Elias ω' code thus modified can, of course, be seen as specifying a corresponding search tree for the set of positive integers (unbounded search). It will turn out soon in what follows that it has the same structure as the so-called ultimate search tree devised by Bentley and Yao [2].

TABLE I

j	$\omega(j)$	$\omega'(j)$
1	0	00
2	10 0	01
3	11 0	10
4	10 100 0	1 100 0
5	10 101 0	1 101 0
6	10 110 0	1 110 0
7	10 111 0	1 111 0
8	11 1000 0	1 100 1000 0
9	11 1001 0	1 100 1001 0
10	11 1010 0	1 100 1010 0
11	11 1011 0	1 100 1011 0
12	11 1100 0	1 100 1100 0
13	11 1101 0	1 100 1101 0
14	11 1110 0	1 100 1110 0
15	11 1111 0	1 100 1111 0
16	10 100 10000 0	1 101 10000 0
17	10 100 10001 0	1 101 10001 0
⋮	⋮	⋮

TABLE II

j	$S_0(j)$	$S_1(j)$	$S_2(j)$
0	0	0 0	00 0
1	1 0	1 0	01 0
2	1 10 0	0 10 0	10 0
3	1 11 0	0 11 0	11 0
4	1 10 100 0	1 100 0	00 100 0
5	1 10 101 0	1 101 0	00 101 0
6	1 10 110 0	1 110 0	00 110 0
7	1 10 111 0	1 111 0	00 111 0
8	1 11 100 0	0 10 1000 0	01 1000 0
9	1 11 1001 0	0 10 1001 0	01 1001 0
10	1 11 1010 0	0 10 1010 0	01 1010 0
11	1 11 1011 0	0 10 1011 0	01 1011 0
12	1 11 1100 0	0 10 1100 0	01 1100 0
13	1 11 1101 0	0 10 1101 0	01 1101 0
14	1 11 1110 0	0 10 1110 0	01 1110 0
15	1 11 1111 0	0 10 1111 0	01 1111 0
16	1 10 100 10000 0	0 11 10000 0	10 10000 0
17	1 10 100 10001 0	0 11 10001 0	10 10001 0
⋮	⋮	⋮	⋮

For small values of integers, Table I gives the codewords for two Elias codes.

A main difference between the Elias code $\omega(j)$ and the modified Elias code $\omega'(j)$ is that the different functions in (1.1) and (1.2) are used in the definitions of each codeword, respectively. The value $\lambda(j)$ is always smaller by one than $\Lambda(j)$, so at each step of the above logarithm-like operations, the corresponding block of the Elias codeword for ω is smaller than that for ω', but the number k of the logarithm-like operations in (1.3) is in general larger than that in (1.5).

Remark 1: Even and Rodeh [3] have given another prefix code ω_{ER} similar to ω'. Their code ω_{ER} is defined by

$$\omega_{\mathrm{ER}}(j) = \begin{cases} (j)_{2,3}0, & \text{if } 1 \le j \le 7 \\ (\Lambda^{k-1}(j))_2 \cdots (\Lambda(j))_2 (j)_2 0, & \text{if } j \ge 8 \end{cases} \tag{1.7}$$

where $k = k(j)$ is the unique positive integer satisfying $\Lambda^k(j) = 3$. The code tree of this code, however, is not complete, that is, there are leaves which are not used for codewords. Moreover, the codewords have a redundant bit. After removing the redundant bit, we have the code ω'_{ER} defined by

$$\omega'_{\mathrm{ER}}(j) = \begin{cases} 0(j)_{2,3}, & \text{if } 1 \le j \le 7 \\ (\Lambda^{k-1}(j))_2 \cdots (\Lambda(j))_2 0[j]_2, & \text{if } j \ge 8. \end{cases} \tag{1.8}$$

The codeword length of the code $\omega'_{\mathrm{ER}}(j)$ is shorter by two bits than that of the code $\omega'(j)$ if $j \ge 8$ at the expense of longer codewords for $j \le 7$.

II. Stout Code for the Nonnegative Integers

In order to define the Stout code, for any integer $d \ge 0$ we introduce the following function:

$$\lambda_{[d]}(x) = \lfloor \log_2 x \rfloor - d \quad (x > 0) \tag{2.1}$$

and let $\lambda^k_{[d]}(x)$ be the k-fold composition of the function $\lambda_{[d]}(x)$. The codeword of the Stout prefix code $S_d(d = 0, 1, 2, \cdots)$ is given by

$$S_d(j) = \begin{cases} (j)_{2,d}0, & \text{if } 0 \le j < 2^d \\ \left(\lambda^k_{[d]}(j)\right)_{2,d}\left(\lambda^{k-1}_{[d]}(j)\right)_2 \cdots & \\ \quad \cdots \left(\lambda^2_{[d]}(j)\right)_2 (\lambda_{[d]}(j))_2 (j)_2 0, & \text{if } j \ge 2^d \end{cases} \tag{2.2}$$

for $j \in \mathcal{N} = \{0, 1, \cdots\}$, where k is the unique positive integer satisfying $0 \le \lambda^k_{[d]}(j) < 2^d$.

Remark 2: Stout [4] has defined the code S_d only for $d \ge 2$. However, we can consider such a code also for $d = 0, 1$. In particular, the case $d = 0$, where $(j)_{2,d}$ in (2.2) is interpreted to be void, is intimately related to the Elias omega code ω as will be seen below.

From definition (2.2), the length of the codeword for j is expressed by

$$|S_d(j)| = d + 1 + \sum_{i \ge 1;\, 0 \le \lambda^i_{[d]}(j)} \left(\lambda^i_{[d]}(j) + d + 1\right). \tag{2.3}$$

We should mention here some relation between the length of Elias ω code and that of S_0 code

$$|S_0(j)| = |\omega(j)| + 1 \quad (j = 1, 2, \cdots). \tag{2.4}$$

This is because $S_0(j) = 1\omega(j)$ for $j \ge 1$. Table II shows codewords of codes S_0, S_1, and S_2 for small numbers j.

In general it is expected that, as d increases, the length of $S_d(j)$ becomes smaller for larger values of j and larger for smaller values of j.

III. Unbounded Binary Search Trees

In most database systems, each unit of records is stored with a key for information retrieval. Each unit can be read, updated, and deleted by using its key. Usually, a specific order, for example, alphabetic order, is assigned to the set of keys. Assume that a user wants to have access to a record unit with a certain key. Comparing his key with the keys of certain units determines whether his key is smaller than those keys, or not. According to these binary answers, he continues to compare his key with other keys. The problem is what locations of keys and in what order he should refer to in order to achieve an

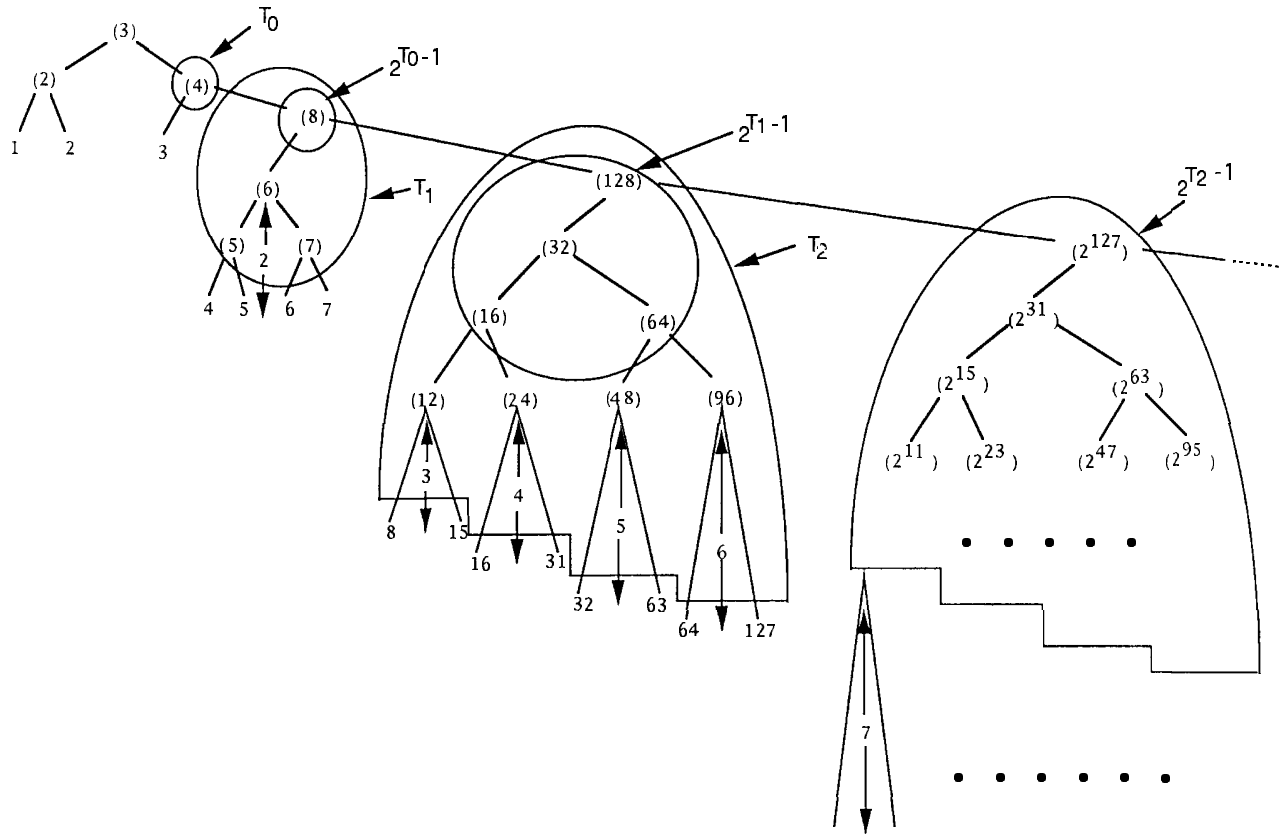

Fig. 1. Bentley–Yao search tree. (k) means that if $j < k$ then go to the left, otherwise to the right.

efficient access to the desired unit. This is one of the most classical problems in computer science.

When the number of units is infinite, the problem is called unbounded search. Bentley and Yao [2] have introduced the so-called ultimate search tree expressed by the tree code $\hat{\omega}'$

$$\hat{\omega}'(j) = \begin{cases} (j-1)_{2,2}, & \text{if } 1 \le j \le 3 \\ (k+2)_1[\Lambda^{k-1}(j)]_2 \cdots [\Lambda(j)]_2[j]_2, & \text{if } j \ge 4 \end{cases} \tag{3.1}$$

where $k = k(j)$ is the unique positive integer satisfying $\Lambda^k(j) = 3$. The unary expression at the head of $\hat{\omega}'(j)$ for $j \ge 4$ describes the number of the subsequent blocks of the form $[\cdot]_2$. Fig. 1 depicts the Bentley–Yao unbounded search tree, where (k) at intermediate nodes specifies the comparison of the target key with the kth key. If the target key is smaller than k, then go on the left branch (labeled "0"), otherwise, go on the right branch (labeld "1"). For example, if your key is the 23rd key, you can reach the desired unit after comparing your key with the third, fourth, eighth, 128th, 32th, 16th, 24th, 20th, 22nd, and 23rd keys, which corresponds to travelling on the path 1110 01 0111 starting with the root (cf. Fig. 1).

It is easy to see that the 1's and the 0 in the unary part $(k+2)_1$ of the definition of $\hat{\omega}'$ can be interleaved into the subsequent blocks $[\cdot]_2$ to obtain the form of definition (1.5) for ω', that is, $1(\Lambda^{k-1}(j))_2 \cdots (\Lambda(j))_2(j)_2 0$. Thus Bentley–Yao search tree is essentially identical to the modified Elias omega code ω'.

A similar idea generates a search tree $\hat{\omega}$ induced from the original Elias omega code ω. Fig. 2 shows that tree. It should be noticed that there exists a more beautiful recursive structure for $\hat{\omega}$ than that for $\hat{\omega}'$. This search tree $\hat{\omega}$ is expressed as

$$\hat{\omega}(j) = \begin{cases} 0, & \text{if } j = 1 \\ (k+1)_1[\lambda^{k-1}(j)]_2 \cdots [\lambda(j)]_2[j]_2, & \text{if } j \ge 2. \end{cases} \tag{3.2}$$

It is also easy to see that the 1's and the 0 in the unary part $(k+1)_1$ in the definition of $\hat{\omega}$ can be interleaved into the subsequent blocks $[\cdot]_2$ to obtain the original form of definition (1.3) for ω, that is, $(\lambda^{k-1}(j))_2 \cdots (\lambda(j))_2(j)_2 0$.

We can also derive search trees from Stout codes S_d with the same kind of prescription

$$\hat{S}_d(j) = \begin{cases} 0(j)_{2,d}, & \text{if } 0 \le j < 2^d \\ (k+1)_1\left(\lambda^k_{[d]}(j)\right)_{2,d}\left[\lambda^{k-1}_{[d]}(j)\right]_2 \cdots \\ \quad \cdots [\lambda_{[d]}(j)]_2[j]_2, & \text{if } j \ge 2^d. \end{cases} \tag{3.3}$$

Fig. 3 depicts the recursive structure of these unbounded search trees $(d = 1)$. Each subtree T_i is recursively reproduced as the next subtree $T_{i+1} \equiv 2^{T_i+d}$ in the search tree $\hat{S}_d$ corresponding to the code S_d.

Such a recursive structure can be more explicitly described as follows. Let $\mathcal{T}$ be the set of binary search trees, that is, the set of binary trees with *nodes* (i.e., root and intermediate nodes) labeled each as (k) and with *leaves* (terminal nodes) labeled each by distinct integers j. As was stated, (k) means the comparison "$j < k$," where j is the number to be searched.

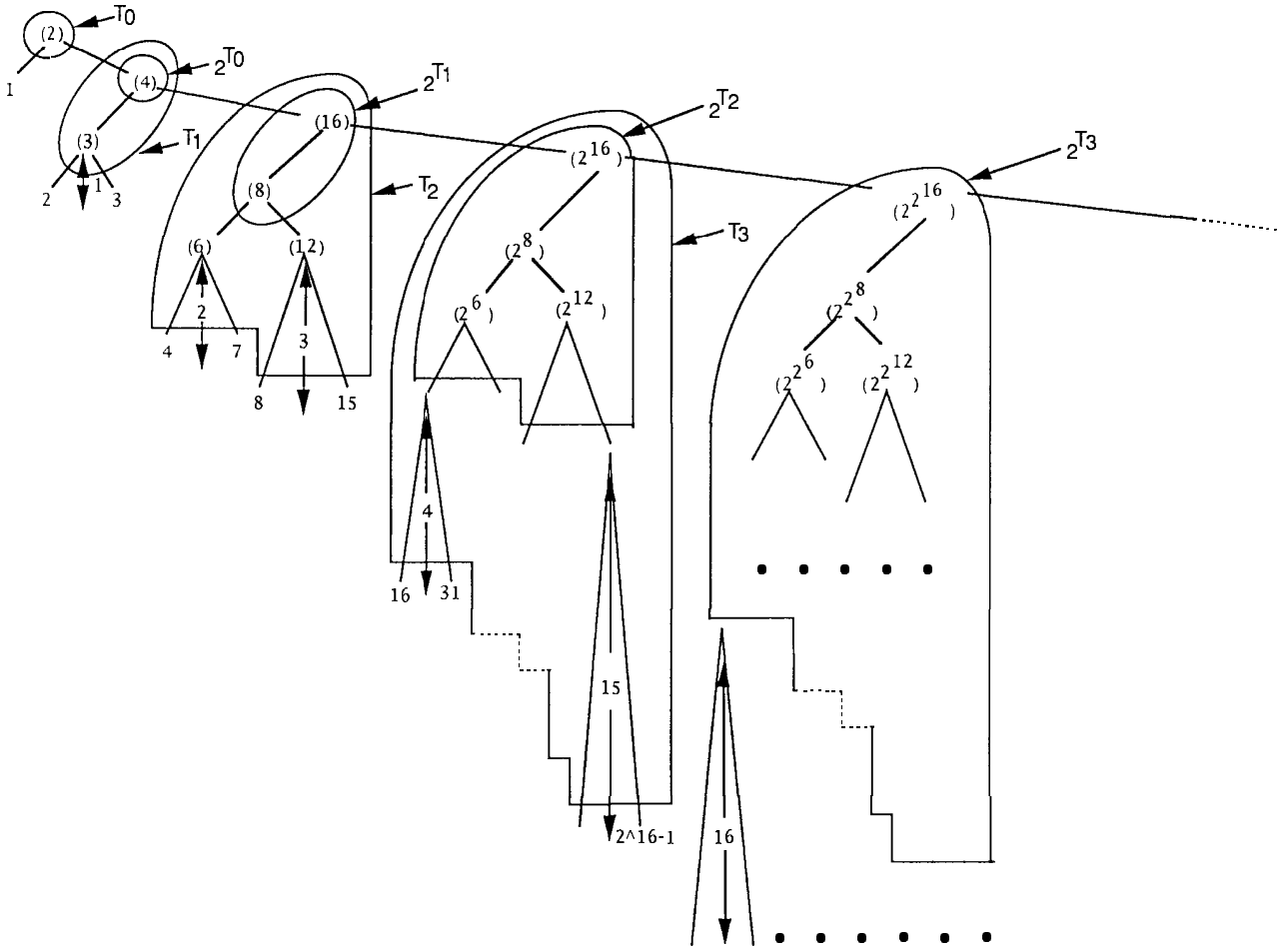

Fig. 2. Search tree induced by Elias ω code. (k) means that if $j < k$ then go to the left, otherwise to the right.

A binary search tree $T \equiv T_0T_1T_2\cdots$ is said to be the *unary concatenation* of binary search trees $T_0, T_1, T_2, \cdots$ if the root of each T_i is connected to the root of the next T_{i+1} by the right branch (labeled 1) for all $i = 0, 1, \cdots$. It is obvious that the above search trees $\hat{\omega}'$, $\hat{\omega}$, and $\hat{S}_d$ are all of concatenated forms in this sense. Moreover, it will be seen in these search trees that the subtree T_{i+1} is recursively constructed from the previously defined subtree T_i.

In order to make more specific what we mean by the recursive construction, let us focus our discussion, for example, on the search tree $\hat{\omega}$ induced by Elias ω code.

First, let T_0 be the tree having two nodes, that is, the root labeled (2) and the left leaf labeled by 1 (connected to the root with the left branch). In order to construct T_{i+1} from T_i, let us next define the *exponentiation operation* $\exp_2 : \mathcal{T} \to \mathcal{T}$ as follows:

1) Given a search tree T_i, all the labels (k) of the nodes of T_i are replaced by the new labels (2^k).
2) The left leaf of T_i labeled as $j = k - 1$, the parent of which has new label (2^k), is replaced by a complete binary subtree with depth $k-1$ whose 2^{k-1} leaves have labels $2^{k-1}, 2^{k-1}+1, \cdots, 2^k - 1$ in this order from left to right, and the nodes of this subtree are given labels indicating the due comparisons which is to be specified below in 4).
3) The right leaf of T_i labeled as $j = k$, the parent of which has new label (2^k), is replaced by a complete binary subtree with depth k whose 2^k leaves have labels $2^k, 2^k+1, \cdots, 2^{k+1} - 1$ in this order from left to right, and the nodes of this subtree are given labels indicating the due comparisons which is to be specified below in 4).
4) The roots of the complete subtrees appearing in cases 2) and 3) are given labels $(2^{k-1} + 2^{k-2})$ and $(2^k + 2^{k-1})$, respectively.

Suppose now that a node a at level lower by $m \geq 0$ from the root of the subtree was given label (h).

Then, in case 2), the left child of the node a (at level lower by $m+1$ from the root of the subtree) is given label $(h - 2^{k-m-3})$, while the right child of the node a (at level lower by $m+1$ from the root of the subtree) is given label $(h + 2^{k-m-3})$.

On the other hand, in case 3), the left child of the node a (at level lower by $m+1$ from the root of the subtree) is given label $(h - 2^{k-m-2})$, while the right child of the node a (at level lower by $m+1$ from the root of the subtree) is given label $(h + 2^{k-m-2})$.

It is evident that, under this recursive labeling, the parents of all the leaves in these subtrees have labels (*odd*) and all the remaining nodes have labels (*even*).

5) Denote the resulting tree by $T_{i+1} \equiv \exp_2 T_i$.

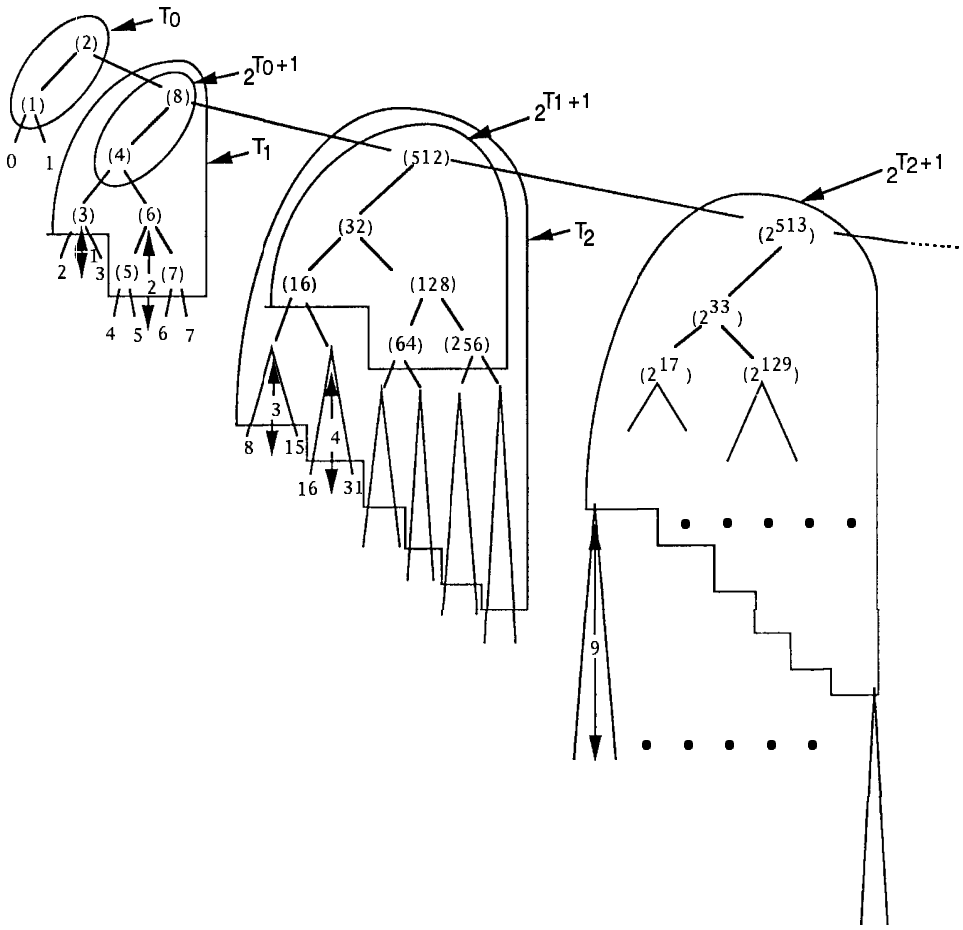

Fig. 3. Search tree induced by Stout code $S_1(d=1)$. (k) means that if $j < k$ then go to the left, otherwise go to the right.

In this manner we have a sequence $T_0, T_1, T_2, \cdots$ of trees, and the concatenated search tree $T = T_0T_1T_2, \cdots$ coincides with the search tree induced by Elias ω code (cf. Fig. 2).

For the Stout codes S_d, we can define similar recursions with the exponentiation operation $\exp_2$ replaced by a similar operation $\exp_2(\cdot + d)$, which means that the labels (k) of all the nodes in T_i are replaced by the new labels (2^{k+d}), and all the leaves of T_i with labels $j = k$ are replaced by a complete binary tree with depth $k+d$. From this viewpoint, Bentley–Yao search tree may be regarded as a special case with $d = -1$ of the Stout search tree.

Remark 3: As one may notice, if our focus is on the asymptotic behavior of the code, we can modify any unbounded search tree to obtain another improved search tree having shorter searching time than that of the original tree. For example, in Fig. 2 we can pull up the comparison "(16)" to the top and perform the due modification of the tree for integers from 1 to 15. Then we have the searching time shorter by 2 for integers greater than 15. This means that it is always possible to improve any prefix code for integers by any finite length for larger values of integers.

IV. Bounds on the Codeword Lengths

In this section we first study the performance of Elias ω code. Codeword length (1.4) of Elias ω code is defined as a function on the set of positive integers

$$c_E(j) \equiv |\omega(j)| = \sum_{i\geq 1:\, \lambda^i(j)\geq 0} (\lambda^i(j)+1) \quad (j = 1, 2, \cdots). \tag{4.1}$$

As we want to treat the problem analytically, we extend the domain of the function $\lambda(j)$ in (1.1) to the set of real numbers $x \geq 1$, that is,

$$\lambda(x) = \lfloor \log_2 x \rfloor \quad (x \geq 1). \tag{4.2}$$

Then (4.1) is also extended to the function on the set of real numbers $x \geq 1$ given by

$$c_E(x) = \sum_{k\geq 1;\lambda^k(x)\geq 0} (\lambda^k(x)+1) \quad (x \geq 1). \tag{4.3}$$

Here, $c_E(x)$ is a monotone-increasing right-continuous step function, because it contains the floor function $\lfloor\cdot\rfloor$.

In order to obtain bounds for $c_E(x)$, we define the *log-star function* $\log_2^*(x)$ for $x \geq 1$ as

$$\log_2^*(x) \equiv \log_2(x) + \log_2\log_2(x) + \cdots + \log_2^{w_2^*(x)}(x) \tag{4.4}$$

where $\log_2^k(x)$ is the k-fold composition of the function $\log_2(x)$, and $w_2^*(x)$ is the largest positive integer w satisfying $\log_2^w(x) \geq 0$. Therefore, $w_2^*(x) = 1, \log_2^*(x) = 0$ for $x = 1$. The function $\log_2^*(x)$ is a monotone-increasing continuous function for $x \geq 1$. Moreover, it is clear that $w_2^*(x)$ is a

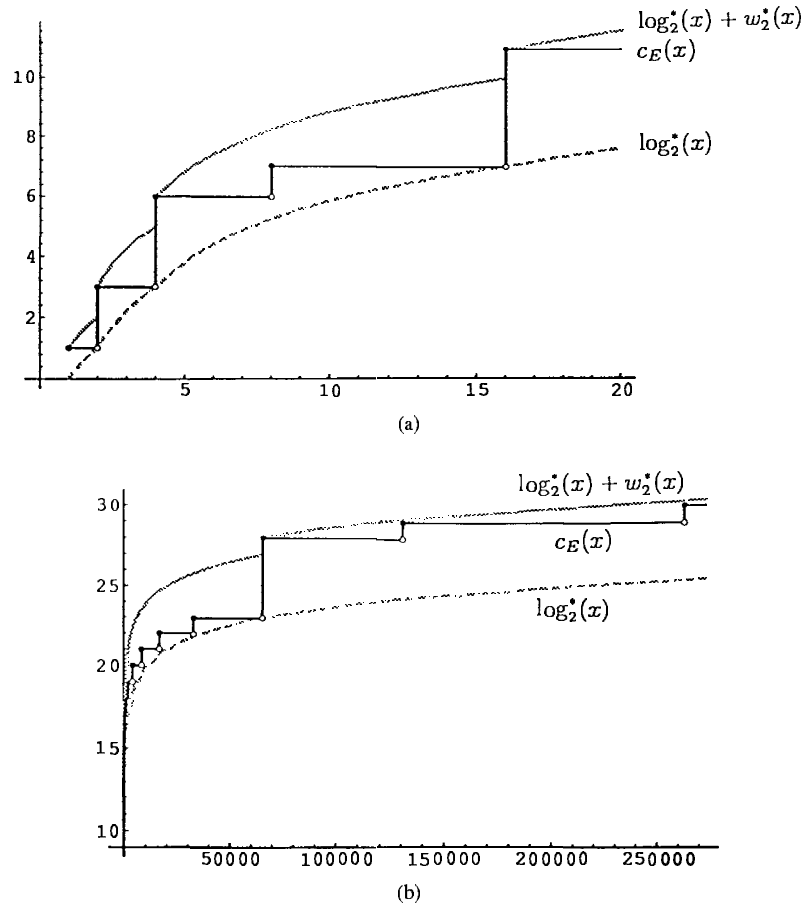

Fig. 4. $c_E(x)$ with its lower (dotted curve) and upper (gray curve) bounds. (a) $1 \le x \le 20$. (b) $1 \le x \le 2^{18}$.

monotone-increasing right-continuous step function. Here, for $\exp_2^m(0) \le x < \exp_2^{m+1}(0)$, we have $w_2^*(x) = m$, where $\exp_2(x) = 2^x$ and $\exp_2^k(x)$ is the k-fold composition of the function $\exp_2(x)$.

Then, we have the following result, which can be used to establish a lower bound for $c_E(x)$.

Lemma 1: For any positive integer k and any real $x \ge \exp_2^k(0)$, we have

$$\log_2^k(x) < \lambda_2^k(x) + 1. \tag{4.5}$$

Proof: Obviously, for $k = 1$ it holds that

$$\log_2(x) < \lfloor \log_2 x \rfloor + 1. \tag{4.6}$$

To show the inequality by induction, assume that it is valid for $k = l$ and $x \ge \exp_2^{l+1}(0)$. Then we have

$$\begin{aligned} \log_2^{l+1}(x) &= \log_2\left(\log_2^l(x)\right) < \log_2\left(\lambda_2^l(x) + 1\right) \quad (4.7) \\ &\le \left\lceil \log_2\left(\lambda_2^l(x) + 1\right) \right\rceil \\ &= \left\lfloor \log_2 \lambda_2^l(x) \right\rfloor + 1 = \lambda_2^{l+1}(x) + 1 \end{aligned}$$

where (4.7) is due to the induction assumption. □

From Lemma 1 and the obvious inequality $\lambda_2^k(x) \le \log_2^k(x)$ for $x \ge \exp_2^k(0)$, we have

Theorem 1: For any real $x \ge 1$

$$\log_2^*(x) < c_E(x) \le \log_2^*(x) + w_2^*(x). \tag{4.8}$$

The graph of $c_E(x)$ together with these lower and upper bounds is shown in Fig. 4(a) and (b) for $1 \le x \le 20$ and $1 \le x \le 2^{18}$.

Remark 4: Through a simple consideration we can easily check that the upper bound $\log_2^*(x) + w_2^*(x)$ in (4.8) is attained at the points $j_m = \exp_2^m(0)$ $(m = 1, 2, \cdots)$. On the other hand, the lower bound $\log_2^*(x)$ in (4.8) is also attained at the same points j_m $(m = 2, 3, \cdots)$ in the sense that

$$\lim_{x \uparrow j_m} c_E(x) = \log_2^*(j_m). \tag{4.9}$$

Therefore, these two bounds are the best possible as far as we restrict the bounding functions to a class of such smooth functions.

We can make a similar argument also to obtain bounds for the codeword length of the Stout code S_d [4]. For that purpose, we define a function $\log_{[d]}$ by removing the floor operation in $\lambda_{[d]}$ as

$$\log_{[d]}(x) = \log_2(x) - d \quad (x > 0) \tag{4.10}$$

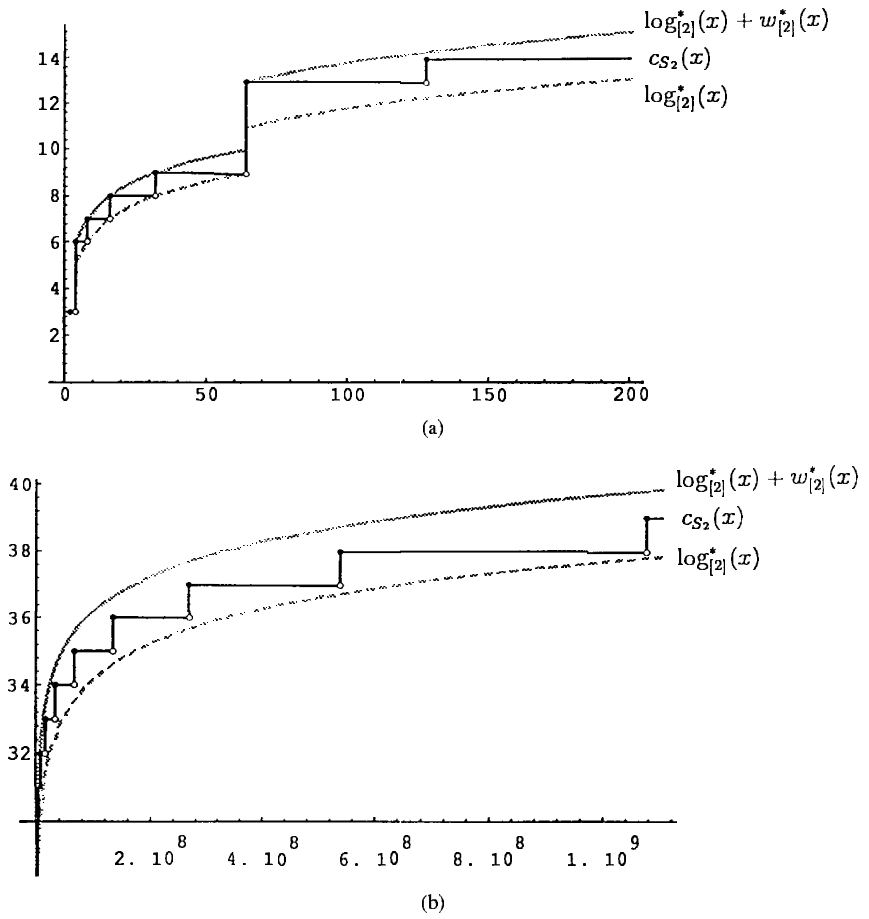

Fig. 5. $c_{S_2}(x)$ with its lower (dotted curve) and upper (gray curve) bounds. (a) $1 \le x \le 200$. (b) $1 \le x \le 2^{30}$.

and let $\log^k_{[d]}(x)$ be the k-fold composition of $\log_{[d]}(x)$. Define for $x \ge 2^d$ the function $\log^*_{[d]}$ as

$$\log^*_{[d]}(x) = \sum_{i=1}^{w^*_{[d]}} \left(\log^i_{[d]}(x) + d \right) + d + 1, \tag{4.11}$$

where $w^*_{[d]} = w^*_{[d]}(x)$ is the integer k satisfying $0 \le \log^k_{[d]}(x) < 2^d$. In particular, in case $d = 0$ we have

$$\log^*_{[d]}(x) = \log^*_2(x) + 1$$

and

$$w^*_{[d]}(x) = w^*_2(x).$$

Let us define the function $\exp_{[d]}(x)$, which is the inverse of $\log_{[d]}(x)$, by $\exp_{[d]}(x) = 2^{x+d}$ for any real x, and let us put

$$j_{m[d]} \equiv \exp^m_{[d]}(0) \quad (m = 2, 3, \cdots).$$

Then, $\log^*_{[d]}(x) (x \ge 2^d)$ is a right-continuous function, i.e., a continuous function except for discontinuities (of jump d) at $j_{m[d]}$ $(m = 2, 3, \cdots)$. It is also evident that $w^*_{[d]}(x)$ is a right-continuous step function with discontinouities of jump 1 at $j_{m[d]}$ $(m = 2, 3, \cdots)$.

The extended length function $c_{S_d}(x) \equiv |S_d(x)|$ satisfies the following bounds.

Theorem 2: For any real $x \ge 2^d$

$$\log^*_{[d]}(x) < c_{S_d}(x) \le \log^*_{[d]}(x) + w^*_{[d]}(x). \tag{4.12}$$

Proof: Let us show by induction that, for $x \ge \exp^k_{[d]}(0)$

$$0 \le \log^k_{[d]}(x) < \lambda^k_{[d]}(x) + 1 \quad (k = 1, 2, \cdots). \tag{4.13}$$

For $k = 1$, we have

$$\begin{aligned} \log_{[d]}(x) &= \log_2(x) - d < \lfloor \log_2(x) \rfloor + 1 - d \\ &= \lambda_{[d]}(x) + 1 \end{aligned} \tag{4.14}$$

and (4.13) is satisfied for $x \ge 2^d$. Next, we assume that (4.13) holds for $k = l$. Then, for $x \ge \exp^{l+1}_{[d]}(0)$

$$\begin{aligned} 0 \le \log^{l+1}_{[d]}(x) &= \log_2 \left(\log^l_{[d]}(x) \right) - d \\ &< \log_2 \left(\lambda^l_{[d]}(x) + 1 \right) - d \\ &\le \left\lceil \log_2 \left(\lambda^l_{[d]}(x) + 1 \right) \right\rceil - d \\ &= \left\lfloor \log_2 \left(\lambda^l_{[d]}(x) \right) \right\rfloor + 1 - d \\ &= \lambda^{l+1}_{[d]}(x) + 1 \end{aligned} \tag{4.15}$$

thus establishing (4.13) for $k = l + 1$. On the other hand, it trivially holds that, for $x \ge \exp^k_{[d]}(0)$

$$0 \le \lambda^k_{[d]}(x) \le \log^k_{[d]}(x). \qquad \square$$

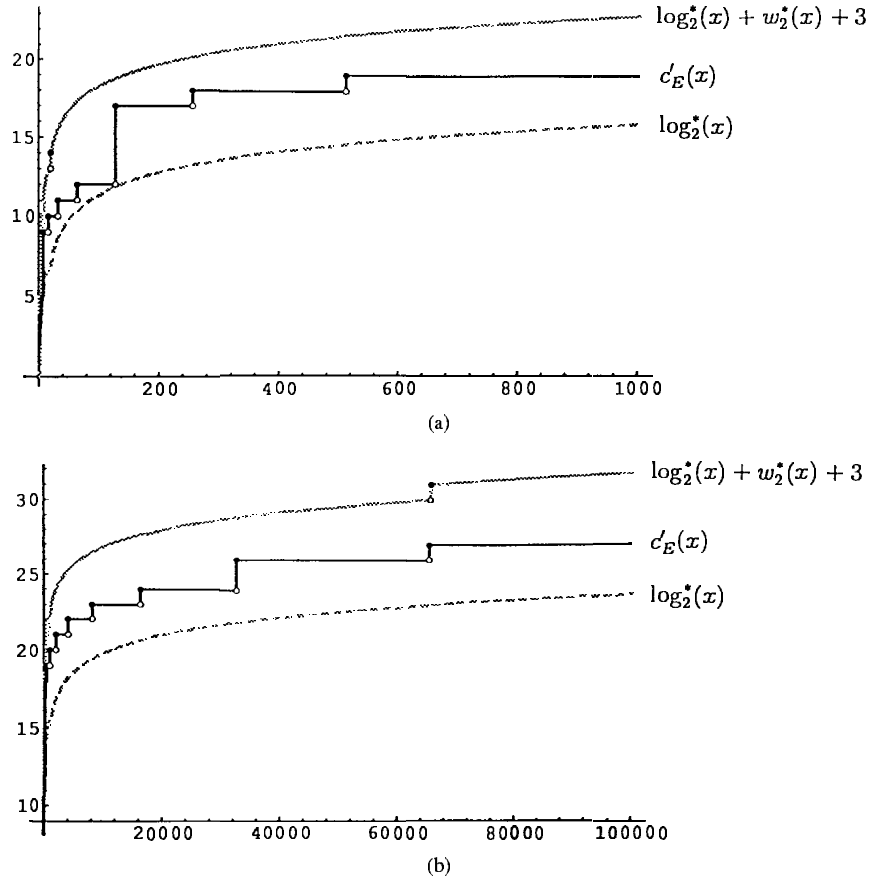

Fig. 6. $c'_E(x)$ with its lower (dotted curve) and upper (gray curve) bounds. (a) $1 \leq x \leq 1000$. (b) $1 \leq x \leq 100\,000$.

Remark 5: As in the Remark 4, we can show the tightness of the bounds in Theorem 2 The upper bound in (4.12) is attained at the points $x = j_{m[d]} \equiv \exp^m_{[d]}(0)$ $(m = 2, 3, \cdots)$. On the other hand, the lower bound in (4.12) is also attained at the same points in the sense that

$$\lim_{x \uparrow j_{m[d]}} c_{S_d}(x) = \lim_{x \uparrow j_{m[d]}} \log^*_{[d]}(x) \quad (m = 1, 2, \cdots). \tag{4.16}$$

For example, the graph of $c_{S_2}(x)$ together with these bounds is shown in Fig. 5(a) and (b) for $1 \leq x \leq 200$ and for $1 \leq x \leq 2^{30}$, respectively.

Next, in order to establish bounds on the length function of the modified Elias code ω', we extend the length function $|\omega'(j)|$ in (1.6) on $\mathcal{N}^+$ to

$$c'_E(x) = |\omega'(x)| = 2 + \sum_{i \geq 1 : \Lambda^i(x) \geq 3} \Lambda^i(x) \quad (\text{real } x \geq 1) \tag{4.17}$$

by extending the function $\Lambda(j) = \lfloor \log_2 j \rfloor + 1$ to the real domain $\{x \geq 1\}$. Then we have

Theorem 3: For any real $x \geq 4$, except for $7 < x < 8$, we have the bounds on the codeword length function (4.17) of the modified Elias code ω'

$$\log_2^*(x) \leq c'_E(x) \leq \log_2^*(x) + w_2^*(x) + 3. \tag{4.18}$$

We give the proof in the Appendix, because it is based on rather subtle arguments and lengthy.

Fig. 6(a) and (b) depicts the graphs of the codeword length function $c'_E(x)$ of the modified Elias code with its lower bound $l_L(x) \equiv \log_2^*(x)$ and its upper bound $l_U(x) \equiv \log_2^*(x) + w_2^*(x) + 3$. The difference of the upper and lower bound is within six bits for $x \leq 100\,000$. Thus these bounds may be considered as rather satisfactory ones to the step function $c'_E(x)$.

Remark 6: The function $w_2^*(x)$ which appeared in the right-hand sides in (4.8) and (4.18) is a quite slowly increasing function in comparison with $\log_2^*(x)$. For example, $w_2^*(100) = 4$ and $w_2^*(1\,000\,000) = 5$.

Remark 7: Due to the inequality

$$\lambda^k(x) + 1 \leq \Lambda^k(x) \quad (x \geq 1, k = 1, 2, \cdots)$$

which can be shown by induction with respect to k, we have a relation between two codeword length functions of the codes ω and ω'

$$c_E(x) \leq c'_E(x) + 1 \quad (x \geq 1). \tag{4.19}$$

It can also be checked that (4.19) holds with equality for $x = j_m$ $(m = 2, 3, \cdots)$. It should be noticed here that the leftmost block of ω' after removing the top bit 1 consists of three bits, and the leftmost block of ω consists of two bits. So the lengths $c_E(x)$ and $c'_E(x)$ of two codes will interchange when x approaches infinity as is shown in Fig. 7.

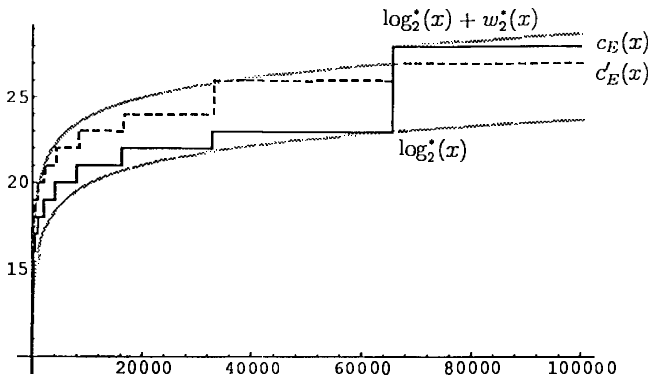

Fig. 7. $c_E(x)$ (solid curve) with its lower and upper (gray curve) bounds and $c'_E(x)$ (dotted curve) for $1 \le x \le 100\,000$.

V. Properties of the Log-Star Function

A. Prefix Code on $\mathcal{N}^+$ Induced by the Log-Star Function

In the previous sections we have shown an intimate relation between the codeword length functions $c_E(x)$ and $c'_E(x)$ of Elias omega codes and the log-star function $\log_2^*(x)$. In this section, we will review some properties of the log-star function itself. First, Leung-Yan-Cheng and Cover [5] have shown that the sum of $\exp_2(-\log_2^*(j))$ is finite:

Lemma 2:

$$\sum_{j=1}^{\infty} 2^{-\log_2^*(j)} < +\infty. \tag{5.1}$$

Proof: This inequality follows immediately from (6.20) with $r = 2, \alpha = 0$ in Corollary 2 below. □

From Lemma 2 we have the normalizing constant $d^* > 0$ such that

$$\sum_{j=1}^{\infty} 2^{-\left(d^* + \log_2^*(j)\right)} = 1.$$

This constant can be evaluated by the method of Rissanen [6] as $d^* = 1.5185\ldots$. Thus we can define the probability distribution p_0 on the set $\mathcal{N}^+$ of positive integers by

$$p_0(j) = 2^{-(d^* + \log_2^*(j))} \qquad (j \in \mathcal{N}^+) \tag{5.2}$$

which was called the "uniform distribution" on $\mathcal{N}^+$ by Rissanen [6]. Based on this distribution, we are able to obtain a prefix Shannon code $\mathcal{C}_0$ with the codeword length function c_0 defined by

$$c_0(j) = \lceil d^* + \log_2^*(j) \rceil \qquad (j \in \mathcal{N}^+) \tag{5.3}$$

where $\lceil x \rceil$ is the ceiling function of x, that is, the smallest integer which is greater than or equal to x.

Comparing this length function c_0 with c'_E of the modified Elias omega code, we see that $c'_E(j) < c_0(j)$ for small values of j, but the inequality $c'_E(j) > c_0(j)$ holds for any larger j, as shown in Figs. 8(a) and (b). Therefore, the code $\mathcal{C}_0$ shows a better performance than the modified Elias omega code or the Bentley–Yao's so-called "ultimate" code.

On the other hand, as will be seen by comparing the length function c_0 with c_E of the original Elias ω code,

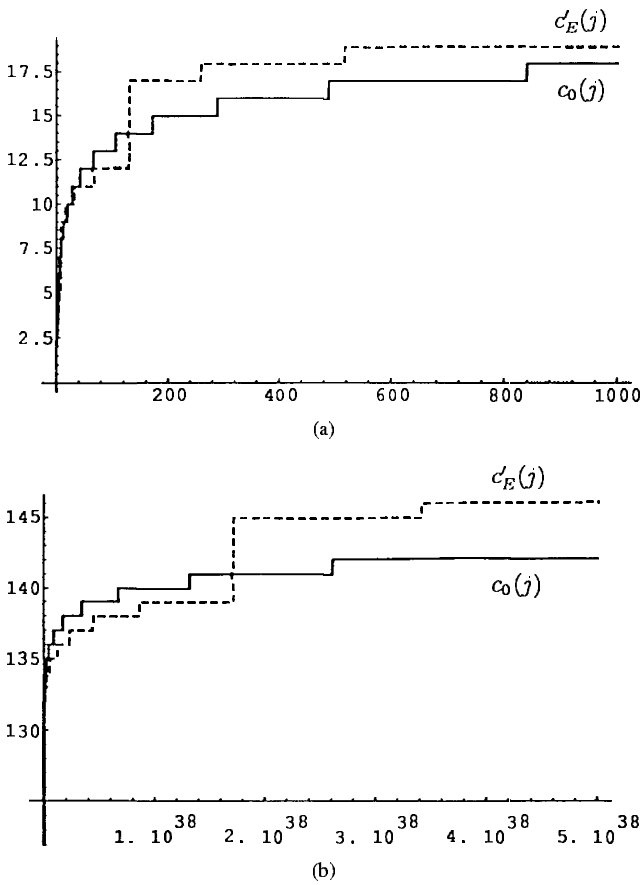

Fig. 8. $c_0(x)$ (solid curve) of Shannon code corresponding to the *uniform* distribution $p_0(j)$ and $c'_E(x)$ (dotted curve). (a) $1 \le x \le 100\,000$. (b) $1 \le x \le 5 \times 10^{38}$.

Remark 4 implies that there are infinitely many j satisfying $c_E(j) < c_0(j)$. Nevertheless, for almost all values of j it holds $c_E(j) > c_0(j)$. In Section VII we will show the existence of an asymptotically better code than code $\mathcal{C}_0$.

Remark 8: There is still no simple and elegant algorithm, such as that for the Elias omega code ω, the modified Elias omega code ω', or the Benty–Yao search tree $\hat{\omega}'$, for constructing the prefix Shannon code with the length function $c_0(j)$ $(j \in \mathcal{N}^+)$ defined in (5.3).

B. Infinitely Often Lower Bound on the Length Function of Prefix Codes

Another property concerning the log-star function was given by Bentley and Yao [2]. Now, let $f(j)(j \in \mathcal{N}^+)$ be the codeword length of an arbitrary prefix code on $\mathcal{N}^+$, then it must satisfy the Kraft inequality

$$\sum_{j=1}^{\infty} 2^{-f(j)} \le 1. \tag{5.4}$$

Theorem 4: Assume that an integer-valued function f satisfies Kraft inequality (5.4). Then, for infinitely many $j \in \mathcal{N}^+$

$$f(j) \ge \log_2^*(j) - \left\{w_2^*(j) + \log_2^*\left(w_2^*(j)\right)\right\}. \tag{5.5}$$

Theorem 8 in the following gives a stronger result. Its proof is simpler than that of Bentley and Yao.

VI. Modified Log-Star Functions

We have shown that the codeword length function c_E of Elias ω code is lower-bounded by the log-star function $\log_2^*(j)$ and is upper-bounded by the log-star function plus $w_2^*(j)$, and also that there exists a prefix code C_0 with the length function $c_0(j) = \lceil d^* + \log_2^*(j) \rceil$ in view of Lemma 2. So we might be tempted to think that c_0 gives an asymptotically *best* prefix code for integers. Is it true?

On the other hand, in contrast to (5.1), the argument of Leung-Yan-Cheng and Cover [5] shows that, for any integer $r \geq 3$

$$\sum_{j=1}^{\infty} r^{-\log_r^*(j)} = +\infty \tag{6.1}$$

where the definition of the generalized log-star function $\log_r^*(j)$ will be given by (6.6). Thus we cannot have any r-ary prefix code with the length function of the form $\lceil d_r^* + \log_r^*(j) \rceil$ for any constant d_r^*. This phenomenon is in contrast to the binary case. Then, what is the essential difference between the binary and r-ary cases ($r \geq 3$)?

This section is motivated by such questions. Incidentally, we will show the existence of an infinite sequence of prefix codes having better performance for larger integers, that is, codes which have shorter codeword lengths than the log-star $\log_2^*(j)$ for large integers.

We begin with some definitions. Let $w_r^*(x)$ $(x \geq 1)$ be the largest positive integer w satisfying $\log_r^w(x) \geq 0$ as in the binary $(r = 2)$ case. Similarly, define $w_r^+(x)$ $(x \geq 1)$ as the largest nonnegative integer w satisfying $\log_r^w(x) > 0$. By convention, $w_r^+(1) = 1$. Thus both functions $w_r^*(x)$ and $w_r^+(x)$ are monotone increasing step functions with discontinuities at $x = \exp_r^i(0)$ $(i = 1, 2, \cdots)$. These functions take the same values for $x \neq \exp_r^i(0)$ $(i = 2, 3, \cdots)$

$$w_r^*(x) = w_r^+(x) \tag{6.2}$$

and at discontinuities $x = \exp_r^i(0)(i = 2, 3, \cdots)$

$$w_r^*(x) = w_r^+(x) + 1 \tag{6.3}$$

where w_r^* is right-continuous, and w_r^+ is left-continuous. Explicity, we can write these functions as follows:

$$w_r^*(x) = i, \quad \text{for } \exp_r^i(0) \leq x < \exp_r^{i+1}(0) \quad (i = 1, 2, \cdots) \tag{6.4}$$

$$w_r^+(x) = i, \quad \text{for } \exp_r^i(0) < x \leq \exp_r^{i+1}(0) \quad (i = 1, 2, \cdots). \tag{6.5}$$

Then, for integer $r \geq 2$, the log-star function $\log_r^*$ to the base r is defined as

$$\log_r^*(x) \equiv \log_r(x) + \log_r \log_r(x) + \cdots + \log_r^{w_r^*(x)}(x) \quad (x \geq 1). \tag{6.6}$$

Let us define, for any integer $r \geq 2$ and any real number α, β, the *modified log-star functions* $\log_{r,\alpha,\beta}^*$ and $\log_{r,\alpha,\beta}^+$ by

$$\log_{r,\alpha,\beta}^*(x) = \log_r^*(x) - \alpha w_r^*(x) + \beta \log_r \left(w_r^*(x)\right) \tag{6.7}$$

$$\log_{r,\alpha,\beta}^+(x) = \log_r^*(x) - \alpha w_r^+(x) + \beta \log_r \left(w_r^+(x)\right). \tag{6.8}$$

It should be noted that we have the log-star function $\log_2^*$ as a special case with $\alpha = 0$, $\beta = 0$, $r = 2$.

The properties of the modified log-star functions are described in the following theorem and its corollaries.

Theorem 5: Set $\alpha_r^* = \log_r(\log_r e)$, then we have

1) For $\beta > 1$

$$\sum_{j=1}^{\infty} r^{-\log_{r,\alpha_r^*,\beta}^+(j)} < +\infty. \tag{6.9}$$

2) For $\beta \leq 1$

$$\sum_{j=1}^{\infty} r^{-\log_{r,\alpha_r^*,\beta}^*(j)} = +\infty. \tag{6.10}$$

Proof: 1) First, we set

$$I_{r,\beta}^+ = \sum_{j=1}^{\infty} r^{-\log_{r,\alpha_r^*,\beta}^+(j)}. \tag{6.11}$$

Then, rewriting $\log_{r,\alpha_r^*,\beta}^+(x)$ with $a^* = r^{-\alpha_r^*}$ as

$$\log_{r,\alpha_r^*,\beta}^+(x) = \left(\log_r x + \log_r a^*\right) + \left(\log_r \log_r x + \log_r a^*\right) + \cdots + \left(\log_r^{w_r^+(x)}(x) + \log_r a^*\right) + \log_r \left(w_r^+(x)\right)^{\beta}$$

we have

$$I_{r,\beta}^+ = \sum_{j=1}^{\infty} \frac{1}{a^* j \cdot a^* \log_r j \cdots a^* \log_r^{w_r^+(j)-1}(j) \cdot \left(w_r^+(x)\right)^{\beta}}.$$

On the other hand, we have an indefinite integral for any integer $m \geq 1$

$$\int \frac{dx}{a^* x \cdot a^* \log_r x \cdots a^* \log_r^{m-1}(x) \cdot m^{\beta}} = \frac{1}{m^{\beta}} \left(\frac{\log_e r}{a^*}\right)^m \log_r^m(x) = \frac{1}{m^{\beta}} \log_r^m(x).$$

Therefore, by setting

$$S_{m,\beta} \equiv \int_{\exp_r^m(0)}^{\exp_r^{m+1}(0)} \frac{dx}{a^* x \cdot a^* \log_r x \cdots a^* \log_r^{w_r^+(x)-1}(x) \cdot \left(w_r^+(x)\right)^{\beta}} \tag{6.12}$$

we obtain

$$S_{m,\beta} = \frac{1}{m^{\beta}} \left[\log_r^m(x)\right]_{\exp_r^m(0)}^{\exp_r^{m+1}(0)} = \frac{1}{m^{\beta}}. \tag{6.13}$$

Noting that $r^{-\log_{r,\alpha_r^*,\beta}^+(x)}$ on the right-hand side of (6.11) is left-continuous for real $x \geq 1$ and continuous and monotone-decreasing in each interval $(\exp_r^i(0), \exp_r^{i+1}(0))$ $(i = 1, 2, \cdots)$, we can establish the inequality

$$I_{r,\beta}^+ \leq r^{\alpha_r^*} + \sum_{m=1}^{\infty} S_{m,\beta} = r^{\alpha_r^*} + \sum_{m=1}^{\infty} \frac{1}{m^{\beta}} \tag{6.14}$$

where we have used the fact

$$r^{-\log_{r,\alpha_r^*,\beta}^*(x)} = r^{\alpha_r^*}, \quad \text{for } x = 1.$$

We notice here that the second term on the right-hand side of (6.14) is nothing but the Zeta function $\zeta(\beta)$. Consequently, by (6.14), if $\beta > 1$ then we have $I^{+}_{r,\beta} < +\infty$.

2) Next, let us set

$$I^{*}_{r,\beta} = \sum_{j=1}^{\infty} r^{-\log^{*}_{r,\alpha^{*}_{r},\beta}(j)} \tag{6.15}$$

where $r^{-\log^{*}_{r,\alpha^{*}_{r},\beta}(x)}$ on the right-hand side is right-continuous for real $x \geq 1$ and continuous, monotone-decreasing in each interval $(\exp^{i}_{r}(0), \exp^{i+1}_{r}(0))$ $(i = 1, 2, \cdots)$. Noting that the value of the integeral is invariant (due to (6.2)) even if we replace $w^{+}_{r}(x)$ by $w^{*}_{r}(x)$, we obtain the inequality similar to (6.14)

$$I^{*}_{r,\beta} \geq \sum_{m=1}^{\infty} S_{m,\beta} = \sum_{m=1}^{\infty} \frac{1}{m^{\beta}}. \tag{6.16}$$

Therefore, if $\beta \leq 1$ then we have $I^{*}_{r,\beta} = +\infty$. □

In view of the inequality

$$w^{+}_{r}(x) \leq w^{*}_{r}(x) \leq w^{+}_{r}(x) + 1$$

it is easy to see that the boundedness of $I^{+}_{r,\beta}$ means that of $I^{*}_{r,\beta}$, and the infiniteness of $I^{*}_{r,\beta}$ means that of $I^{+}_{r,\beta}$. Thus we have the following result as a consequence of the previous theorem. Notice here that the positions of symbols * and $^{+}$ are exchanged.

Corollary 1:

1) For $\beta > 1$

$$\sum_{j=1}^{\infty} r^{-\log^{*}_{r,\alpha^{*}_{r},\beta}(j)} < +\infty. \tag{6.17}$$

2) For $\beta \leq 1$

$$\sum_{j=1}^{\infty} r^{-\log^{+}_{r,\alpha^{*}_{r},\beta}(j)} = +\infty. \tag{6.18}$$

As a special case of the modified log-star functions $\log^{*}_{r,\alpha,\beta}(x)$, we may consider the following functions ($\beta = 0$):

$$\log^{*}_{r,\alpha}(x) \equiv \log^{*}_{r,\alpha,0}(x) = \log^{*}_{r}(x) - \alpha w^{*}_{r}(x). \tag{6.19}$$

Then, an immediate consequence of Theorem 5 and Corollary 1 is the following property due to Levenshtein [7].

Corollary 2:

1) For $\alpha < \alpha^{*}_{r}$

$$\sum_{j=1}^{\infty} r^{-\log^{*}_{r,\alpha}(j)} < +\infty. \tag{6.20}$$

2) For $\alpha \geq \alpha^{*}_{r}$

$$\sum_{j=1}^{\infty} r^{-\log^{*}_{r,\alpha}(j)} = +\infty. \tag{6.21}$$

Proof: It is easy to see that

$$\log^{*}_{r,\alpha^{*}_{r},\beta}(j) \leq \log^{*}_{r,\alpha}(j) \tag{6.22}$$

for $\alpha < \alpha^{*}_{r}, \beta > 1$ and for all sufficiently large j, and also that

$$\log^{*}_{r,\alpha^{*}_{r},\beta}(j) \geq \log^{*}_{r,\alpha}(j) \tag{6.23}$$

for $\alpha \geq \alpha^{*}_{r}$, $0 \leq \beta \leq 1$ and for all sufficiently large j. From (6.22) combined with (6.17) in Corollary 1, (6.20) follows, whereas from (6.23) combined with (6.10) in Theorem 5, (6.21) follows. □

The inequalities in Corollary 2 do not yet attain the best possible bound. To see this, for example, let us define

$$\log^{**}_{r,\alpha}(j) \equiv \log^{*}_{r,\alpha^{*}_{r}}(j) + \log^{*}_{r,\alpha}\left(w^{*}_{r}(j)\right) \tag{6.24}$$

for $j = 1, 2, \cdots$. Then, we have a stronger version of Corollary 2 as follows.

Theorem 6:

1) For $\alpha < \alpha^{*}_{r}$

$$\sum_{j=1}^{\infty} r^{-\log^{**}_{r,\alpha}(j)} < +\infty. \tag{6.25}$$

2) For $\alpha \geq \alpha^{*}_{r}$

$$\sum_{j=1}^{\infty} r^{-\log^{**}_{r,\alpha}(j)} = +\infty. \tag{6.26}$$

Proof: Taking account of Corollary 2, the claim can be proven in the same manner as in the proof of Theorem 5. □

It is possible also to put forth Corollary 2 and Theorem 6 into a much more general form; let us denote by $w^{(k)*}_{r}(x)$ the k-fold composition of $w^{*}_{r}(x)$ and define, for real $x \geq 1$,

$$\log^{(m)*}_{r,\alpha}(x) \equiv \sum_{k=0}^{m-2} \log^{*}_{r,\alpha^{*}_{r}}\left(w^{(k)*}_{r}(x)\right) + \log^{*}_{r,\alpha}\left(w^{(m-1)*}_{r}(x)\right) \tag{6.27}$$

where $w^{(0)*}_{r}(x) = x$ by convention. It is evident that

$$\log^{(1)*}_{r,\alpha}(x) = \log^{*}_{r,\alpha}(x)$$
$$\log^{(2)*}_{r,\alpha}(x) = \log^{**}_{r,\alpha}(x).$$

Then, repeated application of the arguments leading to Corollary 2 and Theorem 6 establishes the following general Theorem 7. It should be noted here that, for $m = 1, 2, \cdots$,

$$\log^{(m)*}_{r,\alpha}(x) - \log^{(m+1)*}_{r,\beta}(x) \to +\infty \quad (x \to +\infty)$$

whenever $\alpha < \alpha^{*}_{r}$. Therefore, the larger m, inequality (6.28) in Theorem 7 is the stronger.

Theorem 7: For any integer $m = 1, 2, \cdots$, the following holds.

1) For $\alpha < \alpha^{*}_{r}$

$$\sum_{j=1}^{\infty} r^{-\log^{(m)*}_{r,\alpha}(j)} < +\infty. \tag{6.28}$$

2) For $\alpha \geq \alpha^{*}_{r}$

$$\sum_{j=1}^{\infty} r^{-\log^{(m)*}_{r,\alpha}(j)} = +\infty. \tag{6.29}$$

Let us now consider several properties of prefix codes on the basis of Theorem 5 and Corollary 1. When $\beta > 1$, we may think of two alternatives, $\log^*_{r,\alpha^*_r,\beta}(j)$ and $\log^+_{r,\alpha^*_r,\beta}(j)$, as length functions satisfying Kraft inequality. In this case there exist normalizing constants $d^*_{r,\beta}, d^+_{r,\beta}$ such that

$$\sum_{j=1}^{\infty} r^{-\left(d^*_{r,\beta}+\log^*_{r,\alpha^*_r,\beta}(j)\right)} = 1 \tag{6.30}$$

$$\sum_{j=1}^{\infty} r^{-\left(d^+_{r,\beta}+\log^+_{r,\alpha^*_r,\beta}(j)\right)} = 1. \tag{6.31}$$

Therefore,

$$p^*_{r,\beta}(j) = r^{-(d^*_{r,\beta}+\log^*_{r,\alpha^*_r,\beta}(j))} \tag{6.32}$$

$$p^+_{r,\beta}(j) = r^{-(d^+_{r,\beta}+\log^+_{r,\alpha^*_r,\beta}(j))} \tag{6.33}$$

specify probability distributions on $\mathcal{N}^+$. The corresponding Shannon codeword length functions are given by

$$c^*_{r,\beta}(j) = \lceil d^*_{r,\beta} + \log^*_{r,\alpha^*_r,\beta}(j) \rceil \tag{6.34}$$

$$c^+_{r,\beta}(j) = \lceil d^+_{r,\beta} + \log^+_{r,\alpha^*_r,\beta}(j) \rceil \tag{6.35}$$

for $j \in \mathcal{N}^+$.

A. Binary Codes

Consider the binary code alphabet case, that is, $r = 2$. Then, $\alpha^*_2 = \log_2(\log_2 e) = 0.5287\ldots$, that is, $\alpha^*_r > 0$, and hence we can choose a positive α in order to achieve a length function (satisfying Kraft's inequality) for prefix codes.

1) Let us consider the probability distribution on $\mathcal{N}^+$

$$\begin{aligned} p^*_{2,\beta}(j) &= 2^{-(d^*_{2,\beta}+\log^*_{2,\alpha^*_2,\beta}(j))} \\ &= 2^{-(d^*_{2,\beta}+\log^*_2(j)-\alpha^*_2 w^*_2(j)+\beta\log_2(w^*_2(j)))} \end{aligned} \tag{6.36}$$

with $\beta > 1$. The distribution $p^*_{2,\beta}$ is not monotone-decreasing, unlike the Rissanen's "uniform" distribution p_0 (cf.(5.2)), because the right-hand side of (6.36) contains $w^*_2(j)$ with a negative sign, although in each interval $(\exp^i_2(0), \exp^{i+1}_2(0))$ $(i = 1, 2, \cdots)$ it is a continuous, monotone-decreasing function. The term $w^*_2(j)$ works to make the decreasing speed of $p^*_{2,\beta}(j)$ slower than that of the distribution $p_0(j)$, and hence $p^*_{2,\beta}$ is more uniform than the distribution p_0.

2) The corresponding Shannon codeword length function $c^*_{2,\beta}$

$$\begin{aligned} c^*_{2,\beta}(j) &= \lceil d^*_{2,\beta} + \log^*_{2,\alpha^*_2,\beta}(j) \rceil \\ &= \lceil d^*_{2,\beta} + \log^*_2(j) - \alpha^*_2 w^*_2(j) + \beta\log_2\left(w^*_2(j)\right) \rceil \end{aligned} \tag{6.37}$$

is asymptotically shorter by $\alpha^*_2 w^*_2(j)$ than $c_0(j)$ given by (5.3). In view of Theorems 1 and 3, it is also shorter by $\alpha^*_2 w^*_2(j)$ than those of Elias omega codes. Thus $c^*_{2,\beta}$ specifies a better code than Elias codes. However, the algorithmic construction of such an asymptotically better code remains open.

3) We have an infinitely often lower bound on codeword lengths of prefix codes as follows.

Theorem 8: Let $f(j)$ $(j \in \mathcal{N}^+)$ be any integer-valued function satisfying Kraft's inequality (5.4). Then, for infinitely many $j \in \mathcal{N}^+$

$$f(j) \geq \log^*_2(j) - \alpha^*_2 w^*_2(j) + \log_2\left(w^*_2(j)\right). \tag{6.38}$$

Proof: If the statement is not valid, then there is a constant d such that for all $j \in \mathcal{N}^+$

$$\begin{aligned} f(j) &< d + \log^*_2(j) - \alpha^*_2 w^*_2(j) + \log_2\left(w^*_2(j)\right) \\ &\equiv d + \log^*_{2,\alpha^*_2,1}(j). \end{aligned}$$

Therefore, we have

$$\sum_{j=1}^{\infty} 2^{-f(j)} > 2^{-d} \sum_{j=1}^{\infty} 2^{-\log^*_{2,\alpha^*_2,1}(j)}.$$

The right-hand side equals $+\infty$ by (6.18) of Corollary 1. This contradicts Kraft's inequality (5.4). □

Remark 9: In Theorem 4, the counterpart of the sum of the second and third terms on the right side of (6.38) is $w^*_2(j) + \log^*_2(w^*_2(j))$ (see (5.5)). In light of $\alpha^*_2 < 1$, the following simple inequality holds:

$$\alpha^*_2 w^*_2(j) - \log_2\left(w^*_2(j)\right) < w^*_2(j) + \log^*_2\left(w^*_2(j)\right) \tag{6.39}$$

which shows that Theorem 8 gives a tighter lower bound. It is easy to see that the right-hand side of (6.38) can be replaced by the following tighter one:

$$\log^{(2)*}_{2,\alpha^*_2}(j) \equiv \log^*_{2,\alpha^*_2}(j) + \log^*_{2,\alpha^*_2}\left(w^*_2(j)\right)$$

or more generally by

$$\log^{(m)*}_{2,\alpha^*_2}(j) \quad (m \geq 2).$$

A similar remark is valid also for Theorem 9 below.

B. r-ary Codes for $r \geq 3$

When $r \geq 3$, $\alpha^*_r = \log_r(\log_r e)$ is negative, and so we have to choose a negative α in order to establish a codeword length function (satisfying Kraft inequality) for prefix codes. In this case, the probability distribution (6.32) on $\mathcal{N}^+$

$$p^*_{r,\beta}(j) = r^{-(d^*_{r,\beta}+\log^*_r(j)-\alpha^*_r w^*_r(j)+\beta\log_r(w^*_r(j)))} \tag{6.40}$$

with any $\beta > 1$ is monotone-decreasing in $j \in \mathcal{N}^+$. The Shannon codeword length function $c^*_{r,\beta}$ corresponding to this distribution $p^*_{r,\beta}$ is given by

$$\begin{aligned} c^*_{r,\beta}(j) &= \lceil d^*_{r,\beta} + \log^*_{r,\alpha^*_r,\beta}(j) \rceil \\ &= \lceil d^*_{r,\beta} + \log^*_r(j) - \alpha^*_r w^*_r(j) + \beta\log_r\left(w^*_r(j)\right) \rceil \end{aligned} \tag{6.41}$$

which is clearly a monotone-increasing function. On the other hand, owing to (6.21), the length function given by

$$c^*_r(j) = \lceil d_r + \log^*_r(j) \rceil$$

which is defined as a naive generalization of the binary case (cf. (5.3)), cannot be a codeword length function for any prefix code at all. As in the binary case, getting β closer to 1 from above, we have an infinite sequence of prefix codes with

shorter lengths $c^*_{r,\beta}(j)$. The algorithmic construction of such codes remains still an open problem.

We conclude this section by giving the r-ary counterpart of Theorem 8 for the binary case.

Theorem 9: Let $r \geq 3$ be any integer, and let $f(j)(j \in \mathcal{N}^+)$ be any integer-valued function satisfying Kraft inequality

$$\sum_{j=1}^{\infty} r^{-f(j)} \leq 1. \tag{6.42}$$

Then, for infinitely many $j \in \mathcal{N}^+$,

$$f(j) \geq \log^*_r(j) - \alpha^*_r w^*_r(j) + \log_r\left(w^*_r(j)\right) \tag{6.43}$$

where it should be noted that $\alpha^*_r < 0$ for $r \geq 3$.

Proof: The proof is quite similar to that of Theorem 8. As was pointed out in Remark 9, the right-hand side of (6.43) can be replaced by $\log^{(m)*}_{r,\alpha^*_r}$ $(m \geq 2)$. □

APPENDIX
PROOF OF THEOREM 3

First let us recall that the codeword length (4.17) of the modified Elias ω' code is given by

$$c'_E(x) = |\omega'(x)| = 2 + \sum_{i \geq 1 : \Lambda^i(x) \geq 3} \Lambda^i(x) \quad (\text{real } x \geq 1) \tag{A1}$$

where $\Lambda(x)$ is the extended function of $\Lambda(j)$ in (1.2)

$$\Lambda(x) = \lfloor \log_2 x \rfloor + 1 \quad (\text{real } x \geq 1) \tag{A2}$$

and Λ^i is the i-fold composition of Λ. It is obvious that c'_E is a monotone-increasing right-continuous step function, because it contains the floor function $\lfloor \cdot \rfloor$.

A. Lower Bound on the Codeword Length of ω'

We can bound $c'_E(x)$ for $x \geq 4$ from below as follows:

$$c'_E(x) \geq 2 + \log_2(x) + \log_2\log_2(x) + \cdots + \log_2^{k^*-1}(x) + \Lambda^{k^*}(x) \tag{A3}$$

by repeatedly using the inequalities for any real number $x \geq 1$,

$$\Lambda^i(x) > \log_2^i(x) \quad (i = 1, 2, \cdots) \tag{A.4}$$

where k^* is the unique positive integer satisfying $\Lambda^{k^*}(x) = 3$. Here we use the convention that $\log_2^k(x) = 0$ if $\log_2^k(x)$ is negative or undefined.

Now we assume $k^* \geq 2$. Then, $k^* - 1 \geq 1$ and hence $\Lambda^{k^*-1}(x)$ must be an integer. Therefore, by the definition (A2) of $\Lambda(x)$ it follows that $4 \leq \Lambda^{k^*-1}(x) \leq 7$, from which we have $\log_2^{k^*-1}(x) \leq 7$. Therefore, we obtain the following inequalities:

$$\begin{aligned} \log_2^{k^*}(x) &\leq \log_2 7 \\ \log_2^{k^*+1}(x) &\leq \log_2\log_2 7 \\ \log_2^{k^*+2}(x) &\leq \log_2\log_2\log_2 7 < 1. \end{aligned}$$

Hence,

$$w^*_2(x) \leq k^* + 2.$$

Thus in the case of $k^* \geq 2$, we obtain the lower bound as follows:

$$\begin{aligned} c'_E(x) &\geq \log^*_2(x) + 5 - \left(\log_2^{k^*}(x) + \log_2^{k^*+1}(x) + \log_2^{k^*+2}(x)\right) \\ &\geq \log^*_2(x) + 5 - (\log_2 7 + \log_2\log_2 7 + \log_2\log_2\log_2 7) \\ &\geq \log^*_2(x) + 5 - 4.877\cdots \\ &> \log^*_2(x) \equiv l_L(x). \end{aligned} \tag{A5}$$

Remark 10: In the above derivation, when $k^* = 1$ it holds that $\Lambda^{k^*-1}(x) = \Lambda^0(x) = x$. Hence, $\Lambda^{k^*}(x) = 3$ is equivalent to $4 \leq x < 8$. We have already shown that $c'_E(x) > \log^*_2(x)$ for $4 \leq x \leq 7$. Therefore, it may occur that $c'_E(x) < \log^*_2(x)$ only for $7 < x < 8$, and it really occurs.

B. Upper Bound on the Codeword Length of ω'

For convenience, we set

$$g(x) = \log_2 x + 1 \quad (\text{real } x > 1) \tag{A6}$$

and observe that

$$\Lambda(x) \leq g(x) \quad (\text{real } x > 1). \tag{A7}$$

Now we investigate the relation between $g^r(x)$ (the r-fold composition of $g(x)$) and $\log_2^r(x)$. For $r = 2$, we have

$$\begin{aligned} g^2(x) &= \log_2(\log_2 x + 1) + 1 \\ &= \log_2\left\{\log_2 x\left(1 + \frac{1}{\log_2 x}\right)\right\} + 1 \\ &= \log_2\log_2 x + \log_2\left(1 + \frac{1}{\log_2 x}\right) + 1 \\ &\leq \log_2^2(x) + \frac{a}{\log_2 x} + 1 \end{aligned}$$

where $a = \log_2 e$. For $r = 3$, it holds in the same way that

$$g^3(x) \leq \log_2^3(x) + \frac{a^2}{\log_2^2(x)\log(x)} + \frac{a}{\log_2^2(x)} + 1.$$

After repeating similar calculations, we have, for any positive integer k,

$$\Lambda^k(x) \leq g^k(x) \leq \log_2^k(x) + A_k(x) + 1 \tag{A8}$$

where

$$A_k(x) = \sum_{j=0}^{k-2} \frac{a^{j+1}}{\log_2^{k-1}(x)\log_2^{k-2}(x)\cdots\log_2^{k-1-j}(x)} \tag{A9}$$

and we have set $A_1(x) = 0$.

Now we determine the positive integer w for any real number $x \geq 4$ such that

$$2 \leq \log_2^w(x) < 4 = \exp_2(2). \tag{A10}$$

Since for any integer l $(w \geq l \geq 1)$

$$\log_2^{w-l}(x) \geq \exp_2^l(2)$$

we have for any integer k satisfying $2 \leq k \leq w$

$$A_k(x) \leq \sum_{j=0}^{k-2} \frac{a^{j+1}}{\exp_2^{w-k+1}(2)\exp_2^{w-k+2}(2)\cdots\exp_2^{w-k+j+1}(2)}.$$

Therefore,

$$\begin{aligned}
&\sum_{k=1}^{w} A_k(x) \\
&\quad \le \sum_{k=2}^{w}\sum_{j=0}^{k-2} \frac{a^{j+1}}{\exp_2^{w-k+1}(2)\exp_2^{w-k+2}(2)\cdots\exp_2^{w-k+j+1}(2)} \\
&\quad = \sum_{j=0}^{w-2} a^{j+1} \\
&\qquad \times \sum_{k=j+2}^{w} \times \frac{1}{\exp_2^{w-k+1}(2)\exp_2^{w-k+2}(2)\cdots\exp_2^{w-k+j+1}(2)} \\
&\quad = \sum_{j=0}^{w-2} a^{j+1} \\
&\qquad \times \sum_{i=0}^{w-j-2} \frac{1}{\exp_2^{i+1}(2)\exp_2^{i+2}(2)\cdots\exp_2^{i+j+1}(2)} \\
&\quad \le \sum_{j=0}^{w-2} a^{j+1} \sum_{i=0}^{w-j-2} \frac{1}{\{\exp_2^{i+1}(2)\}^{j+1}}.
\end{aligned}$$

Applying an obvious inequality $\exp_2^{i+1}(2) \ge \{\exp_2(2)\}^{i+1}$, we obtain

$$\sum_{k=1}^{w} A_k(x) \le \sum_{j=0}^{w-2} a^{j+1} \sum_{i=0}^{w-j-2} \frac{1}{\{\exp_2(2)\}^{(i+1)(j+1)}}. \quad \text{(A11)}$$

To upper-bound the right-hand side, we continue the computation as follows:

$$\begin{aligned}
\sum_{k=1}^{w} A_k(x) &\le \sum_{j=0}^{w-2} a^{j+1} \sum_{i=0}^{\infty} \frac{1}{(4^{j+1})^{i+1}} \\
&\le \sum_{j=0}^{w-2} \left(\frac{a}{4}\right)^{j+1} \cdot \frac{4}{3} \\
&\le \frac{a}{3} \sum_{j=0}^{\infty} \left(\frac{a}{4}\right)^{j} \\
&= \frac{4a}{3(4-a)}. \quad \text{(A12)}
\end{aligned}$$

On the other hand, $4a/3(4-a) = 0.9092\ldots$ because $a = \log_2 e = 1.4427\ldots$. Hence, from (A12) we have

$$\sum_{k=1}^{w} A_k(x) < 1. \quad \text{(A.13)}$$

From (A8), (A10), and (A13), we obtain

$$\Lambda^w(x) \le g^w(x) \le \log_2^w(x) + A_w(x) + 1 < 4 + 1 + 1 = 6$$

which means $\Lambda^w(x) \le 5$, because $\Lambda^w(x)$ must be an integer.

Now let us investigate the case, when $\Lambda^w(x)$ is 4 or 5. In this case, owing to (A10), we have

$$1 \le \log_2^{w+1}(x) < 2, \quad 0 \le \log_2^{w+2}(x) < 1.$$

Therefore, there is the relation between w defined in (A10) and $w_2^*(x)$ defined in (4.4) such that

$$w_2^*(x) = w + 2. \quad \text{(A14)}$$

Hence, we obtain

$$\Lambda^{w+1}(x) = 3 \le \log_2^{w+1}(x) + \log_2^{w+2}(x) + 2. \quad \text{(A15)}$$

Summarizing (A1), (A8), (A14), and (A15), we obtain an upper bound for the codeword length $c'_E(x)$ of the modified Elias ω' code for any real $x \ge 4$ in the case of $\Lambda^w(x) = 4$ or 5

$$\begin{aligned}
c'_E(x) &= 2 + \sum_{i\ge 1:\Lambda^i(x)\ge 3} \Lambda^i(x) \\
&\le 2 + \sum_{i\ge 1:\Lambda^i(x)\ge 3} g^i(x) \\
&\le l_U(x) \quad \text{(A16)}
\end{aligned}$$

where we have set

$$l_U(x) = \log_2^*(x) + w_2^*(x) + 3. \quad \text{(A17)}$$

By virtue of a similar discussion, we can derive the same upper bound (A16) also for the case $\Lambda^w(x) \le 3$.

Thus we have established Theorem 3 by means of the above a), b), and Remark 10.

ACKNOWLEDGMENT

The authors wish to thank H. Yamamoto and A. Barg for bringing [4] and [7], respectively, to their attention.

REFERENCES

[1] P. Elias, "Universal codword sets and representation of the integers," *IEEE Trans. Inform. Theory*, vol. IT-21, pp. 194–203, 1975.
[2] J. L. Bentley and A. C. Yao, "An almost optimal algorithm for unbounded searching," *Inform. Processing Lett.*, vol. 5, no. 3, pp. 82–87, 1976.
[3] S. Even and M. Rodeh, "Economical encoding of commas between strings," *Commun. ACM*, vol. 21, no. 4, pp. 315–317, 1978.
[4] Q. F. Stout, "Improved prefix encodings of the natural numbers," *IEEE Trans. Inform. Theory*, vol. IT-26, pp. 607–609, 1980.
[5] S. K. Leung-Yan-Cheng and T. M. Cover, "Some equivalences between Shannon entropy and Kolmogorov complexity," *IEEE Trans. Inform. Theory*, vol. IT-24, pp. 331–338, 1978.
[6] J. Rissanen, "A universal prior for integers and estimation by minimum description length," *Ann. Statist.*, vol. 11, pp. 416–431, 1983.
[7] V. I. Levenshtein, "On the redundancy and delay of decodable coding of natural numbers," *Probl. Cybern.*, vol. 20, pp. 173–179, 1968, (in Russian).