Improved Prefix Encodings of the Natural Numbers

QUENTIN F. STOUT

***Abstract*—Two classes of encodings of the natural numbers are introduced, all of which are universal asymptotically optimal. The asymptotically best encodings in these classes are determined and are found to improve on previous encodings. The results are related to channel capacity and unbounded searching.**

I. Introduction

Suppose there is an infinite sequence $\sigma_1, \sigma_2, \cdots$ of message strings to be transmitted in the order given through a noiseless channel. Suppose there are no *a priori* bounds on the length of each message, so a procedure must be devised to separate one from the next. Usually one adds a new symbol, a "comma", to separate the messages and recodes each message in this extended character set. Unfortunately this makes each string grow by an amount proportional to its length, resulting in a fixed decrease in channel capacity. If the messages are very long then better encodings have been devised by Elias [2] and Even and Rodeh [3]. In Elias' terminology, these encodings are *universal asymptotically optimal*, i.e., for each encoding there is a function R: $(0,\infty)\to(0,\infty)$, with $\lim_{t\to\infty} R(t)=1$, such that, for any set of numbers with probability distribution P and entropy $H(P)$, if $E(P)$ is the resulting average codeword length then $E(P)/\max(1, H(P)) \leqslant R(H)$.

This correspondence considers a class of prefix codes which includes those of Elias and Even and Rodeh as special cases. We define an ordering of codes called "improvement" and use it to analyze this class. A second class is introduced which is somewhat better than the first, removing a redundancy in the first class. We also show how a code in either class determines a search algorithm for unbounded searching. Since better codes yield better search algorithms, this work gives a slight improvement upon the search algorithm of Bentley and Yao [1]. Finally, some of these codes are compared over a realistic range of message lengths.

II. R_l Codes

For simplicity, all messages and codes are binary and all logarithms are to the base 2. If the messages are $\sigma_1, \sigma_2, \cdots$, and if E is an encoding function, then our resulting message stream will be

$$E(\text{length}(\sigma_1))0\sigma_1E(\text{length}(\sigma_2))0\sigma_2\cdots.$$

This simplifies the description of the code, as well as the actual encoding/decoding, because we need to encode only natural numbers. The 0 signals the end of $E(\text{length}(\sigma))$ and the start of σ in a manner that is made clear below. We allow for the possibility that $\text{length}(\sigma)=0$. This differs from Even and Rodeh, who use the null message to signal the end of all messages. One could use that interpretation in our codes also, but it is not required.

Let $B(n,l)$ denote the l-bit binary representation of n, where $0\leqslant n\leqslant 2^l-1$. For example, $B(6,5)=00110$. The standard binary representation of n $(n\geqslant 1)$ is $B(n,\lfloor\log n\rfloor+1)$. Note that the leftmost bit of $B(n,\lfloor\log n\rfloor+1)$ is a 1. For any $l\geqslant 2$, we define an encoding R_l of the natural numbers by

$$R_l(n)=\begin{cases} B(n,l), & \text{if } 0\leqslant n\leqslant 2^l-1,\\ R_l(\lfloor\log n\rfloor+1)B(n,\lfloor\log n\rfloor+1), & \text{if } n\geqslant 2^l.\end{cases}$$

Manuscript received September 5, 1979; revised January 2, 1980.

The author is with the Department of Mathematical Sciences, State University of New York at Binghamton, Binghamton, NY 13901.

For example, $R_3(1000)=100\ 1010\ 1111101000$, where for readability there are blanks between $\lfloor\log(\lfloor\log 1000\rfloor+1)\rfloor+1$, $\lfloor\log 1000\rfloor+1$, and 1000. Each such group is called a *block*. The decoding is quite simple. Suppose R_3 was used and the first ten bits are 1101000110. Since we are using R_3 we look at the first three bits, which represent a six. The first bit of the second block is a 1, so it is not a message string. Therefore we look at the six bits of the second block, which represent 35. The next bit is a 0, so the 35 bits following the 0 are the first message string. Following those 35 bits would be a block of three bits starting the representation of the second message.

R_3 is the code R of Even and Rodeh. R_2 is similar to Elias' "penultimate" code (there is no "ultimate" code), except that his code uses $R(\lfloor\log n\rfloor)$ instead of $R(\lfloor\log n\rfloor+1)$ in the recursive step. His code is intermediate between R_2 and the S_2 code we introduce later.

Notice that if $l<k$, then $R_l(n)$ has at least as many blocks as $R_k(n)$, and all blocks but the leftmost one of $R_k(n)$ are also in $R_l(n)$. If the leftmost block of $R_k(n)$ represents 2^l-1 or less than $R_l(n)$ and $R_k(n)$ will have the same number of blocks and the leftmost block of $R_l(n)$ will be $k-l$ bits shorter. Therefore

$$\min_n[\text{length}\,(R_l(n))-\text{length}(R_k(n))]=l-k. \tag{1}$$

On the other hand, if the leftmost block of $R_k(n)$ represents 2^k-1, then on reading from right to left $R_l(n)$ and $R_k(n)$ will be identical until $R_k(n)$ ends, at which point R_l will still need to encode k. Therefore

$$\max_n[\text{length}(R_l(n))-\text{length}(R_k(n))]=\text{length}(R_l(k)). \tag{2}$$

Further, R_l will be $\text{length}(R_l(k))$ bits longer than R_k for any x_i defined by $x_1=2^k-1$ and $x_{i+1}=2^{x_i}-1$. Therefore

$$\limsup_n[\text{length}(R_l(n))-\text{length}(R_k(n))]=\text{length}(R_l(k)). \tag{3}$$

Finally, if $R_k(n)$ has two or more blocks then the leftmost block represents at least $k+1$. R_k takes k bits to encode this and R_l takes $\text{length}(R_l(k+1))$, giving

$$\liminf_n[\text{length}(R_l(n))-\text{length}(R_k(n))]=\text{length}(R_l(k+1))-k. \tag{4}$$

We are interested in long messages, which motivates the following definitions: an encoding A of the natural numbers is a *weak improvement* of encoding B if $\liminf_n[\text{length}(B(n))-\text{length}(A(n))]\geqslant 0$ and $\limsup_n[\text{length}(B(n))-\text{length}(A(n))]>0$. A is an *improvement* of B if $\liminf_n[\text{length}(B(n))-\text{length}(A(n))]>0$. (It is possible to base a definition of improvement in terms of the rate at which the R function in Elias' definition approaches 1, but for our purposes the definitions given are most useful.) Facts (3) and (4) show that R_3 is an improvement of R_2, R_7 is a weak improvement of R_3 (and hence an improvement of R_2), R_4 is an improvement of R_2, and R_5 is a weak improvement of R_2. Straightforward checks show that these are the only R_l codes which weakly improve R_2 or R_3.

Theorem 1: If R_k is a weak improvement of $R_l(k\neq l)$ then $l=2$ or $l=3$.

Proof: Because of (3) we may assume $k>l$. Suppose $2^l-1>k$. Then $\text{length}(R_l(k+1))=l$, so by (4) R_k is not a weak improvement of R_l. Suppose $k\geqslant 2^l-1$ and $l\geqslant 4$. Then any $n>k+1$ such that the leftmost block of $R_k(n)$ represents $k+1$ will have a representation $k-\text{length}(R_l(k+1))$ bits shorter in R_l. By (1) this is at least

$$k-(\text{length}(R_2(k+1))-2+l). \tag{$*$}$$

$l\leqslant\log(k+1)$, so $(*)\geqslant k+2-\text{length}(R_2(k+1))-\log(k+1)$. Since

$l \geqslant 4$, $k \geqslant 15$. For $m \geqslant 12$, $m - \text{length}(R_2(m)) - \log(m) \geqslant 0$, so $(*) \geqslant 1$. Therefore R_k is not a weak improvement of R_l. □

III. S_l Codes

The work of Bentley and Yao [1, theorem A] shows that there is not much room for improvement upon the R_l. Elias mentions one possibility but notes that no improvement would occur until the messages are longer than Eddington's estimate of the number of particles in the universe. Nonetheless, for small l the following codes offer some realistic improvement. For $l \geqslant 2$ define S_l by

$$S_l(n) = \begin{cases} B(n,l), & \text{if } 0 \leqslant n \leqslant 2^l - 1, \\ S_l(\lfloor \log n \rfloor - l)B(n, \lfloor \log n \rfloor + 1), & \text{if } n \geqslant 2^l. \end{cases}$$

Decoding is similar to R_l and will not be given. The S_l code removes the redundancy in R_l whereby a 1 in the $(l+1)$st bit implies that the first l bits represent a number which is at least $l+1$. The S_l are complete prefix codes while the R_l are not. Elias' code is also complete, but his elimination of redundancy favored very small numbers.

Theorem 2: For any $l \geqslant 2$,

a) $\min_n[\text{length}(R_l(n)) - \text{length}(S_l(n))] = 0$,
b) $\lim \inf_n[\text{length}(R_l(n)) - \text{length}(S_l(n))] = 0$,
c) $\lim \sup_n[\text{length}(R_l(n)) - \text{length}(S_l(n))] = \infty$.

Therefore S_l is a weak improvement of R_l.

Proof: a) is obvious. To show b), define the following sequence: $n_1 = 2^l - 1$, and $n_{k+1} = 2^{n_k} - 1$. Then $R_l(n_{k+1})$ is $l + \sum_{i=1}^{k} n_i$ bits while $S_l(n_{k+1})$ is $B(n_1 - l - 1, l)B(n_2 - l - 1, n_1) \cdots B(n_{k+1}, n_k)$, which is also $l + \sum_{i=1}^{k} n_i$ bits. To show c), let n be such that $S_l(n)$ has k blocks, each of which are all ones. Then $R_l(n)$ has at least $k+1$ blocks where each but the first and last is one bit longer than the corresponding block of $S_l(n)$. □

Since their encoding/decoding algorithms are nearly identical, S_l is to be preferred to R_l. Transitivity shows that S_7 is a weak improvement of R_3 and an improvement of R_2, S_3 and S_4 improve R_2, and S_5 weakly improves R_2. Further, if $l < k$, then $\lim \sup_n[\text{length}(S_l(n)) - \text{length}(S_k(n))] = \infty$. Somewhat surprisingly, that is all that can be said, as the following theorem indicates. Its proof is omitted, being similar to Theorems 1 and 2.

Theorem 3: For any k and l, $k \neq l$,

a) if S_k is a weak improvement of R_l then $l = 2$ or 3, and
b) S_k is not a weak improvement of S_l.

IV. Searching

Bentley and Yao [1] have shown that there is a connection between prefix codes for the natural numbers and searching for the integral zero of a monotone function with no bound on its location. If a fixed search method is used then whether each probe is too high or low translates into a 0 or 1 of a binary string which uniquely identifies the zero. They ask if, conversely, every prefix code determines a search algorithm. A detailed examination of this correspondence appears in Stout [4]. The answer depends on how much one is willing to rearrange an encoding, since any search-generated encoding must be in lexicographic order, and for the natural numbers this implies that the number of initial ones in the encoding of n tends to infinity as n does. For the R_l and S_l codes this rearrangement is easily carried out, as shown below. Bentley and Yao describe an "almost optimal" algorithm for searching which is basically a rearranged version of R_2. They first determine the number of blocks needed and then determine the blocks from left to right. For example, such a representation of 1000 is **11110** 1 00 010 111101000, while $R_2(1000) =$ **11** 100 1010 1111101000 **0**, where blanks have been inserted between blocks and the boldface bits are moved between the codes. Incidentally this rearrangement of R_2 is related to the code Elias mentions as a theoretical improvement upon his "penultimate" code. The initial block is just a unary encoding of the number of blocks and hence could be improved by coding it using R_2. Any R_l or S_l can be improved this way, but again, no improvement will occur in this universe.

Bentley and Yao's search algorithm can be improved by finding the number of blocks in $R_3(n)$ and then filling in the blocks. For example, to use R_3 to find a zero located at 1000, we first determine if the zero is less than 2^3, 2^7, or 2^{127}. The yes response to the last probe indicates that three blocks are needed. The first block is $1ab$. To determine a we determine if the zero is less than 2^{31} ($2^{31} - 1$ being the largest three-block number starting with $10b$). Knowing that a is 0, we determine if the zero is less than 2^{15}. Since this is also true, we know the first block is 100, and hence the second block is $1cde$. Continuing in this manner we would determine the rest of the blocks. Using the S_l codes produces even better searches (see the Appendix).

TABLE I
A Comparison of Some Code Lengths

Length of Message	Length of Prefix	(bits)				
(bits)	R_2	S_2	R_3	S_3	S_5	Septenary
10	10	7	8	8	5	9
100	13	13	11	11	13	12
1000	20	16	18	14	16	15
10^4	24	21	22	22	20	18
10^5	28	24	26	25	23	21
10^6	31	28	29	29	26	27
10^7	35	32	33	33	30	30
10^8	38	35	36	36	33	33
10^9	41	38	39	38	36	36
10^{10}	46	42	44	43	40	39
10^{11}	49	46	47	47	43	45
10^{12}	52	49	50	50	52	48
10^{13}	56	53	54	54	56	51
10^{14}	59	56	57	57	59	54
10^{15}	62	59	60	60	62	57

V. Conclusion

Elias and Even and Rodeh introduced prefix encodings of the natural numbers that are universal asymptotically optimal but that are capable of some improvement (as are any encodings). We first extend these codes to the class R_l, all of which are universal asymptotically optimal, and analyze when one code improves upon another. For example, R_3 improves R_2, and R_7 weakly improves R_3. However, each R_l has a slight redundancy whereby a 1 in the $(l+1)$st bit implies that the first l bits represent a number that is at least $l+1$. The S_l codes remove this redundancy, yielding improved codes with almost no additional encoding/decoding difficulty. These improved codes also yield slight improvements in the unbounded searching problem considered by Bentley and Yao.

It must be understood however that most of the improvements discussed here are in the realm of theory and not practice. For example, even though $\lim \sup_n[\text{length}(R_7(n)) - \text{length}(S_7(n))] = \infty$, n must be at least 2^{127} ($> 10^{38}$) for there to be any difference in the lengths. For those more interested in the physical world, Table I lists the prefix length versus message length for R_2, S_2, R_3, S_3, S_5, and a septenary code. The septenary code uses $000, 001, \cdots, 111$ to represent $0, 1, \cdots, 6$, " , " respectively. Each message is prefixed with the symbol for the comma followed by the message length written in base 7. For example, if σ as 102 bits then it is encoded as 010000100111σ. Asymptotically the septenary code is quite inferior.

Appendix

The following algorithm is a search algorithm corresponding to S_l. When it is finished, the root is at the position n. It assumes that there is some way to test if the root is greater than a given value. For example, this is possible for monotone functions. The first line recursively defines a function largest(a, c) which gives the largest number such that S_l has c blocks and the leftmost block represents the number a.

```
largest($a,c$)$\leftarrow$*if* $c=1$ *then* $a$ *else* largest $(2^{a+l+1}-1, c-1)$;
blocks$\leftarrow$1;
*while* root$>$largest $(2^{l}-1,$ blocks$)$ *do* blocks$\leftarrow$blocks$+1$;
length$\leftarrow$0;
$n\leftarrow 0$;
*for* $b\leftarrow$blocks *downto* 1 *do begin*
   *for* $k\leftarrow n+l$ *downto* 1 *do*
      *if* root$>$largest (length$+2^{k+1}-1, b$)
         *then* length$\leftarrow$length$+2^{k-1}$;
   $n\leftarrow$length
   length$\leftarrow 2^{n+l}$;
*endfor*;
```

The algorithm for R_l is nearly identical. The $l+1$ is removed from the first line, the fifth line is $n\leftarrow l+1$, and in lines 7 and 11 "$n+l$" is changed to "$n-1$".

References

[1] J. L. Bentley and A. C. Yao, "An almost optimal algorithm for unbounded searching," *Inform. Processing Letters*, vol. 5 pp. 82–87, 1976.

[2] P. Elias, "Universal codeword sets and representations of the integers," *IEEE Trans. Inform. Theory*, vol. IT-21, no. 2 pp. 194–203, Mar. 1975.

[3] S. Even and M. Rodeh, "Economical encoding of commas between strings," *Comm. ACM*, vol. 21, no. 4, pp. 315–317, Apr. 1978.

[4] Q. F. Stout, "Searching and sorting for infinite ordered sets," to be published.

Phonetic Test Sentences

EDWARD F. MOORE, MEMBER, IEEE

***Abstract*—A method has been devised for obtaining lists of words which use each sound of the English language (with American pronunciation) exactly once. The method is sufficiently automatic that it can be carried out by a person knowing no phonetics, or even by a digital computer. Trying the procedure with slight modifications produced several such lists, one of which could be transformed into a sentence: "Hum, thou whirring fusion; yes, Joy, pay each show; vie, thaw two wool dock bags." This sentence, or one of the lists, might be usable for testing telephone transmission, testing speech defects, or providing the pronunciation key at the bottom of the page of a dictionary. Care was taken to avoid the use of words which have different pronunciations in different parts of the U.S. or in different phrases. Although this list was composed in terms of an assumed list of 42 English sounds, it seems quite likely that anyone who considers a slightly different list of sounds to be the basic ones could do the same thing by applying this method, with appropriate modifications, to his list.**

Manuscript received February 1, 1980.

The author is with the Computer Sciences Department, University of Wisconsin—Madison, Madison, WI 53706.

Editor's Note: This Correspondence first appeared as a Bell Laboratories internal memorandum in September 1956, and is now published for the first time (at the Editor's request) because of renewed interest in its contents.

A fairly explicit procedure is given for constructing the shortest possible sentence containing all the sounds of English. The following list of 42 speech sounds of English was chosen somewhat arbitrarily, although the same method could easily be repeated with any other designated list. The list was arranged with the hard-to-use sounds first, and the easy ones last. The symbols are those of the International Phonetic Association:

[ʒ, ɜ, ju, ʊ, u, i, ɔɪ, *a*, hw, ʌ, aʊ, *θ*, aɪ,
æ, ð, ɛ, ɔ, tʃ, o, e, ŋ, ɪ, dʒ, j, w, v, d,
l, m, h, ʃ, f, g, p, k, s, n, b, t, ə, z, r].

This list was originally obtained by arranging the sounds in the approximate order of increasing frequency, with some revisions, since [ɜ] is always followed by [r], [ʊ] is rare in words which do not have a pronounciation using [u] or [ʌ], and other specific corrections. The first few times the method was applied to the original list it failed, but the list was improved after each failure by moving the leftover sounds nearer to the front of the list. After four attempts (in which the leftover sounds were all vowels, since our list has too many vowels per consonant), the list above was obtained, with the vowels mainly in the first half of the list. The method then worked on five of the next six attempts, even though deviations from it were made to try to find words which could be more plausibly joined together into sentences.

The method consists of choosing words one at a time, marking off the above list those sounds which have been used, and choosing each word to use sounds which are as close to the beginning of the list as possible. Be sure to use up the first unused sound each time and try to avoid using any sounds near the end of the list. Try to use words which have no more vowels than consonants. The following lists of words were obtained on the first three trials with the list above:

1) fusion, err, pull, woo, each, joy, dock, whet, young, thou, thaw, vie, ash, home, say, big;
2) vision, err, youth, full, woo, each, joy, pod, whey, young, house, shy, cam, though, egg, awe;
3) fusion, whirring, wool, two, each, joy, dock, hum, thou, thaw, vie, bag, yes, show, pay.

Each of these word lists use all 42 sounds except [z], which can be added to most nouns (plural) or verbs (third person singular). The omission of [z] was done to give more freedom to arrange the words into a sentence. List 3 was rearranged to give the sentence finally chosen, which is [hʌm, ðaʊ hwɜrɪŋ fjuʒən; jɛs dʒɔɪ, pe itʃ ʃo; vaɪ, *θ*ɔ tu wʊl d*a*k bægz.]

All the words chosen were checked in *A Pronouncing Dictionary of American English*, by Kenyon and Knott, to get their phonetic alphabet representation. No words were used which had more than one pronounciation listed, except for the substitution of [ɪu] for [ju] and of [ɝ] for [ɜr], both of which seem to be unavoidable. This attempt, to be sure that whoever reads the sentence would utter all 42 sounds, resulted in the exclusion of many common English words (garage, father, was, a, the, and, but, in, etc.) which, if allowed, would have permitted the construction of a sentence much more like colloquial English.

Incidentally, the corresponding problem for letters was solved by C. E. Shannon, H. O. Pollak, and the author: "Squdgy fez, blank jimp crwth vox.", "Batz jink frev squdgy cwm phlox." and "Shiv fyrd cwm qung jab tez phlox.", all of which are legal and meaningful English sentences.