least $d_0^{(j)} = d_1^{(j)} d_2^{(j)}$, where $d_1^{(j)}$ is given in Lemma A1. Since

$$d_{0M}(r_0) = \min_j \left[d_0^{(j)} \right],$$

we choose $d_0^{(j)} = d_0$ for all j. Computing the overall rate r_0 by (A26), we have

$$1 - r_2^{(1)} = \left[r_1/(1 - r_1) \ln (1 - r_1)^{-1} \right] (1 - r_0/r_1), \quad \text{(A30)}$$

where $r_1 = r_1^{(1)}$. Substitution of (A30) into $d_{0M}(r_0) = d_1^{(1)} d_2^{(1)}$ gives (A29).

This theorem shows that the minimum distance for the code A_M is larger than that for the code A.

References

[1] P. Elias, "Error-free coding," *IRE Trans. Inform. Theory*, vol. PGIT-4, pp. 29–37, Sept. 1954.

[2] S. Hirasawa, M. Kasahara, Y. Sugiyama, and T. Namekawa, "Certain generalizations of concatenated codes—Exponential error bounds and decoding complexity," *IEEE Trans. Inform. Theory*, vol. IT-26, pp. 527–534, Sept. 1980.

[3] ——, "An improvement of error exponents at low rates for the generalized version of concatenated codes," *IEEE Trans. Inform. Theory*, vol. IT-27, pp. 350–352, May 1981.

[4] G. D. Forney, Jr., *Concatenated Codes*. Cambridge, MA: M.I.T., 1966.

[5] J. E. Savage, "A note on the performance of concatenated codes," *IEEE Trans. Inform. Theory*, vol. IT-16, pp. 512–513, July 1970.

[6] E. A. Bucher and J. A. Heller, "Error probability bounds for systematic convolutional codes," *IEEE Trans. Inform. Theory*, vol. IT-16, pp. 214–224, Mar. 1970.

A Comparison of Enumerative and Adaptive Codes

JOHN G. CLEARY AND IAN H. WITTEN, MEMBER, IEEE

Abstract—**The problem of noiselessly encoding a message when prior statistics are known is considered. The close relationship of arithmetic and enumerative coding for this problem is shown by computing explicit arithmetic coding probabilities for various enumerative coding examples. This enables a comparison to be made of the coding efficiency of Markov models and enumerative codes as well as a new coding scheme intermediate between the two. These codes are then extended to messages whose statistics are not known *a priori*. Two adaptive codes are described for this problem whose coding efficiency is upper-bounded by the extended enumerative codes. On some practical examples the adaptive codes perform significantly better than the nonadaptive ones.**

I. Introduction

A SEQUENCE of symbols $x_1, x_2, \cdots, x_N$ is noiselessly encoded as another sequence $y_1, y_2, \cdots, y_M$ when the original sequence x can be uniquely determined from y. If the x_i and y_i are drawn from the same alphabet (say $A = \{0, 1\}$), then y is a compressed form of x when $M < N$. Compression is important when x must be transmitted down a slow costly communication link. Then it may be preferable to send y and have the receiver reconstitute x from this [1].

Manuscript received July 14, 1982; revised March 8, 1983. This research was supported by the Natural Sciences and Engineering Research Council of Canada.

The authors are with the Man–Machine Systems Group, Department of Computer Science, University of Calgary, Canada.

We consider two techniques for achieving such compression. The first, enumerative encoding [2], involves enumerating the entire set of possible strings x and then sending the index y, within some suitable ordering, of the actual string x. Providing the receiver has agreed with the sender about the universe of possible messages, and a suitable ordering of them, then x can be reconstituted from its index.

The second technique, arithmetic coding, has been discussed recently by a number of authors [3], [4], [5], [6]. It differs from enumerative coding in that it assumes an explicit probabilistic model of the symbol sequence constituting the message. After a sequence $x_1, \cdots, x_{i-1}$, the possible values for x_i are ordered arbitrarily, say $w_1, \cdots, w_m$, and assigned probabilities $p(w_1), \cdots, p(w_m)$. The encoding y is computed iteratively using only these probabilities at each step.

Arithmetic coding has some interesting properties which are important in what follows.

- The symbol probabilities $p(w_k)$ may be different at each step. Therefore a wide range of algorithms can be used to compute the $p(w_k)$, independent of the arithmetic coding method itself.
- It is efficient and can be implemented using a small fixed number of finite arithmetic operations as each symbol of x is processed.

0018-9448/84/0300-0306$01.00

- The output code length is determined by the probabilities of the symbols x_i and can arbitrarily closely approximate the sum $\sum -\log_2 p(x_i)$ if arithmetic of sufficient accuracy is used.
- Production of the output code y can begin before the whole of the input sequence x has been seen.

It has been shown by Rissanen and Langdon [4] and Rissanen [7] that arithmetic and enumerative coding are equivalent. In Section II, this relationship is made explicit by calculating arithmetic probabilities from the enumeration of the possible strings x. Using these probabilities both methods will construct encodings of the same length. We also show that such "faithful" probability estimates can be easily computed for two interesting coding problems from Cover [2].

Unfortunately, this method incurs very considerable complications in practice, when higher order models are used. However, arithmetic codes lend themselves naturally to approximation. Probabilities can easily be calculated which are nearly "faithful" but not quite so. Two such methods are examined in Section III, one encompassing Markov models. For both approximations, we bound the decrease in coding efficiency over the exact enumerative method.

Implicit in both enumerative and arithmetic coding is an underlying model of the string x. (An example of such a model is a count of the number of 0's and 1's which occur in x.) In the former case this allows the possible strings x to be enumerated; in the latter it provides the basis for calculating the symbol probabilities. In general it is necessary to encode strings where the parameters of the model are not known in advance. To solve this "parameterized" encoding problem, codes analogous to those of Section III can be constructed by first transmitting the model parameters using some suitable scheme and then using an enumerative code (or an approximation) to encode the sequence itself. We will refer to these codes as *parameterized enumerative* (PE) codes. Alternatively, the model parameters can be estimated "adaptively" as y is received, with the advantage that encoding and decoding may proceed in real time with neither prior information nor blocked transmission.

While adaptive coding saves sending the model, a penalty is paid because the probabilities only approximate the true values computed from a complete prior knowledge of the sequence. Therefore the encoded sequence y will be correspondingly longer. Section IV presents the main result of this paper that the adaptive codes are intimately related to the PE codes. They produce encodings whose length is closely bounded by that of the PE codes.

In the final section, these theoretical results are supported by some examples where English text and programming language text are compacted. These show that in some cases the adaptive codes can be a significant improvement on the PE codes. Support for the more general efficacy of adaptive codes is provided by the experience of Langdon and Rissanen [8] in compressing black-and-white images using adaptive codes and by the results reported by Cleary and Witten [9]. The latter describe a more efficient but less theoretically tractable adaptive code which achieved 2.2 bits/ character over a long sequence of English text.

II. Enumerative Coding

Consider some finite set $\Omega \subset A^*$ strings over an alphabet A, which serves as the domain of possible strings x which might be encoded. For example, Ω might be the set of sequences containing exactly four 0's and four 1's. An enumerative code for Ω is a mapping from it to the binary integers 0 through $|\Omega| - 1$, each of which is represented by $\lceil \log_2 |\Omega| \rceil$ bits. Note that both the sender and receiver must have complete knowledge of Ω before transmission begins and have agreed on an appropriate ordering. In Sections IV and V a more general problem is considered where only the alphabet A and a model of Ω are known beforehand.

Denote the set of prefixes of strings in Ω by

$$\bar{\Omega} \equiv \{u: uv \in \Omega\, v \in A^*\},$$

so that $\Omega \subseteq \bar{\Omega} \subset A^*$. A minor technical problem arises because of the need to know when the string x ends. To ensure no ambiguity, we assume that no string in Ω is a proper prefix of another string in Ω:

$$u, uv \in \Omega \Rightarrow v = \Lambda,$$

where Λ is the empty string. In most of the examples studied in this paper this unambiguous prefix property holds because the length of x is implied by the model which describes Ω. In other cases it may be necessary to add a terminating symbol to x.

An important role in the development will be played by the *enumeration function* E: $\bar{\Omega} \to Z^+$, which counts for any string u how often if appears as a prefix in Ω:

$$E(u) \equiv |\{v \in A^*: uv \in \Omega\}|.$$

Note that $E(\Lambda) = |\Omega|$.

An arithmetic coding is defined by the values of the next-symbol probability estimates, contingent upon that part of the sequence seen so far, namely, $x_1, \cdots, x_i$. For each symbol $w \in A$, the probability that it will follow $x_1, \cdots, x_i \in \bar{\Omega}$ will be computed as the ratio of the number of strings where w follows $x_1, \cdots, x_i$ divided by the total number of strings following $x_1, \cdots, x_i$:

$$p(w|x_1, \cdots, x_i) = \frac{E(x_1, \cdots, x_i, w)}{E(x_1, \cdots, x_i)}. \tag{1}$$

Probabilities which obey this relationship are described as being *faithful* to the enumeration.

The number of bits required to transmit the arithmetic coding of $x_1, \cdots, x_N$ is simply the sum of the bits needed for each symbol of the sequence [7],

$$M = \left\lceil \sum_{i=1}^{N} -\log_2 p(x_i|x_1, \cdots, x_{i-1}) \right\rceil$$

$$= \left\lceil \sum_{i=1}^{N} \log_2 \frac{E(x_1, \cdots, x_{i-1})}{E(x_1, \cdots, x_i)} \right\rceil.$$

Successive terms cancel, so

$$M = \left\lceil \log_2 \frac{E(\Lambda)}{E(x_1,\cdots,x_N)} \right\rceil.$$

As $x_1,\cdots,x_N$ is in the domain of strings which may be encoded, it belongs to Ω and so $E(x_1,\cdots,x_N) = 1$. (This rests on the unambiguous prefix property discussed above.) Therefore $M = \lceil \log_2 E(\Lambda) \rceil = \lceil \log_2 |\Omega| \rceil$, exactly the same as an enumerative encoding.

Faithful probabilities are useful because they provide a link between enumerative codes and purely probabilistic codes (such as Markov models) and enable their coding efficiencies to be compared. Also, faithful probabilities can be computed easily in some cases. When coupled with arithmetic coding, they permit an efficient realization of enumerative codes. Two simple examples are considered below. Unfortunately the complexity of the problem grows dramatically in more realistic situations. (We address this difficulty in Section III.)

Example 1: Order-0 weighted model (Cover [2, ex. 2]). Let Ω be the set of sequences x over the alphabet $A = \{0,1\}$ which have a specified number of 0's and a specified number of 1's. Define c^0: $A^* \to Z^+$ to be the number of 0's in a string, and c^1 similarly. It is shown by Cover [2] that the number of ways of completing a string is the number of ways of arranging the remaining 0's and 1's, giving the binomial expression

$$E(x_1,\cdots,x_i) = \frac{(N-i)!}{c^0(x_{i+1},\cdots,x_N)!\, c^1(x_{i+1},\cdots,x_N)!}.$$

After substitution of this into (1), canceling and rearranging gives the faithful probabilities

$$p(w|x_1,\cdots,x_i) = \frac{c^w(x_{i+1},\cdots,x_N)}{N-i}.$$

That is, the probability that w will be the next symbol is the ratio of the number of times w will occur in the rest of the string divided by the total number of characters in the rest of the string.

Example 2: Order-1 weighted model (Cover [2, sect. 3]). Consider an order-1 weighted model of x over the alphabet $A = \{0,1\}$. This model will have as parameters the four counts $c^{00}(x)$, $c^{01}(x)$, $c^{10}(x)$, and $c^{11}(x)$, where, for example, $c^{00}(x)$ is the number of times the subsequence 00 occurs in x. The set Ω then comprises all values of x which satisfy the given set of counts.

To construct a formula for E, let us first dispense with the special case at the very beginning of the sequence:

$$E(\Lambda) = E(0) + E(1).$$

When computing $E(x_1,\cdots,x_i)$ there will be two cases, depending on whether x_i is 0 or 1, and in each case the expression will depend on counts taken over the completion of the string, for example, $c^{00}(x_{i+1},\cdots,x_N)$ which will be abbreviated by c^{00}. As shown by Cover [2],

$$E(x_1,\cdots,x_{i-1},0) = \frac{(c^{00}+c^{10})!}{c^{00}!c^{10}!} \cdot \frac{(c^{11}+c^{01}-1)!}{c^{11}!(c^{01}-1)!},$$

and

$$E(x_1,\cdots,x_{i-1},1) = \frac{(c^{11}+c^{01})!}{c^{11}!c^{01}!} \cdot \frac{(c^{00}+c^{10}-1)!}{c^{00}!(c^{10}-1)!}.$$

The complexity of these expressions compared with the previous example portends a rapid increase in complexity of enumeration when higher order models are used.

Using this formula for E in (1), it is not difficult to show that the faithful probabilities $p(w|x_1,\cdots,x_i)$ depend only upon w and x_i. There are two cases, one where the next symbol w is equal to x_i and the other where it is not ($x_i = \bar{w}$). This allows the probabilities to be expressed compactly for the general case where $i \geq 1$ as

$$p(w|\cdots,w) = \frac{c^{ww}}{c^{ww}+c^{\bar{w}w}},$$

$$p(\bar{w}|\cdots,w) = \begin{cases} \dfrac{c^{\bar{w}w}}{c^{ww}+c^{\bar{w}w}}, & c^{ww} > 0; \\ 1, & c^{ww} = 0, \end{cases} \tag{2a}$$

where $w = 0$ or 1 and $\bar{w} = 1$ or 0, respectively. Note that these formulas do *not* satisfy the naive interpretation that the probability of w is the number of times w follows the symbol x_i in the rest of the sequence divided by the number of times x_i occurs in the rest of the sequence. Such an interpretation is only appropriate to an order-zero model. In the special case for the first symbol of the sequence,

$$p(w|\Lambda) = \frac{c^{w\bar{w}}(c^{ww}+c^{\bar{w}w})}{c^{w\bar{w}}(c^{ww}+c^{\bar{w}w}) + c^{\bar{w}w}(c^{\bar{w}\bar{w}}+c^{w\bar{w}})}. \tag{2b}$$

In the examples above the expressions for the faithful probabilities involve only the counts $c^w(x_{i+1},\cdots,x_N)$ and $c^{wv}(x_{i+1},\cdots,x_N)$, respectively. It is possible to maintain these counts as simple variables, one of which is decremented each time a symbol of x is encoded (see Table I for an example). For each symbol in x, the complete encoding process involves a small fixed number of arithmetic operations on small intergers for each of the three steps: a) calculating the probability; b) decrementing one counter; and c) the arithmetic encoding itself. This realization of enumerative coding is more efficient than a direct implementation of the formulas given by Cover [2] which involves manipulating numbers which are as long as the output code y.

III. Approximate Enumerative Codes

The previous section showed how to implement enumerative codes efficiently. As enumeration is min–max optimal, it may not be clear why we wish to investigate alternative codes. Unfortunately the faithful probability expressions obtained in the examples become much more complex for higher order models. This section therefore exhibits codes which maintain the computational simplicities of the scheme above but avoid its complexity for higher order models. Two such codes are examined. They have the added advantage of bearing a close relationship to the adaptive codes introduced in Section IV.

One of these approximate codes includes Markov codes, and like them assumes that the probabilities are fixed throughout the encoding. First, however, a novel scheme is examined which is intermediate between faithful probabilities and fixed probabilities. It is theoretically tractable and gives a very simple expression for the probabilities used to generate an arithmetic coding. This scheme is called *approximate enumeration*. We conclude by demonstrating bounds for the differences in coding efficiency between these methods and exact enumerative encoding.

Generalized Notation: To provide a suitably general setting for these two techniques we adopt the terminology used by Rissanen and Langdon [4]. At each step in the encoding of x, the prefix $x_1,\cdots,x_i$ is assigned to one of a finite set of *conditioning classes*. The significance of this is that the probabilities assigned for the next symbol x_{i+1} depend only on the conditioning class of the string seen so far $(x_1,\cdots,x_i)$ and not on the string itself. For example, in an order-1 model the conditioning class for $x_1,\cdots,x_i$ is uniquely determined by x_i.

Let there be K conditioning classes in the set $Z = \{0,1,\cdots,K-1\}$. The conditioning class for a particular string will be determined by a *structure function* f mapping all possible prefixes into the set of conditioning classes; that is, f: $\bar{\Omega} \rightarrow Z$. For example, an order-1 model for Example 2 in Section II might have

$$Z = \{0,1\},$$
$$f(x_1,\cdots,x_i) = x_i,$$
$$f(\Lambda) = 0 \quad \text{(this is arbitrary, } f(\Lambda) = 1 \text{ is equally valid)}.$$

The counting functions $c^0(x)$ and $c^1(x)$ played an important role in the analysis above of an order-0 weighted model, as did $c^{00}(x)$, $c^{01}(x)$, $c^{10}(x)$, and $c^{11}(x)$ in the order-1 case. Using the structure function, these can be generalized to $c^{zw}(x)$, which counts the number of times the symbol w follows a prefix with conditioning class z in the string x, that is the number of times there is a transition from z to w. It will be useful to define $c^z(x)$ as the total number of times some symbol follows a conditioning class z in x: $c^z(x) \equiv \sum_{w\in A} c^{zw}(x)$. (Because no transition follows the last symbol this total may be one less than the number of times z occurs in x.)

Now consider a generalized enumerative coding problem. Assume that both sender and receiver know the following about the string x which will be transmitted:

- its alphabet, A;
- a structure function f, and its associated set of conditioning classes Z;
- the set of counts $c^{zw}(x)$, $x \in Z$, $w \in A$.

The set Ω to be enumerated consists of all strings x over A which satisfy the given set of counts.

Exact Enumeration: The formulas of the last section carry over exactly to this new problem, except for the special cases which arise when computing $E(\Lambda)$. Because Λ is assigned to a conditioning class, the probabilities of x_1 can be computed with respect to this conditioning class. In the example above Λ is assigned the same conditioning class as a string ending in 0, that is, $f(\Lambda) = 0$. If this structure function is applied to Example 2 in the last section, $p(x_1|\Lambda) = p(x_1|0)$ which can be computed using the general formula (2a).

Approximate Enumeration: For a given string x and conditioning class z we can read off the order of occurrence of each of the c^z symbols which follow z. Conversely, if the ordering of symbols for each conditioning class is known then a unique x can be reconstructed. (Start with $z = f(\Lambda)$ and make x_1 the first symbol for z; then continue with $z = f(x_1)$ and its first symbol; and so on until all of $x_1,\cdots,x_N$ is built up.) The product of the number of possible orderings for each conditioning class gives an enumeration of all possible strings x. An example below shows this enumeration is only approximate, because some possible sets of orderings correspond to no valid string x.

The number of ways of ordering the symbols for z is the number of ways of distributing the c^{zw} times each symbol w occurs among the total transitions c^z. This is given by the multinomial $c^z(x)!/\prod_{w\in A} c^{zw}(x)!$. The product over all conditioning classes gives the generalized approximate enumeration

$$E^{\sim}(x_1,\cdots,x_i) = \prod_{z\in Z} \frac{c^z(x_{i+1},\cdots,x_N)!}{\prod_{w\in A} c^{zw}(x_{i+1},\cdots,x_N)!}. \quad (3)$$

The superscript $\sim$ is used to indicate that the enumeration is approximate. The corresponding faithful probabilities can be found from (1) to be

$$p^{\sim}(w|x_1,\cdots,x_i) = \frac{c^{zw}(x_{i+1},\cdots,x_N)}{c^z(x_{i+1},\cdots,x_N)}, \quad (4)$$

where $z = f(x_1,\cdots,x_i)$. This simple probability expression is computationally efficient because it can be implemented using decrementing counters as in Section II.

An example will now be used to show that the enumeration is not exact. Consider an order-1 model which has as parameters the four counts $c^{00}(x) = c^{01}(x) = c^{10}(x) = c^{11}(x) = 2$, two conditioning classes 0 and 1, and where $f(\Lambda) = 0$. One member of Ω is thus 01100110. Consider the ordering 0, 1, 1, 0 for conditioning class 0 and 0, 0, 1, 1 for conditioning class 1. This generates the initial substring 010100 which cannot be completed because all transitions from 0 to 1 have been exhausted and yet two 11 transitions remain. Thus an enumeration based on counting all possible orderings of transitions will always overcount the number of possibilities. It can still be used for coding members of Ω, as explained above, because all members of Ω are included uniquely in the count. There will, however, be some loss of coding efficiency. Bounds on this loss are computed below.

This same example is used in Table I to show the differences between exact and approximate enumeration and a third fixed frequency encoding introduced below. The table shows successive values of the decrementing counters used by all three methods, the probabilities computed at each step, and the final rounded code lengths.

TABLE I
COMPARISON OF THREE ENCODINGS FOR AN ORDER - 1 MODEL

i	1	2	3	4	5	6	7	8	
$c^{00}(x_i \cdots x_N)$	2	1	1	1	1	0	0	0	0
$c^{01}(x_i \cdots x_N)$	2	2	1	1	1	1	0	0	0
$c^{10}(x_i \cdots x_N)$	2	2	2	2	1	1	1	1	0
$c^{11}(x_i \cdots x_N)$	2	2	2	1	1	1	1	0	0
x_i		0	1	1	0	0	1	1	0
$p(x_i \mid x_1 \cdots x_{i-1})$		$\frac{1}{2}$	$\frac{2}{3}$	$\frac{2}{3}$	$\frac{1}{2}$	$\frac{1}{2}$	1	1	1 M = 5 bits
$p^{\sim}(x_i \mid x_1 \cdots x_{i-1})$		$\frac{1}{2}$	$\frac{2}{3}$	$\frac{1}{2}$	$\frac{2}{3}$	$\frac{1}{2}$	1	$\frac{1}{2}$	1 $M^{\sim}$ = 6 bits
$p^{F}(x_i \mid x_1 \cdots x_{i-1})$		$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$ M^{F} = 8 bits
$f(\Lambda) = 0$	$f(\cdots 0) = 0$				$f(\cdots 1) = 1$				

Fixed Frequency (Markov) Model: Markov models are commonly used in practical coding applications. They assume that the frequencies of symbols are constant and can thus be estimated from counts over the entire string. This notion of using fixed frequencies can be generalized to any structure function and gives the the arithmetic coding probabilities

$$p^{F}(w|x_1,\cdots,x_i) = \frac{c^{zw}(x)}{c^{z}(x)},$$

where $z = f(x_1,\cdots,x_i)$ and the superscript F is used to indicate Fixed frequency encoding. It can be seen to be an approximation to exact enumeration by computing an enumeration function E^{F} from these probabilities using (1):

$$E^{F}(x_1,\cdots,x_i) = \prod_{z\in Z} \frac{c^{z}(x)^{c^{z}(x_{i+1},\cdots,x_N)}}{\prod_{w\in A} c^{zw}(x)^{c^{zw}(x_{i+1},\cdots,x_N)}}. \quad (5)$$

Analytic Comparison: We derive bounds on the relative efficiencies of the three codes introduced so far. We will show that exact enumeration is better than approximate enumeration which is better than a fixed frequency enumeration. Bounds are also given on how much worse than exact enumeration the other two codes are. In the limit of long strings x these bounds are proportional to $\log_2 N$.

To show that the coding length of exact enumeration is shorter than that of approximate enumeration it is sufficient to show that $E(\Lambda) \leqslant E^{\sim}(\Lambda)$ (because the code length M for an enumeration E is given by the monotonic nondecreasing function $M = \lceil \log_2 E(\Lambda) \rceil$). This inequality follows directly from the definition of $E^{\sim}$ because it enumerates a superset of Ω.

Similarly, approximate enumeration is better than fixed frequency enumeration because $E^{\sim}(\Lambda) \leqslant E^{F}(\Lambda)$. This can be shown by arguing directly from (3) and (5). The inequality can be decomposed into a sufficient set of inequalities for each conditioning class,

$$\frac{c^{z}!}{\prod_{w\in A} c^{zw}!} \leqslant \frac{(c^{z})^{c^{z}}}{\prod_{w\in A} (c^{zw})^{c^{zw}}}.$$

These individual inequalities can then be proven by decomposing the multinomial into a product of binomials as described by Gallager [10, prob. 5.8b] and Shannon and Weaver [11, p. 49].

The relative coding efficiencies of approximate and fixed frequency enumeration can be compared by computing an upper bound on the expression $\log_2 E^{F}(\Lambda) - \log_2 E^{\sim}(\Lambda)$ which will be within one bit of the actual (integral) difference in code lengths. Applying Stirling's approximation (see Abramowitz and Stegun [12]) to (3) and (5) yields

$$\log_2 E^{F}(\Lambda) - \log_2 E^{\sim}(\Lambda) \leqslant \frac{1}{2}\left[K(q-1)\left(\log_2 2\pi + \frac{1}{6}\right) + \sum_{z\in Z,\, w\in A} \log_2 c^{zw} - \sum_{z\in Z} \log_2 c^{z}\right].$$

Assuming the string x is long ($N \gg Kq$), we get the asymptotic bound

$$\log_2 E^{F}(\Lambda) - \log_2 E^{\sim}(\Lambda) \leqslant \frac{1}{2}K(q-1)\log_2 N + O(Kq\log_2 K).$$

The efficiency of exact and approximate enumeration will be compared by computing an upper bound for the expression $\log_2 E^{\sim}(\Lambda) - \log_2 E(\Lambda)$. This is a more difficult enterprise, the crux of which is the calculation of a lower bound for $E(\Lambda)$. The result on which this lower bound rests is proven in the Appendix in the form of an inequality on the faithful enumerative probabilities $p(w|x_1,\cdots,x_i)$. In order to prove this it is necessary to assume that the structure function f represents a finite state machine (FSM). This is not a great restriction as it includes many models of practical interest. For example, the structure functions for Markov models are always FSM's. The essence of an FSM is that the next state of the machine can be computed from its last state and the current input symbol. In our current context this translates into a requirement that the next conditioning class follows from the previous conditioning class and the next input symbol. This can be stated more formally by requiring that there be some next-state function μ: $Z \times A \to Z$ such that

$$f(uw) = \mu(f(u), w), \qquad uw \in \bar{\Omega},\ w \in A.$$

Making this assumption, the Appendix shows that

$$p(w|x_1,\cdots,x_i) \leqslant \frac{c^{zw}(x_{i+1},\cdots,x_N)}{c^{z}(x_{i+1},\cdots,x_N) - 1}, \qquad z = f(x_1,\cdots,x_i).$$

Applying (1) gives

$$E(\Lambda) \geqslant \prod_{z\in Z} \frac{(c^{z}(x)-1)!}{\prod_{w\in A} c^{zw}(x)!}.$$

This can be combined with (3) and rearranged to give the inequality

$$\log_2 E^{\sim}(\Lambda) - \log_2 E(\Lambda) \leqslant \sum_{z\in Z} \log_2 c^{z}(x).$$

If the input string x is long ($N \gg K$), the maximum for this bound is attained when all the $c^{z}(x)$ are equal to N/K.

TABLE II
ASYMPTOTIC BOUNDS ON OUTPUT LENGTH FOR ENUMERATIVE CODES

$E(\Lambda) \leq E^{\sim}(\Lambda) \leq E^{F}(\Lambda)$
$\log_2 E^{\sim}(\Lambda) - \log_2 E(\Lambda) \leq K \log_2 N$
$\log_2 E^{F}(\Lambda) - \log_2 E(\Lambda) \leq \frac{1}{2} K(q+1) \log_2 N$
$\log_2 E^{F}(\Lambda) - \log_2 E^{\sim}(\Lambda) \leq \frac{1}{2} K(q-1) \log_2 N$

It can then be written as

$$\log_2 E^{\sim}(\Lambda) - \log_2 E(\Lambda) \leq K \log_2 N - K \log_2 K.$$

The two bounds above also imply a bound on the difference in efficiency of fixed frequency and exact enumeration. Table II summarizes the results in their asymptotic form.

IV. Encoding Without Prior Statistics

The rest of this paper considers a rather different coding problem. Now the sender and receiver know the alphabet A, of the string to be transmitted, a structure function f, its associated conditioning classes Z, and an upper bound on the length of the string to be transmitted, $N_{\max}$. This corresponds much more closely to practical coding problems than assuming that complete message statistics are available prior to transmission. It is possible to enumerate all possible strings up to the maximum length to provide a code for this problem. However, this gives no compression as it sends an output code of the same length (approximately $N_{\max}$) irrespective of the input code length. Where most messages are much shorter than the maximum possible an enumerative code will cause expansion rather than compression!

The three codes introduced so far can be extended to cover this new problem more gracefully. First the message parameters are sent in the form of the counts $c^{zw}(x)$, and then the string x is transmitted using an enumerative code as before. This gives codes whose output lengths are linear in N plus terms of order $\log_2 N$, which is intuitively reasonable in the absence of statistics about message lengths. These extended codes are called *parameterized enumeration* (PE) codes.

$$p^{A}(w|x_1,\cdots,x_i) = \begin{cases} \dfrac{c^{zw}(x_1,\cdots,x_i)}{1 + c^{z}(x_1,\cdots,x_i)}, & c^{zw}(x_1,\cdots,x_i) > 0 \\ \dfrac{1}{1 + c^{z}(x_1,\cdots,x_i)} \cdot \dfrac{1}{q - t^{z}(x_1,\cdots,x_i)}, & c^{zw}(x_1,\cdots,x_i) = 0 \end{cases}$$

Denote the code lengths for the enumerative codes of Sections II and III by M, $M^{\sim}$, M^{F} and for the parameterized enumerative codes by T, $T^{\sim}$, T^{F}, respectively. If it takes S bits to transmit the message parameters then $T^{F} = M^{F} + S$, etc. S is the same for all three codes so the relative bounds of Section III generalize immediately. While previously we could say that enumerative coding was min–max optimal, this is no longer true for the new problem. Any justification of the codes in this section must ultimately be in terms of observed performance in actual coding problems. Some results of actual performance on English text and program text are given in the next section.

We now consider for the first time *adaptive* codes. The general idea of an adaptive code is that message parameters are not transmitted explicitly but accumulated as the message is transmitted. They gradually "adapt" toward the true message parameters. If the encoding probabilities are based on these changing parameters both sender and receiver must accumulate the parameters in concert. To see how this might be done consider an initial sequence of y, say $y_1,\cdots,y_j$. After it has been transmitted (using an arithmetic code), the only statistics available to both sender and receiver are $c^{zw}(x_1,\cdots,x_i)$, where $x_1,\cdots,x_i$ is the subsequence that can be decoded from $y_1,\cdots,y_j$. By analogy with (4) we might estimate the *adaptive* probabilities as

$$p^{\text{adaptive}}(w|x_1,\cdots,x_i) = \frac{c^{zw}(x_1,\cdots,x_i)}{c^{z}(x_1,\cdots,x_i)},$$

where $z = f(x_1,\cdots,x_i)$. This is insufficient, however, as some count c^{zw} may be zero and yet the symbol w may still occur. An estimate must therefore be made of the probability that a symbol will occur in a context in which it has not occurred previously. In the absence of any *a priori* knowledge of the statistics, it does not seem possible to provide a sound theoretical basis for doing this. The two adaptive techniques described below are apparently reasonable solutions to this problem.

In order to introduce the adaptive codes, the following notation is needed. $t^{z}(x)$ is the number of distinct symbols which occur following some conditioning class z in the string x, that is, $t^{z}(x) \equiv |\{w\colon c^{zw}(x) > 0\}|$.

Adaptive Code A: The first adaptive code estimates the probabilities for arithmetic coding by allocating one count to the possibility that a previously unseen symbol will occur next. The probability for a previously seen symbol is then $c^{zw}/(1 + c^{z})$ rather than c^{zw}/c^{z}. There are $q - t^{z}$ symbols which have not yet been seen, and they are all assumed to be equally likely. So the probability of one of these is the product $1/(1 + c^{z}) \cdot (q - t^{z})$. This gives the following probabilities for an arithmetic encoding:

$$p^{A}(w|x_1,\cdots,x_i) = \begin{cases} \dfrac{c^{zw}(x_1,\cdots,x_i)}{1 + c^{z}(x_1,\cdots,x_i)}, & c^{zw}(x_1,\cdots,x_i) > 0 \\ \dfrac{1}{1 + c^{z}(x_1,\cdots,x_i)} \cdot \dfrac{1}{q - t^{z}(x_1,\cdots,x_i)}, & c^{zw}(x_1,\cdots,x_i) = 0 \end{cases}$$

where $z = f(x_1,\cdots,x_i)$.

Adaptive Code B: The second adaptive code effectively adds one to all the counts. Each symbol w is treated as if it has been counted $c^{zw} + 1$ times to give a total of $c^{z} + q$ for context z. If w has not been seen before it is allocated a count of 1. The arithmetic coding probabilities are given by the single expression

$$p^{B}(w|x_1,\cdots,x_i) = \frac{c^{zw}(x_1,\cdots,x_i) + 1}{c^{z}(x_1,\cdots,x_i) + q},$$

$$z = f(x_1,\cdots,x_i).$$

TABLE III
COMPARISON OF ADAPTIVE CODES FOR AN ORDER - 1 MODEL

i	1	2	3	4	5	6	7	8	
$c^{00}(x_1 \cdots x_i)$	0	1	1	1	1	2	2	2	2
$c^{01}(x_1 \cdots x_i)$	0	0	1	1	1	1	2	2	2
$c^{10}(x_1 \cdots x_i)$	0	0	0	0	1	1	1	1	2
$c^{11}(x_1 \cdots x_i)$	0	0	0	1	1	1	1	2	2
x_i		0	1	1	0	0	1	1	0
$p^A(x_i \mid x_1 \cdots x_{i-1})$		$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{3}$	$\frac{1}{4}$	$\frac{2}{5}$	$\frac{1}{4}$
$p^B(x_i \mid x_1 \cdots x_{i-1})$		$\frac{1}{2}$	$\frac{1}{3}$	$\frac{1}{2}$	$\frac{1}{3}$	$\frac{1}{2}$	$\frac{2}{5}$	$\frac{1}{2}$	$\frac{2}{5}$

$T^A = 11$ bits
$T^B = 10$ bits

$f(\Lambda) = 0 \quad f(\cdots 0) = 0 \quad f(\cdots 1) = 1$

TABLE IV
CALCULATION OF ADAPTIVE PROBABILITIES

w	1	2	3	4
c^{zw}	1	2	0	0
$p^A(w \mid z)$	$\frac{1}{4}$	$\frac{2}{5}$	$\frac{1}{8}$	$\frac{1}{8}$
$p^B(w \mid z)$	$\frac{1}{7}$	$\frac{2}{7}$	$\frac{1}{7}$	$\frac{1}{7}$

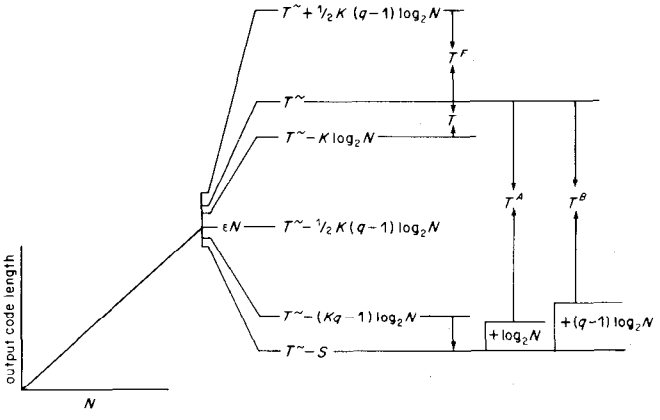

Fig. 1. Relationship between the bounds on coding efficiency.

Table III gives an example of the calculation of these adaptive codes using the same order-1 model and string as in Table I. Table IV gives a more detailed example of the calculation of p^A and p^B at one step in an encoding.

The probability formulas above involve terms of the form $c^{zw}(x_1, \cdots, x_i)$. These can be implemented simply and efficiently as incrementing counters, one of which is incremented each time a symbol is encoded or decoded. This resembles approximate enumeration which uses decrementing counters for the terms $c^{zw}(x_{i+1}, \cdots, x_N)$. Thus the earlier remarks about the computational efficiency of approximate enumeration apply in this case too.

Table V summarizes upper and lower bounds for the output coding lengths of the adaptive and PE codes. The bounds for the PE codes follow directly from the results of the last section and those for adaptive codes are computed in the following subsections. The form of each is $T^{\sim}$ plus or minus terms which in the limit of long strings are proportional to $\log_2 N$. Furthermore, in the limit $T, T^{\sim}, T^F, T^A, T^b$ are proportional to the input string length N plus terms of order $Kq \log_2 N$. Fig. 1 illustrates the relationship between the various bounds.

TABLE V
ASYMPTOTIC BOUNDS ON OUTPUT CODING LENGTH ($N \gg Kq$)[1]

$\frac{1}{2}K(q-1)\log_2 N + T^{\sim} \geq T^F \geq T^{\sim}$

$T^{\sim} \geq T \geq T^{\sim} - K\log_2 N$

$T^{\sim} \geq T^A \geq T^{\sim} - S + \log_2 N$

$T^{\sim} \geq T^B \geq T^{\sim} - S + (q-1)\log_2 N$

$S \geq (Kq-1)\log_2 N$

$M^F \approx \varepsilon N$

[1] *Note:* Terms of at worst order $Kq \log_2 Kq$ have been neglected.

Transmission of Model Parameters: The first step in comparing the adaptive and PE codes is to bound the number of bits required to send the model statistics. This can then be added to the results of the last section to give a total value for the output code length.

For a given model, at most Kq different counts c^{zw} will need to be transmitted. Provided that they are independent of one another, a lower bound is obtained by using an enumerative code to send the counts. To enumerate the set of counts, the number of symbols in x is transmitted using $\lceil \log_2 N_{\max} \rceil$ bits and then the index within the set of possible counts is sent. Given $N = \sum c^{zw}$ which is to be divided among Kq different counts, there are $(N + Kq - 1)!/N!(Kq-1)!$ ways of doing this. So the total number of bits to send the model parameters is

$$S \geq \log_2 \frac{(N + Kq - 1)!}{N!(Kq-1)!} + \log_2 N_{\max}.$$

Using Stirling's formula and rearranging, this expression can be approximated by

$$\begin{aligned} S \geq\ & \left(N + Kq - \tfrac{1}{2}\right)\log_2 (N + Kq - 1) \\ & - \left(N - \tfrac{1}{2}\right)\log_2 N \\ & - \left(Kq + \tfrac{1}{2}\right)\log_2 (Kq - 1) + O(1). \end{aligned} \tag{6}$$

For a long string ($N \gg Kq$), a further approximation is

$$S \geq (Kq - 1)\log_2 N.$$

As the model counts become large, each will occupy approximately $\log_2 N$ bits.

Adaptive Code A: Consider the first of the adaptive codes. When the symbol $w = x_{i+1}$ is encoded it contributes $-\log_2 c^{zw}(x_1, \cdots, x_i)/(1 + c^z(x_1, \cdots, x_i))$ or $\log_2 (1 + c^z(x_1, \cdots, x_i)) \cdot (q - t^z(x_1, \cdots, x_i))$ to the encoded length, depending on whether this is the first time w has occurred or not following the conditioning class z. When summing all the contributions over x for a single conditioning class z, each of the values $1, 2, 3, \cdots, c^z(x)$ will occur once as divisor, and for each symbol w which occurs following class z the values $1, 2, 3, \cdots, c^{zw}(x) - 1$ will occur as dividends and $q - t^z(x_1, \cdots, x_i)$ once as a divisor. Approximate this last term by q, sum over all z, and rearrange using (4) to obtain

$$T^A \leq M^{\sim} + \sum_{w \in A,\, z \in Z} \log_2 c^{zw}(x) + \log_2 q \sum_{z \in Z} t^z(x).$$

It can be shown by considering the worst possible values

for c^{zw} and using (6) that

$$\sum \log_2 c^{zw}(x) \leqslant S + \frac{1}{2}\log_2(Kq-1) + O(1).$$

Combining these two results shows that parameterized approximate enumeration is almost an upper bound for the adaptive code,

$$T^A \leqslant T^{\sim} + \frac{1}{2}\log_2(Kq-1) + \log_2 q \sum_{z\in Z} t^z(x).$$

A similar lower bound can be obtained, $T^A \geqslant T^{\sim} - S + \log_2 N$. This shows that at best adaptive enumeration will require only the code length for approximate enumeration without the model parameters.

Adaptive Code B: A similar argument gives a bound on the differences in coding efficiency of the second adaptive code T^B and approximate enumeration $T^{\sim}$. When calculating p^B for a particular class z and summing over the entire string x, each of $q, q+1, \cdots, q+c^z(x)-1$ will occur once as a divisor, and for each symbol w which occurs following z; $1, 2, \cdots, c^{zw}(x)$ will occur as a dividend. After summing over the contributions of each state, using (3) and rearranging, the total code length is given by

$$T^B = M^{\sim} + \log_2 \prod_{z\in Z} \frac{(q+c^z(x)-1)!}{(q-1)!c^z(x)!}.$$

The second term is bounded above by S and below by $\log_2(N+q-1)!/(q-1)!N!$. So the adaptive code is bounded above by approximate enumeration and below by approximate enumeration less the code length for transmitting parameters plus an additional term

$$T^{\sim} \geqslant T^B \geqslant T^{\sim} - S + \log_2 \frac{(N+q-1)!}{(q-1)!N!}.$$

When x is long ($N \gg q$), the last term above can be approximated by $(q-1)\log_2 N$.

Asymptotic Bounds: If x is long, the bounds on all the coding schemes differ by terms of order $Kq\log_2 N$. Assume as above that $N \gg Kq$ and that the string is generated by a stationary ergodic source, that is, the transition probabilities $p(w|z)$ can be computed using the limits $\lim_{N\to\infty} c^{zw}(x)/c^z(x) = p(w|z)$. Then it is easy to show that M^F is a linear function of N. The coefficient is the entropy of the probabilities $p(w|z)$:

$$M^F \approx \epsilon N,$$

$$\epsilon \equiv \sum_{z\in Z} p(z)\log_2 p(z) - \sum_{w\in A} p(w|z)\log_2 p(w|z),$$

$$p(z) = \lim_{N\to\infty} c^z/N.$$

As all the other coding schemes differ from M^F by terms of order $Kq\log_2 N$, their output coding lengths all have a basically linear form. As is illustrated in Fig. 1, the coding schemes can be envisaged as clustering about the term ϵN and mutually converging on it as N increases.

These asymptotic results are of interest in that they give an intuitive weight to the meaning of the various bounds which have been derived. However, one is often interested in cases where $N \approx Kq$ and where the neglected terms dominate the output code length. Then useful bounds can only be found from the more precise expressions.

TABLE VI
COMPARISON OF COMPRESSION ALGORITHMS ON TWO SAMPLE TEXTS[2]

N	English Text 3614 characters (25.2 Kbit)				C Program Text 3913 characters (27.3 Kbit)			
k	0	1	2	3	0	1	2	3
$T^{\sim}$	16.9	24.8	43.9	65.6	20.8	24.3	44.6	70.1
T^F	17.1	25.7	45.4	66.6	21.6	25.3	45.5	70.7
T^A	16.7	15.5	16.6	17.9	20.8	14.8	12.4	13.3
T^B	16.8	17.8	21.1	22.7	20.8	16.6	20.4	21.8
S	0.8	13.6	38.4	63.6	0.8	14.3	41.1	68.5

[2]*Note:* All values are given in Kbit.

V. SAMPLE ENCODINGS

The algorithms discussed in this paper were applied to two pieces of text using models of various orders. The first was a technical paper containing 3614 ASCII characters. It included some formatting controls as well as a newline character at the end of each line. The second text was that of a program written in *C* [13], including comments and newline characters. This contained 3913 ASCII characters. Because the 7-bit ASCII character set was used, $q = 128$ and the total precompression length of the texts were 25 298 bits and 27 391 bits, respectively. The models used were of order $k = 0$, 1, 2, and 3. The number of counts to be transmitted as statistics was $Kq = q^{k+1}$.

The total of the output code length for the PE and adaptive codes are tabulated in Table VI for the two texts and for each model size. (Exact parameterized enumeration is omitted because it was infeasible to compute it.) Because of the very rapid increase in the number of counts as k increases, the size of the model parameters rapidly outstrips the size of the sequence to be transmitted. As a consequence, for $k \geqslant 2$ the nonadaptive codes actually take more bits than the original sequence. The two adaptive codes are more robust than this. Even for very large values of k they do not exceed the original input sequence length. Indeed it can easily be shown for both code A and B that as the size of the model increases, the output code length will asymptotically approach the input length, for a fixed-size message.

Although a stricter bound has been placed on code B, code A performs better in practice. In fact in all the practical cases tested $T^A < T^B$ although Table III shows a small counter-example to this. Furthermore, the behavior of code A is rather different from the others in that it has a pronounced minimum in total coding length at $k = 2$ and $k = 1$, for the English text and program text, respectively. For both nonadaptive codes, the coding length increases with the model size. Code B reaches a less pronounced minimum at $k = 1$ for the program text.

Apropos of the asymptotic approximations used in the last section it may be noted that the size of the model (as exemplified by S) is only significantly smaller than the code length for $k = 0$. To obtain meaningful bounds, the exact expressions should be used for $k \geqslant 1$.

While the main purpose of this paper has been to provide a foundation for analyzing and comparing adaptive and nonadaptive noiseless codes rather than to invent new and powerful noiseless coding schemes, code A still succeeds in producing respectable compression ratios. For example, Pike [14] reports on the difficulties of reducing English text below 4 bits per character. Despite having no prior knowledge of English text statistics (and hence a coding of 7 bits per character at the start), code A achieved 4.3 bits per character at $k = 1$ for the English text and 3.2 bits per character at $k = 2$ for the C program text. These results are all the more remarkable for being accomplished with fairly short message sequences, which contain insufficient information to estimate high-order statistics accurately.

VI. Summary

Intuitively it might seem that because of incomplete statistics, adaptive codes will be inferior to nonadaptive codes with *a posteriori* knowledge of the statistics. However, we have shown that there are very strong links between adaptive codes and certain enumerative codes and that the adaptive codes are never significantly worse than these enumerations. This instills confidence that adaptive codes may be of practical use. The small examples we have given verify the superiority of adaptive codes. This is further supported by results reported by Cleary and Witten [9]. Using an adaptive code based on code A, a coding of 2.2 bits/character was achieved on mixed case English text. While practical, this code does not seem to be amenable to the analytic techniques introduced above. Langdon and Rissanen [8] also report results using adaptive codes to compress black-and-white video signals which were 20–30 percent more efficient than the best previously published codes. While adaptive codes can consume a great deal of storage, the cost of memory is continually decreasing. They have the advantage that very little *a priori* analysis need be done on the material to be encoded. Also, nonadaptive codes presented with inputs that do not conform to their prior assumptions can perform badly whereas the adaptive codes can be expected to extract whatever regularity is present.

Acknowledgment

The authors would like to thank Radford Neal for helpful discussions and for suggesting code B.

Appendix

Given a finite state model, it follows that if $c^z(x_{i+1},\cdots,x_N) \geq 2$ then

$$\frac{c^{zw}(x_{i+1},\cdots,x_N)-1}{c^z(x_{i+1},\cdots,x_N)-1} \leq p(w|x_1,\cdots,x_i) \leq \frac{c^{zw}(x_{i+1},\cdots,x_N)}{c^z(x_{i+1},\cdots,x_N)-1}$$

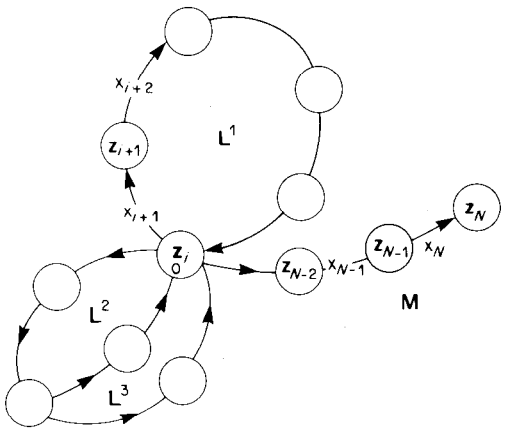

Fig. 2. Decomposition of state diagram into loops.

or if $c^z(x_{i+1},\cdots,x_N) = 1$ then $p(x_{i+1}|x_1,\cdots,x_i) = 1$ and $p(w|x_1,\cdots,x_i) = 0$, for $w \neq x_{i+1}$.

Proof: The special case when $c^z = 1$ can be dealt with immediately, as there is only one possible symbol which can follow z, that is, x_{i+1}. So it can be the only symbol with a nonzero probability.

For the more general case recall that a model is a finite state model if the structure function $f: \bar{\Omega} \to Z$ can be written in terms of a state transition function μ in such a way that the next conditioning class can be computed from the current conditioning class and the next symbol. That is, $\mu: Z \times A \to Z$ and $f(xw) = \mu(f(x), w)$ where $xw \in \bar{\Omega}$ and $w \in A$.

If the model can be represented this way then it can be thought of as a directed graph. The conditioning classes (which are now synonymous with states) form the nodes of the graph and the transitions from each state z to all the states $f(zw)$ which can be reached from it form the edges of the graph.

We will now define a sequence of states z which is the sequence of states traversed as x is input. z is defined by $z_i \equiv f(x_1,\cdots,x_i)$, $1 \leq i \leq N$, and $z_0 \equiv f(\Lambda)$.

The completion of z following z_i is some path through the graph of states and transitions. Because of its importance in the development below we will let z_i equal o (the 'origin' state). Each such path corresponds precisely with one completion $x_{i+1},\cdots,x_N$ of x. So to compute E it is sufficient to count the possible state paths. Any such state path must pass through the state o $c^o(x_{i+1},\cdots,x_N)$ times so that the counts will all be satisfied. (Notice that this includes the first time o is exited.) For our subsequent analysis we will divide the path sequence into a series of loops where each loop finishes at o and does not otherwise visit it. (Fig. 2 illustrates the type of structure we are considering here.) At the end of the sequence there is a tail $\boldsymbol{M}$ which does not include o and which terminates on x_N. We will denote the set of loops into which $z_i,\cdots,z_N$ $(o = z_i)$ is divided by $\boldsymbol{L}$ and the individual loops by $\boldsymbol{L}^h$. The length of the loop $\boldsymbol{L}^h$ will be written $l(h)$. We can more formally define the loops as

$$\boldsymbol{L}^1\boldsymbol{L}^2,\cdots,\boldsymbol{L}^m\boldsymbol{M} \equiv z_{i+1},\cdots,z_N;$$
$$\boldsymbol{L}^h_{l(h)} = z_i = o, \quad 1 \leq h \leq m;$$
$$\boldsymbol{L}^h_j \neq o, \quad 1 \leq h \leq m, 1 \leq j < l(h);$$
$$\boldsymbol{M}_h \neq o, \quad 1 \leq h \leq \text{length of } \boldsymbol{M}.$$

The essence of our argument is that the loops can be rearranged in any order and still give a valid completion for z and x. This is easily seen because each loop starts from the same state (o) so the reassignment of the state loops is paralleled by a corresponding rearrangement of the symbols in x. Note that this argument requires that f represent an FSM. Every completion of x corresponds to some set of loops so by summing over all possible sets of loops and then over all possible rearrangements of loops it is possible to compute $E(x_1,\cdots,x_i,w)$ and $E(x_1,\cdots,x_i)$. To obtain our bound on the ratio of these two quantities we will first obtain a bound on the ratio when consideration is restricted to some *fixed set of loops*. Let ${}^{L}E(x_1,\cdots,x_i)$ be the set of all possible completions to $x_1,\cdots,x_i$ which can be constructed using just the set of loops $\boldsymbol{L}$. Similarly let ${}^{L}E(x_1,\cdots,x_i,w)$ be the set of all possible completions which start with w and which are constructed from the set of loops $\boldsymbol{L}$. We will now compute a bound on the ratio

$$ {}^{L}R(w) \equiv \frac{{}^{L}E(x_1,\cdots,x_i,w)}{{}^{L}E(x_1,\cdots,x_i)}. $$

Let j be the index position at the end of the last loop. That is $z_{i+1},\cdots,z_j = \boldsymbol{L}^1,\cdots,\boldsymbol{L}^m$ and $\boldsymbol{M} = z_{j+1},\cdots,z_N$. We first compute a value for ${}^{L}E(x_1,\cdots,x_iw)$ in terms of the counts over the partial completion from $x_{i+1},\cdots,x_j$ ($c^{ow}(x_{i+1},\cdots,x_j)$ and $c^{o}(x_{i+1},\cdots,x_j)$) rather than all of $x_{i+1},\cdots,x_N$. This value is just the number of ways that the loops in $\boldsymbol{L}$ can be rearranged while ensuring that the first symbol of the first loop equals w, that is, $x_{i+1} = w$. This can be computed as the product of three terms, the first of which is the number of ways those loops that begin with w can be rearranged within themselves. There will be exactly $c^{ow}(x_{i+1},\cdots,x_j)$ such loops. Because most of the counts in what follows are over this range we will just write c^{ow} for this value. This first term will be $c^{ow}!/B$ where B is a term that depends on the number of loops that start with w and how many of them are equal. The second term is the number of ways those loops that do not start with w can be rearranged within themselves. This will be $(c^o - c^{ow})!/C$ where C is a term which depends on the number of loops that do not start with w and how many of them are equal. The third term is the number of ways these loops can be reordered to form the completion while maintaining one of the loops that begins with w as the first loop. This can be done $(c^o - 1)!/(c^{ow} - 1)!(c^o - c^{ow})!$ ways. Collecting these three terms we get

$$ {}^{L}E(x_1,\cdots,x_i,w) = \frac{c^{ow}!}{B} \cdot \frac{(c^o - c^{ow})!}{C} \cdot \frac{(c^o - 1)!}{(c^{ow} - 1)!(c^o - c^{ow})!}. $$

Using similar reasoning we get

$$ {}^{L}E(x_1,\cdots,x_i) = \frac{c^{ow}!}{B} \cdot \frac{(c^o - c^{ow})!}{C} \cdot \frac{c^o!}{c^{ow}!(c^o - c^{ow})!}. $$

Together these give the ratio

$$ {}^{L}R(w) = \frac{c^{ow}(x_{i+1},\cdots,x_j)}{c^{o}(x_{i+1},\cdots,x_j)}. $$

Because $\boldsymbol{M}$ does not visit the state o the counts over the range $i+1,\cdots,j$ are simply related to the counts over $i+1,\cdots,N$. The count for the state o will be one less than for the whole range and the count for the transition ow will be the same over the two ranges except in the particular case when $w = x_{i+1}$; that is

$$ c^{o}(x_{i+1},\cdots,x_j) = c^{o}(x_{i+1},\cdots,x_N) - 1 $$

$$ c^{ox_{i+1}}(x_{i+1},\cdots,x_j) = c^{ox_{i+1}}(x_{i+1},\cdots,x_N) - 1 $$

and

$$ c^{ow}(x_{i+1},\cdots,x_j) = c^{ow}(x_{i+1},\cdots,x_N), \qquad w \neq x_{i+1}. $$

As a consequence we can bound the ratio derived above by

$$ \frac{c^{ow}(x_{i+1},\cdots,x_N) - 1}{c^{o}(x_{i+1},\cdots,x_N) - 1} \leqslant {}^{L}R(w) \leqslant \frac{c^{ow}(x_{i+1},\cdots,x_N)}{c^{o}(x_{i+1},\cdots,x_N) - 1}. \tag{A1} $$

In the special case when $\boldsymbol{M}$ is empty the counts over the range $i+1,\cdots,j$ will be the same as those over $i+1,\cdots,N$ and (A1) will still hold. (If $1 \leqslant a$, $2 \leqslant b$, and $a \leqslant b$ then $(a-1)/(b-1) \leqslant a/b \leqslant a/(b-1)$.)

We have proven the required bounds in the restricted case when some fixed set of loops is considered. E is the sum of ${}^{L}E$ over all possible values of $\boldsymbol{L}$. All the terms in the bound (A1) are positive and independent of $\boldsymbol{L}$ so the following simple result from arithmetic can be used. If given some set of positive numbers $a_1,\cdots,a_r$ and $b_1,\cdots,b_r$ such that $u \leqslant a_s/b_s \leqslant v$ for $1 \leqslant s \leqslant r$ then

$$ u \leqslant \frac{\sum_{s=1}^{r} a_s}{\sum_{s=1}^{r} b_s} \leqslant v. $$

Applied to (A1) this gives the main inequality of the theorem. □

References

[1] H. K. Reghbati, "An overview of data compression techniques," *IEEE Trans. Comput.*, vol. C-14, pp. 71–75, Apr. 1981.

[2] T. M. Cover, "Enumerative source encoding," *IEEE Trans. Inform. Theory*, vol. IT-19, pp. 73–77, Jan. 1973.

[3] C. B. Jones, "An efficient coding system for long source sequences," *IEEE Trans. Inform. Theory*, vol. IT-27, pp. 280–291, May 1981.

[4] J. Rissanen and G. G. Langdon, "Universal modeling and coding," *IEEE Trans. Inform. Theory*, vol. IT-27, pp. 12–23, Jan. 1981.

[5] M. Guazzo, "A general minimum-redundancy source-coding algorithm," *IEEE Trans. Inform. Theory*, vol. IT-26, pp. 15–25, Jan. 1980.

[6] R. Pasco, "Source coding algorithms for fast data compression," Ph.D. dissertation, Stanford University, CA, 1976.

[7] J. J. Rissanen, "Arithmetic codings as number representations," *Acta. Polytech. Scandinavica*, vol. Math. 31, pp. 44–51, 1979.

[8] G. G. Langdon and J. Rissanen, "Compression of black-white images with arithmetic coding," *IEEE Trans. Commun.*, vol. COM-6, pp. 858–867, June 1981.

[9] J. G. Cleary and I. H. Witten, "Data compression using adaptive coding and partial string matching," *IEEE Trans. Commun.*, vol. COM-32, pp. 396–402, April 1984.

[10] R. G. Gallager, *Information Theory and Reliable Communication*. New York: Wiley, 1968.

[11] C. E. Shannon and W. Weaver, *The Mathematical Theory of Communication*. Bloomington, IN: University of Illinois, 1949.

[12] M. O. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions*. Washington, DC: National Bureau of Standards, 1964.

[13] B. W. Kernighan and D. M. Ritchie, *The C Programming Language*. Englewood Cliffs, NJ: Prentice-Hall, 1978.

[14] J. Pike, "Text compression using a 4-bit coding scheme," *Comput. J.*, vol. 24, pp. 324–330, 1981.