On Empirical Cumulant Generating Functions of Code Lengths for Individual Sequences

Neri Merhav

Department of Electrical Engineering
Technion - Israel Institute of Technology
Technion City, Haifa 32000, ISRAEL
E–mail: `merhav@ee.technion.ac.il`

Abstract

We consider the problem of lossless compression of individual sequences using finite–state (FS) machines, from the perspective of the best achievable empirical cumulant generating function (CGF) of the code length, i.e., the normalized logarithm of the empirical average of the exponentiated code length. Since the probabilistic CGF is minimized in terms of the Rényi entropy of the source, one of the motivations of this study is to derive an individual–sequence analogue of the Rényi entropy, in the same way that the FS compressibility is the individual–sequence counterpart of the Shannon entropy. We consider the CGF of the code-length both from the perspective of fixed–to–variable (F-V) length coding and the perspective of variable–to–variable (V-V) length coding, where the latter turns out to yield a better result, that coincides with the FS compressibility. We also extend our results to compression with side information, available at both the encoder and decoder. In this case, the V–V version no longer coincides with the FS compressibility, but results in a different complexity measure.

Index Terms Individual sequences, compressibility, finite–state machines, cumulant generating function, Rényi entropy, Lempel–Ziv algorithm.

1 Introduction

The celebrated paper by Ziv and Lempel [11] was one of the earliest works (if not the first) in the information theory literature that adopted the individual–sequence approach as an alternative to the traditional probabilistic approach (see, e.g., [6, Sections III, IV] and many references therein). In the context of lossless source coding, according to this approach, the system model imposes certain limitations on the resources of the encoder (which is modeled as a finite–state machine) rather than on the statistics of the source sequence to be compressed. One of the most important concepts contributed in [11] was the notion of *finite–state compressibility* of a given infinite source sequence, i.e., the best compression ratio achievable by any finite–state (FS) machine that may compress this sequence. The importance of the FS compressibility is rooted in the fact that it is the individual–sequence analogue of the notion of the entropy rate: while the entropy rate is an asymptotically achievable lower bound on the minimum normalized expected code length in the probabilistic scenario, the FS compressibility is an asymptotically tight lower bound on the minimum normalized empirical expectation of the code length (achievable by FS encoders) in the individual–sequence setting. Moreover, the FS compressiblity of a realization of a finite–alphabet, stationary and ergodic process is equal to the entropy rate almost surely [11, Theorem 4].

Turning for a moment to the traditional probabilistic setting, it is well known that, while the normalized expected code length (or equivalently, the expected compression ratio) has always been the most customary figure of merit, other figure of merits for compression have also been raised in the literature. Most importantly, the cumulant generating function (CGF) of the code length, namely, the normalized logarithm of the exponential moment of the code-length[1] was first proposed by Campbell [2] as a performance criterion for lossless compression, along with a corresponding coding theorem in terms of the Rényi entropy. Campbell's motivation was that the CGF enhances more strongly the contribution of the longest codewords (even if they are weighted by small probabilities), and so, the resulting code optimization is more conservative since the code length fluctuations tend to be reduced. In the realm of stochastic control (and referring to more general problems with any utility function, not necessarily just code length), such a property is called *risk–sensitivity*, and accordingly, the CGF cost function is called a *risk–sensitive* cost. Additional motivations for the

---

[1]A more precise definition will follow in the sequel.

CGF of a cost function in general, include: (i) robustness against uncertainty in the source statistics, (ii) optimization of the *full* distribution of the cost (in some cases), and not just the first moment, and (iii) intimate relationship to the large deviations performance (via the Chernoff bound), which in the context of data compression, has implications on design considerations concerning the buffer overflow probability, see, e.g., [3], [4], [5], [8], [9]. It is also intimately related to the problem of guessing [1]. For a somewhat more elaborate discussion on risk–sensitive cost functions, see, e.g., [7, Introduction] and many references therein.

Combining the contents of the above two paragraphs together, it is now natural to raise the question of what can be said about the individual–sequence counterpart of the CGF of the code–length, namely, the empirical CGF of the code length. In other words, we wish to find an achievable lower bound on the normalized logarithm of the empirical average of an exponential function of the code–length. Such an achievable lower bound would then play a role in a natural definition of an individual–sequence analogue of the Rényi entropy, in parallel to the analogy between the FS compressibility and the Shannon entropy rate. It should be noted, however, that there is an important difference between the Shannon entropy and the Rényi entropy, in this context. While the Shannon entropy rate, defined by the limit of normalized joint entropies, always exists for a stationary ergodic source, there is no established Rényi entropy rate for such a process in general, as the corresponding limit does not always exist. For this reason, there will be no attempt to take this limit, i.e., our results will be stated in terms of a given finite order (or block length).

The above verbal description of the empirical CGF was deliberately given somewhat vaguely, because there is some freedom in the choice of the exact definition, and different definitions turn out to yield different results. Indeed, in the sequel, we will consider a few definitions, and characterize the corresponding achievable lower bounds. In most cases, the achievability will be accomplished by some variant of the Lempel–Ziv (LZ) algorithm [11]. In particular, we will consider the CGF of the code-length both from the perspective of fixed–to–variable (F-V) length coding and the perspective of variable–to–variable (V-V) length coding, where the latter turns out to yield a better result, that coincides with the FS compressibility. We also extend our results to compression with side information, available at both the encoder and decoder. In this case, the V–V version no longer coincides with the FS compressibility, but results in a different complexity measure.

The remaining part of this paper is organized as follows. In Section 2, is the setup is first formulated, and then it is divided into two subsections, the first being devoted to the class of F–V empirical CGFs and the second one – to V–V empirical CGFs. In Section 3, our main results are extended to a situation of coding with side information. Finally, in Section 4, the main findings of this work are summarized.

2 Problem Formulation and Main Results

We begin by reviewing the model of a finite–state encoder of Ziv and Lempel [11]. Let $\boldsymbol{x} = (x_1, x_2, \ldots)$ be a deterministic, infinite source sequence (individual sequence) to be compressed, where each x_i takes values in a finite alphabet $\mathcal{X}$ of size α. An s–state encoder E is defined by a quintuple $(\mathcal{S}, \mathcal{X}, \mathcal{Y}, f, g)$, where $\mathcal{S}$ is a finite set of s states, $\mathcal{X}$ is the finite source alphabet just described, $\mathcal{Y}$ is a finite set of binary words (possibly of different lengths, including the null word for idling), $f : \mathcal{S} \times \mathcal{X} \to \mathcal{Y}$ is the encoder output function, and $g : \mathcal{S} \times \mathcal{X} \to \mathcal{S}$ is the next–state function. When the input sequence $(x_1, x_2, ...)$ is fed sequentially into the encoder E, the latter outputs a sequence of binary words $(y_1, y_2, \ldots)$, $y_i \in \mathcal{Y}$, while going through a sequence of states $(z_1, z_2, \ldots)$, $z_i \in \mathcal{S}$, according to

$$y_i = f(z_i, x_i), \quad z_{i+1} = g(z_i, x_i), \quad i = 1, 2, ... \tag{1}$$

where z_i is the state of encoder E at time instant i. The decoder, on the other hand, receives the sequence $y_1, y_2, \ldots$ and reconstructs the source sequence $x_1, x_2, \ldots$. In the sequel, we will use the conventional shorthand notation x_i^j for the string segment $(x_i, x_{i+1}, \ldots, x_j)$ whenever $j \geq i$. For $i = 1$, we will omit the subscript i and denote $(x_1, x_2, \ldots, x_j)$ by x^j. Similar rules will apply to other sequences, like the state sequence and the encoder output sequence. As in [11], in the sequel, we will use the shorthand notation $f(z_1, x^n)$ and $g(z_1, x^n)$ for the output $y^n = (y_1, \ldots, y_n)$ and the sequence of states $z^n = (z_1, \ldots, z_n)$ that are obtained as the response of E to $x^n = (x_1, \ldots, x_n)$ for a given initial state z_1.

Following the terminology of [11], a FS encoder E is said to be *information lossless* (IL) if for all $z_1 \in \mathcal{S}$ and all $x^n \in \mathcal{X}^n$, the triple $(z_1, f(z_1, x^n), g(z_1, x^n))$ uniquely determines x^n. The length function associated with E is defined as

$$L_E(y^n) = \sum_{i=1}^{n} l(y_i), \tag{2}$$

where $l(y_i)$ is the length of the binary string $y_i \in \mathcal{Y}$, which may include the option of $l(y_i) = 0$ for the null output, which is the case when the encoder is idling, i.e., waiting for additional inputs before producing further compressed output bits.

While the compression ratio $L_E(y^n)/n$ can be viewed as the empirical expectation of the code lengths $\{l(y_i)\}$, in this work, we are focusing on the empirical expectations of several exponential functions of these lengths, viewing them as individual–sequence counterparts of the ordinary probabilistic expectations of these functions. It turns out that there is considerable freedom in the definition of this kind of figure of merit, and the corresponding optimal codes are sensitive to the exact definition.

2.1 Fixed–to–Variable Length CGFs and the Empirical Rényi Entropy

For a given $\lambda > 0$, the simplest objective of this kind is the quantity

$$\frac{1}{\lambda} \log \left[\frac{1}{n} \sum_{t=1}^{n} 2^{\lambda l(y_t)} \right], \tag{3}$$

where here and throughout the sequel, logarithms are defined to the base 2. The problem with this objective function is that many data compression algorithms (with block codes as well as Lempel–Ziv algorithms included) work in "bursts". In other words, most of the time they idle (which means $l(y_t) = 0$) and only in relatively few time instants they actually output chunks of compressed bits. The undesirable property of the objective (3) is that each time instant of such an idling stage contributes a term of $2^{\lambda \cdot 0}$ to the sum $\sum_{t=1}^{n} 2^{\lambda l(y_t)}$, and so, there is overall an additive term, which is almost as large as $n \cdot 2^{\lambda \cdot 0} = n$, even though the code length contributed at these times is zero. One possible remedy to this undesired property is to simply ignore these terms. Another possibility is to define the empirical average of an exponential function of the code length for an ℓ–block, and so, when ℓ is large enough, it is conceivable that at least one $l(y_t)$ within each block is positive, and even if this is not the case, the seemingly superfluous term of $2^{\lambda \cdot 0} = 1$ is added only once in a block, rather than almost each time instant (and so, the relative contribution would be insignificant). Consider then the more general objective function

$$\frac{1}{\lambda \ell} \log_2 \left[\frac{\ell}{n} \sum_{t=0}^{n/\ell - 1} 2^{\lambda L(y_{t\ell+1}^{t\ell+\ell})} \right], \tag{4}$$

5

where it is assumed that ℓ is a positive integer that divides n. We next present a simple result concerning the objective (4).

Theorem 1 *For every IL encoder with s states,*

$$\frac{1}{\lambda\ell} \log_2 \left[\frac{\ell}{n} \sum_{t=0}^{n/\ell-1} 2^{\lambda L(y_{t\ell+1}^{t\ell+\ell})} \right] \geq \hat{H}_\lambda^\ell(x^n) - \frac{\gamma(s,\ell)}{\ell}, \tag{5}$$

where

$$\hat{H}_\lambda^\ell(x^n) = \frac{1+\lambda}{\lambda\ell} \log_2 \left(\sum_{a^\ell \in \mathcal{X}^\ell} [\hat{P}(a^\ell)]^{1/(1+\lambda)} \right), \tag{6}$$

$\hat{P}(a^\ell)$ *being the empirical probability (relative frequency) of $a^\ell \in \mathcal{X}^\ell$ in x^n along its n/ℓ non–overlapping ℓ–blocks, $\{x_{t\ell+1}^{t\ell+\ell}\}$, and*

$$\gamma(s,\ell) = 2\log s + \log\left[1 + \log\left(\frac{s^2 + \alpha^\ell}{s^2}\right)\right]. \tag{7}$$

The first term on the r.h.s. of eq. (5) is the empirical ℓ–th order Rényi entropy associated with x^n, which is the natural individual–sequence counterpart of the ordinary ℓ–th order Rényi entropy of the probabilistic setting. The second term expresses (an estimate of) the extra compression capability allowed by the memory of the FS encoder (captured in its state z_i), which may carry useful information between the successive blocks. When $\ell \gg \log s$, however, this extra compression capability becomes relatively negligible, because the amount of past information memorized by the state is very small compared to the amount of information in each source block of size ℓ. The lower bound of Theorem 1 can be essentially achieved by applying a Shannon code for ℓ blocks, which is matched to the probability distribution that is proportional to $[\hat{P}(a^\ell)]^{1/(1+\lambda)}$, and appending a header of size about $|\mathcal{X}|^\ell \log(n/\ell + 1)$, describing the empirical distribution $\{\hat{P}(a^\ell), \ a^\ell \in \mathcal{X}^\ell\}$ (i.e., the type information). Since this is a logarithmic function of n, this overhead redundancy vanishes for large n. However, there is still a gap here in the sense that the number of states required to implement such an encoder is by far larger than all values of s that keep $\gamma(s,\ell)/\ell$ reasonably small for a given ℓ.

Proof. For a given IL encoder E, let $L'(a^\ell) = \min_{z_1 \in \mathcal{S}} L[f(z_1, a^\ell)]$, and define the probability distribution

$$Q(a^\ell) = \frac{2^{-L'(a^\ell)}}{\sum_{\tilde{a}^\ell} 2^{-L'(\tilde{a}^\ell)}} \tag{8}$$

6

Then,

$$L'(a^\ell) = -\log Q(a^\ell) - \log\left[\sum_{\tilde{a}^\ell} 2^{-L'(\tilde{a}^\ell)}\right] \geq -\log Q(a^\ell) - \gamma(s,\ell) \tag{9}$$

where the second inequality is supported by Lemma 2 of [11] (the generalized Kraft inequality). Thus,

$$
\begin{aligned}
\frac{\ell}{n}\sum_{t=0}^{n/\ell-1}\exp_2\{\lambda L(y_{t\ell+1}^{t\ell+\ell})\} &= \frac{\ell}{n}\sum_{t=0}^{n/\ell-1}\exp_2\{\lambda L[f(z_{t\ell+1},x_{t\ell+1}^{t\ell+\ell})]\} \\
&\geq \frac{\ell}{n}\sum_{t=0}^{n/\ell-1}\exp_2\{\lambda L'(x_{t\ell+1}^{t\ell+\ell})\} \\
&\geq \frac{\ell}{n}\sum_{t=0}^{n/\ell-1}\exp_2\{\lambda[-\log Q(x_{t\ell+1}^{t\ell+\ell}) - \gamma(s,\ell)]\} \\
&= 2^{-\lambda\gamma(s,\ell)}\sum_{a^\ell}\frac{\hat{P}(a^\ell)}{Q^\lambda(a^\ell)} \\
&\geq 2^{-\lambda\gamma(s,\ell)}\left(\sum_{a^\ell}[\hat{P}(a^\ell)]^{1/(1+\lambda)}\right)^{1+\lambda} \\
&\geq \exp_2\{\lambda[\ell\hat{H}_\lambda^\ell(x^n) - \gamma(s,\ell)]\}, \tag{10}
\end{aligned}
$$

where the second to the last inequality is obtained by minimizing the expression $\sum_{a^\ell}\hat{P}(a^\ell)/Q^\lambda(a^\ell)$ w.r.t. the probability distribution Q. Finally, the desired result is obtained by taking the base 2 logarithm of both sides and the normalizing by $\lambda\ell$. This completes the proof of Theorem 1. $\square$

For very large ℓ, there is another (conceptually) simple lower bound that is essentially attained by applying the LZ78 algorithm to each ℓ–block separately, namely, restarting the LZ dictionary at every time instant t which is an integer multiple of ℓ. By Theorem 1 of [11], we know that for any s–state IL encoder, $L(y_{t\ell+1}^{t\ell+\ell})$ is lower bounded by $(c_t + s^2)\log\frac{c_t+s^2}{4s^2}$, where c_t is the maximum number of distinct phrases in $x_{t\ell+1}^{t\ell+\ell}$, and so,

$$
\begin{aligned}
&\frac{1}{\lambda\ell}\log\left[\frac{\ell}{n}\sum_{t=0}^{n\ell-1}\exp_2\{\lambda L(y_{t\ell+1}^{t\ell+\ell})\}\right] \\
&\geq \frac{1}{\lambda\ell}\log\left[\frac{\ell}{n}\sum_{t=0}^{n/\ell-1}\exp_2\left\{\lambda(c_t + s^2)\log[(c_t+s^2)/4s^2]\right\}\right]. \tag{11}
\end{aligned}
$$

The second line can be thought of as an alternative definition of the Rényi counterpart of the compressibility of individual sequences.

2.2 Variable–to–Variable Length CGFs and the LZ Complexity

The problem with the objective function (4) is that for large ℓ and large λ, the performance becomes extremely sensitive to fluctuations in the code lengths, $L(y_{t\ell+1}^{(t+1)\ell})$. Clearly, for a given average of $\{L(y_{t\ell+1}^{(t+1)\ell})\}$, eq. (4) is minimized when all lengths are equal to this average (as can easily be understood from Jensen's inequality), namely, when the fluctuations are completely eliminated. This observation motivates us to expand the scope and redefine our objective in the spirit of variable–to–variable length coding which allows much more freedom in the quest for reducing the length fluctuations.

Specifically, rather than the above segmentation of the source string x^n into fixed–length blocks of size ℓ, consider a sequence–dependent segmentation according to a set (or dictionary) of c distinct variable–length strings, which all have (at least approximately) the same empirical probability, in other words, the empirical distribution of this set of strings is uniform, or nearly uniform. In such a case, it would make sense that, at least in the absence of constraints on the encoder structure, the code lengths for those strings would be all the same (or nearly so), and then the length fluctuations would be eliminated altogether. For the class of FS encoders considered here, we may not be able to guarantee uniform lengths always, but this can certainly serve at least as a guideline for good code design.

A natural way to accomplish such a segmentation with a uniform empirical distribution, is by parsing the sequence into c *distinct* phrases, in the spirit of the parsings described in [11]. In this case, every such phrase (or string) appears exactly once, and so, its empirical probability is $1/c$. Accordingly, for a given x^n and a given parsing the sequence into c distinct[2] phrases, $x_1^{n_1}, x_{n_1+1}^{n_2}, \ldots, x_{n_{c-1}+1}^n$, we define

$$\rho_E^\lambda(x^n) = \frac{c}{n\lambda} \log \left[\frac{1}{c} \sum_{i=1}^c 2^{\lambda L(y_{n_{i-1}+1}^{n_i})} \right], \quad n_0 \equiv 0, \quad n_c \equiv n, \tag{12}$$

where the factor c/n outside the logarithm is meant to normalize the empirical CGF by the average phrase length, n/c, in analogy to the factor of $1/\ell$ outside the logarithm in eq. (4).

Informally speaking, had the dictionary of the various phrases been known in advance to both encoder and decoder, then ideally (i.e., ignoring the finite–state structure of the encoder), the compressed form of each one of these phrases would be of length $L(y_{n_{i-1}+1}^{n_i}) = \log c$, and hence, intuitively,

---

[2]With the possible exception of the last phrase, which may be incomplete.

one would expect that essentially, $\rho_E^\lambda(x^n)$ cannot be smaller than

$$\frac{c}{n\lambda} \log\left[\frac{1}{c}\sum_{i=1}^{c} 2^{\lambda \log c}\right] = \frac{c \log c}{n}, \tag{13}$$

which is also the main term of the lower bound on the ordinary compressibility (see [11]). In other words, it seems plausible that $\sum_{i=1}^{c} \exp_2\{\lambda L(y_{n_{i-1}+1}^{n_i})\}$ (which lacks the normalization by c) should be lower bounded by an expression whose exponential order is as large as $2^{(\lambda+1)\log c}$. The next theorem supports this intuition more formally.

Theorem 2 *Given an arbitrary IL encoder E with no more than s states, and given a source sequence x^n with c distinct phrases,*

$$\sum_{i=1}^{c} \exp_2\{\lambda L(y_{n_{i-1}+1}^{n_i})\} \geq \frac{s^2\left[\exp_2\left\{(\lambda+1)\log\left(\frac{c+s^2}{2s^2}\right)\right\}-1\right]}{2^{\lambda+1}-1}. \tag{14}$$

Proof. Given x^n and its parsing into c different phrases, let c_j denote the number of phrases for which the total compressed bit string is of length j, that is, $L(y_{n_{i-1}+1}^{n_i}) = \sum_{t=n_{i-1}+1}^{n_i} l(y_t) = j$. As argued in [11], the IL property of the encoder implies that $c_j \leq s^2 2^j$ for all j, because the initial state, the final state, and the compressed sequence in between uniquely determine the source string. As is also argued in [11], in order to derive a lower bound, one may assume ideal packing of minimal lengths and thereby overestimate c_j as $s^2 2^j$ for $j = 0, 1, \ldots, k$, where k is the largest integer such that $c \geq \sum_{j=0}^{k} s^2 2^j = s^2(2^{k+1}-1)$, which means that $c < s^2(2^{k+2}-1)$. Thus,

$$\sum_{i=1}^{c} 2^{\lambda L(y_{n_{i-1}+1}^{n_i})} \geq s^2 \sum_{j=0}^{k} 2^{\lambda j} \cdot 2^j$$

$$= \frac{s^2[2^{(k+1)(\lambda+1)}-1]}{2^{\lambda+1}-1}. \tag{15}$$

But from the above definition of k, we have

$$k \geq \log\left(\frac{c+s^2}{s^2}\right) - 2 = \log\left(\frac{c+s^2}{2s^2}\right) - 1, \tag{16}$$

and so,

$$\sum_{i=1}^{c} 2^{\lambda L(y_{n_{i-1}+1}^{n_i})} \geq \frac{s^2\left[\exp_2\left\{(\lambda+1)\log\left(\frac{c+s^2}{2s^2}\right)\right\}-1\right]}{2^{\lambda+1}-1}, \tag{17}$$

which completes the proof of Theorem 2. $\square$

9

An alternative lower bound can be obtained using the same technique as in the lower bound in Subsection 2.1, where instead of averaging w.r.t. the empirical distribution of non–overlapping ℓ– blocks, $\{\hat{P}(x^\ell)\}$, as was done in Subsection 2.1, here we have the uniform empirical distribution $\hat{P}(w) = 1/c$, where $\{w\}$ are the c distinct phrases. In this case, Lemma 2 of [11] (the generalized Kraft inequality) applies too, but with α^ℓ being replaced by c in the definition of $\gamma(s, \ell)$, i.e., here the logarithm of the Kraft sum, $\log\left[\sum_w 2^{-L'(w)}\right]$, is upper bounded by $\gamma(s, \log_\alpha c)$. The resulting alternative to the lower bound of Theorem 2 would then be

$$\sum_{i=1}^{c} \exp_2\{\lambda L(y_{n_{i-1}+1}^{n_i})\} \geq \exp_2\{(\lambda + 1)\log c - \lambda\gamma(s, \log_\alpha c)\}. \tag{18}$$

Here too, the leading term at the exponent of the lower bound is $(\lambda + 1)\log c$. None of the two lower bounds dominates the other, in general. The answer to the question which one is tighter depends on the parameters of the problem.

A compatible upper bound is now established for the case where the c phrases are obtained by the incremental parsing procedure of [11], according to which x^n is phrased sequentially, where each new phrase is the shortest string not encountered before as a phrase.

Theorem 3 *Let x^n be given and let c denote the number of phrases resulting from the incremental parsing procedure. Let $L_{LZ}(x_{n_{i-1}+1}^{n_i})$ denote the total length associated with the compression of the i–th phrase according to the LZ78 algorithm [11]. Then,*

$$\sum_{i=1}^{c} \exp_2\{\lambda L_{LZ}(x_{n_{i-1}+1}^{n_i})\} \leq (2\alpha)^\lambda 2^{(\lambda+1)\log c}. \tag{19}$$

The theorem tells that the LZ78 algorithm essentially achieves the lower bound of Theorem 2 (for this choice of c) in the sense that the exponential order of the upper bound (as an exponential function of $\log c$) is the same as that of the lower bound, as they both behave like $2^{(\lambda+1)\log c}$ in their leading term, and uniformly for every $\lambda > 0$. Accordingly, the above–mentioned lower bound on $\rho_E^\lambda(x^n)$, which is about $\frac{c\log c}{n}$, is asymptotically achieved in Theorem 3. It is interesting to observe that although the LZ78 algorithm behaves like a variable–to–variable length code (as it maps variable– length source phrases into variable–length compressed bit strings), it achieves essentially the same performance as that of the ideal variable–to–fixed length code described before, which is, as said, free of the undesirable length fluctuations. Moreover, unlike that ideal variable–to–fixed length code,

10

which is aware of the dictionary of phrases in advance, the LZ78 algorithm achieves this performance without knowing this dictionary ahead of time, and independendtly of λ.

Proof. We refer the reader to the proof of Theorem 2 in [11]. Let $x^{n_1}, x^{n_2}_{n_1+1}, x^{n_3}_{n_2+1}, \ldots, x^n_{n_c+1}$ denote the phrases that result from the incremental parsing procedure. As described in the constructive proof of [11, Theorem 2] (which describes the LZ78 algorithm), the i–th phrase $x^{n_i}_{n_{i-1}+1}$ is encoded by $L_{\mathrm{LZ}}(x^{n_i}_{n_{i-1}+1}) = \lceil \log(\alpha i) \rceil$ bits. Thus,

$$
\begin{aligned}
\sum_{i=1}^{c} \exp_2\{\lambda L_{\mathrm{LZ}}(x^{n_i}_{n_{i-1}+1})\} &= \sum_{i=1}^{c} \exp_2\{\lambda \lceil \log(\alpha i) \rceil\} \\
&\leq \sum_{i=1}^{c} \exp_2\{\lambda [\log(\alpha i) + 1]\} \\
&\leq c \cdot \exp_2\{\lambda [\log(\alpha c) + 1]\} = (2\alpha)^{\lambda} \cdot 2^{(\lambda+1)\log c},
\end{aligned}
\tag{20}
$$

completing the proof of Theorem 3. $\square$

3 Extension to Coding with Side Information

We now extend our main results to the case where side information is available to both the encoder and decoder. We begin by re-formulating the FS encoder model so as to allow access to side information.

An s-state encoder E with side information is defined by a set of six objects, $(\mathcal{S}, \mathcal{X}, \mathcal{Y}, \mathcal{U}, f, g)$, where $\mathcal{S}$, $\mathcal{X}$ and $\mathcal{Y}$ are as before, $\mathcal{U}$ is a finite alphabet of side information, $f : \mathcal{S} \times \mathcal{X} \times \mathcal{U} \to \mathcal{Y}$ is the encoder output function, and $g : \mathcal{S} \times \mathcal{X} \times \mathcal{U} \to \mathcal{S}$ is the next–state function. When an input sequence $(x_1, x_2, \ldots)$ and a side information sequence $(u_1, u_2, \ldots)$ are fed together, sequentially into E, the encoder outputs a sequence of binary words $(y_1, y_2, \ldots)$, while going through a sequence of states $(z_1, z_2, \ldots)$, $z_i \in \mathcal{S}$, according to

$$
y_i = f(z_i, x_i, u_i), \quad z_{i+1} = g(z_i, x_i, u_i), \quad i = 1, 2, \ldots
\tag{21}
$$

where z_i is the state of E at time instant i. The decoder receives the pair sequence $(y_1, u_1), (y_2, u_2), \ldots$ and reconstructs the source sequence $(x_1, x_2, \ldots)$.

A finite–state encoder E with side information is said to be *information lossless* (IL) if for every $z_1 \in \mathcal{S}$ and all $(x^n, u^n) \in \mathcal{X}^n \times \mathcal{U}^n$, $n \geq 1$, the quadruple (z_1, z_{n+1}, y^n, u^n) uniquely determines x^n,

where z_{n+1} and $y^n = (y_1, \ldots, y_n)$ are obtained by iterating eq. (21) with z_1, x^n, and u^n as inputs. As before, the length function associated with E is defined as $L_E(y^n) = \sum_{i=1}^n l(y_i)$.

As for the fixed–to–variable CGF, one can easily extend the derivation in Subsection 2.1 as follows.

$$
\frac{\ell}{n} \sum_{t=0}^{n/\ell-1} \exp_2\{\lambda L(y_{t\ell+1}^{t\ell+\ell})\} \geq \sum_{u^\ell} \hat{P}(u^\ell) \cdot 2^{\lambda[\hat{H}_\lambda^\ell(x^n|u^\ell)-\gamma(s,\ell)]}
$$

$$
\triangleq 2^{-\lambda\gamma(s,\ell)} \sum_{u^\ell} \hat{P}(u^\ell) \left\{ \sum_{x^\ell} [\hat{P}(x^\ell|u^\ell)]^{1/(1+\lambda)} \right\}^{1+\lambda}
$$

$$
= 2^{-\lambda\gamma(s,\ell)} \sum_{u^\ell} \left\{ \sum_{x^\ell} [\hat{P}(x^\ell, u^\ell)]^{1/(1+\lambda)} \right\}^{1+\lambda}, \tag{22}
$$

whose main factor is related to the empirical conditional Rényi entropy of order ℓ.

For the variable–to–variable CGF, following [10], consider a certain parsing of the sequence of pairs $(x_1, u_1), (x_2, u_2), \ldots, (x_n, u_n)$, into $c \equiv c(x^n, u^n)$ distinct phrases. Let $c(u^n)$ be the number of distinct phrases of u^n and let $c_k(x^n|u^n)$ be the number of distinct phrases of x^n parsed jointly with the k–th distinct phrase $u(k)$ of u^n, $1 \leq k \leq c(u^n)$.[3] The idea is that it is now the empirical *conditional* distribution of an x–phrase given a u–phrase that is uniform and is given by $1/c_k(x^n|u^n)$ for all $c_k(x^n|u^n)$ x-phrases pertaining to $u(k)$. For example,[4] if

$$
x^6 = 0 \mid 1 \mid 0\ 0 \mid 0\ 1 \mid
$$

$$
u^6 = 0 \mid 1 \mid 0\ 1 \mid 0\ 1 \mid
$$

then $c(x^6, u^6) = 4$, $c(u^6) = 3$, $u(1) = \text{`0'}$, $u(2) = \text{`1'}$, $u(3) = \text{`01'}$, $c_1(x^6|u^6) = c_2(x^6|u^6) = 1$, and $c_3(x^6|u^6) = 2$.

Let us now define, similarly as before:

$$
\rho_E^\lambda(x^n|u^n) = \frac{c}{n\lambda} \log_2 \left[\frac{1}{c} \sum_{i=1}^c \exp_2\{\lambda L(y_{n_{i-1}+1}^{n_i})\} \right], \quad n_0 \equiv 0, \ n_c \equiv n. \tag{23}
$$

As for a lower bound,

$$
\sum_{i=1}^c \exp_2\{\lambda L(y_{n_{i-1}+1}^{n_i})\} = \sum_{k=1}^{c(u^n)} \sum_{i=1}^{c_k(x^n|u^n)} \exp_2\{\lambda L(y_{n_{i-1}+1}^{n_i})\}
$$

---

[3] Equivalently, $c_k(x^n|u^n)$ is the number of times $u(k)$ appears as a parsed phrase of u^n.
[4] The same example appears in [10].

$$\geq \quad \frac{s^2}{2^{\lambda+1}-1} \cdot \sum_{k=1}^{c(u^n)} \left(\exp_2 \left\{ (\lambda+1) \log \left[\frac{c_k(x^n|u^n)+s^2}{2s^2} \right] \right\} - 1 \right), \quad (24)$$

which is, to the leading term (with respect to $\rho_E^\lambda(x^n|u^n)$), equivalent to

$$\sum_{k=1}^{c(u^n)} \exp_2 \{ (\lambda+1) \log c_k(x^n|u^n) \}. \quad (25)$$

An analogue of the alternative lower bound (18) can also be derived in the same way:

$$\sum_{i=1}^{c} \exp_2 \{ \lambda L(y_{n_{i-1}+1}^{n_i}) \} \quad = \quad \sum_{k=1}^{c(u^n)} \sum_{i=1}^{c_k(x^n|u^n)} \exp_2 \{ \lambda L(y_{n_{i-1}+1}^{n_i}) \}$$

$$\geq \quad \sum_{k=1}^{c(u^n)} 2^{(\lambda+1)\log c_k(x^n|u^n) - \lambda \gamma(s, \log_\alpha c_k(x^n|u^n))]}, \quad (26)$$

which is again, of the same asymptotic order.

For the upper bound, consider the joint incremental parsing of (x^n, u^n). For every $u(k)$, $k = 1, 2, \ldots, c(u^n)$, apply the LZ algorithm separately, so that as before, the inner sum would contribute

$$\sum_{i=1}^{c_k(x^n|u^n)} \exp_2 \{ \lambda L(y_{n_{i-1}+1}^{n_i}) \} \leq (2\alpha)^\lambda 2^{(\lambda+1)\log c_k(x^n|u^n)}. \quad (27)$$

and so, overall, we get an upper bound of

$$(2\alpha)^\lambda \sum_{k=1}^{c(u^n)} \exp_2 \{ (\lambda+1) \log c_k(x^n|u^n) \}, \quad (28)$$

which is asymptotically equivalent to both lower bounds in terms of the achievability of $\rho_E^\lambda(x^n|u^n)$.

In this context, there is an interesting difference, that we observe, between the case without side information, that was handled in Subsection 2.2, and the case with side information considered here. While in the absence if side information, the empirical CGF agreed with the ordinary LZ compressibility, $\frac{c \log c}{n}$ (see eq. (13)), here there is a difference between the empirical CGF, which is roughly

$$\frac{c}{n\lambda} \log_2 \left[\frac{1}{c} \sum_{k=1}^{c(u^n)} \exp_2 \{ (\lambda+1) \log c_k(x^n|u^n) \} \right], \quad (29)$$

and the ordinary LZ compressibility in the presence of side information (see [10]), which is about $\frac{1}{n} \sum_{k=1}^{c(u^n)} c_k(x^n|u^n) \log c_k(x^n|u^n)$, the natural individual–sequence analogue of the conditional entropy. Of course, the latter can easily be recovered from the former by taking the limit $\lambda \to 0$.

4 Summary and Conclusion

In this work, we have made an attempt to develop complexity measures for individual sequences that are analogous to the Rényi entropy of the probabilistic case in the same spirit that the finite–state complexity is analogous to the entropy rate of a stationary ergodic process. We have examined both F–V and V–V definitions of the code–length CGF and obtained different measures. In the F–V case, the main term was the Rényi entropy derived from empirical distribution of non–overlapping blocks of the given sequence, and an alternative measure was given by the empirical CGF of $\{c_t \log c_t\}$, where c_t was defined as the number of distinct phrases in the t–block. In the V–V version of the empirical CGF, the result actually coincides with the ordinary complexity measure, $\frac{c \log c}{n}$. These findings were finally extended to the setting of coding with side infotrmation at both encoder and decoder, but in this case, there is a difference between the empirical CGF and the ordinary conditional FS complexity.

References

[1] E. Arikan, "An inequality on guessing and its application to sequential decoding," *IEEE Trans. Inform. Theory,* vol. IT–42, no. 1, pp. 99–105, January 1996.

[2] L. L. Campbell, "A coding theorem and Rényi's entropy," *Information and Control*, vol. 8, pp. 423–429, 1965.

[3] P. A. Humblet, "Generalization of Huffman coding to minimize the probability of buffer overflow," *IEEE Transactions on Information Theory*, vol. IT–27, no. 2, pp. 230–232, March 1981.

[4] F. Jelinek, "Buffer overflow in variable length coding of fixed rate sources," *IEEE Transactions on Information Theory*, vol. IT–14, no. 3, pp. 490–501, May 1968.

[5] N. Merhav, "Universal coding with minimum probability of code word length overflow," *IEEE Trans. Inform. Theory*, vol. 37, no. 3, pp. 556–563, May 1991.

[6] N. Merhav and M. Feder, "Universal prediction," *IEEE Trans. Inform. Theory*, vol. 44, no. 6, pp. 2124–2147, October 1998.

[7] N. Merhav, "On optimum strategies for minimizing exponential moments of a loss function," *Communications in Information and Systems*, vol. 11, no. 4, pp. 343–368, 2011.

[8] O. Uchida and T. S. Han, "The optimal overflow and underflow probabilities with variable–length coding for the general source," preprint 1999.

[9] A. D. Wyner, "On the probability of buffer overflow under an arbitrary bounded input-output distribution," *SIAM Journal on Applied Mathematics*, vol. 27, no. 4, pp. 544–570, December 1974.

[10] J. Ziv, "Universal decoding for finite-state channels," *IEEE Trans. Inform. Theory*, vol. IT–31, no. 4, pp. 453–460, July 1985.

[11] J. Ziv and A. Lempel, "Compression of individual sequences via variable-rate coding," *IEEE Trans. Inform. Theory*, vol. IT–24, no. 5, pp. 530–536, September 1978.