

Grammar-Based Coding: New Perspectives¹

En-hui Yang², Da-ke He², and John C. Kieffer³

Abstract — Grammar-based coding is investigated from three new perspectives. First, we revisit the performance analysis of grammar-based codes by proposing context-based run-length encoding algorithms as new performance benchmarks. A redundancy result stronger than all previous corresponding results is established. We then extend the analysis of grammar-based codes to sources with countably infinite alphabets. Let Λ denote an arbitrary class of stationary, ergodic sources with a countably infinite alphabet. It is shown that grammar-based codes can be modified so that they are universal with respect to any Λ for which there exists a universal code. Moreover, upper bounds on the worst-case redundancies of grammar-based codes among large sets of length- n individual sequences from a countably infinite alphabet are established. Finally, we propose a new theoretic framework for compression in which grammars rather than sequential stochastic processes are used as source generating models, and point out some open problems in the framework.

I. INTRODUCTION

Grammar-based codes [1] stand for a new class of universal lossless data compression algorithms. To compress a sequence $x = x_1 \cdots x_n$, a grammar-based code first transforms x into a context-free grammar G , from which x can be fully reconstructed as the only sequence in the language generated by G , and then compresses the context-free grammar G . For a detailed description of grammar-based codes and context-free grammars, the reader is referred to [1], [2].

The class of grammar-based codes is very broad; it includes block codes, Lempel-Ziv types of codes [3], and many other new universal lossless compression algorithms as well. For instance, within the design framework of grammar-based codes, several new lossless data compression algorithms, such as the Yang-Kieffer algorithms [2] and the multilevel pattern matching(MPM) algorithm [4], were proposed. The concept of grammar-based coding was further extended in [5], where context-dependent grammar-based codes were proposed.

In this paper, we investigate grammar-based coding from new perspectives. We shall first take a close look at the per-

formance analysis of grammar-based codes, and show how we can further the analysis, and establish new coding theorems. We then turn our attention back to the fundamental concept of grammar-based coding, and introduce the concept of using grammars rather than sequential stochastic processes as source models. This new concept enables us to propose a new theoretic framework for compression. A series of interesting open problems along this line will be discussed.

The compression performance of grammar-based codes has been analyzed so far by comparing their compression performance with that of the best arithmetic coding algorithms that have a fixed finite number of contexts [1], [2]. Specifically, let x represent an arbitrary individual sequence of length $n > 1$ from a finite alphabet. It was proved in [1], [2] that the difference in compression rate on x between the worst grammar-based code¹ and the best arithmetic coding algorithm with k contexts is at most $d \log \log n / \log n$, uniformly over x , where d is a constant depending only on k and the alphabet size.²

The above worst-case redundancy result characterizes the compression performance of grammar-based codes for individual sequences from a finite alphabet. However, such a characterization becomes less meaningful if arithmetic coding algorithms with k contexts fail to provide good performance benchmarks. Indeed, it can be shown that there exist interesting stationary sources with a finite alphabet for which no arithmetic coding algorithm with k contexts can achieve the sources' Shannon entropy rates [6], no matter how large k is. For sequences emitted from these sources, we need to find new performance benchmarks to analyze the compression performance of grammar-based codes. Evidently, from this perspective, one can derive many (redundancy) results for grammar-based codes by selecting different performance benchmarks. In Section II, we discuss some initial results in this direction. Details of these results can be found in [6].

In addition to the worst-case redundancy result, another important result about the compression performance of grammar-based codes is that they are universal for the class of all stationary, ergodic sources with a finite alphabet in the sense that they can achieve asymptotically the entropy rate of each and every stationary, ergodic source with a finite alphabet [1], [2].

Both the worst-case redundancy result and the universality result above share the assumption that the alphabet is finite. However, in many applications like lossless audio compression and high-quality image compression, where data sequences to be compressed are often from large alphabets with cardinalities even comparable to the sequence lengths, the assumption of a finite alphabet is hardly justified; instead, one may simply assume that data sequences come from a countably infinite alphabet. From this perspective, one wants to investigate the

¹This work was supported in part by the Natural Sciences and Engineering Research Council of Canada under Grants RGPIN203035-98 and RGPIN203035-02 and under Collaborative Research and Development Grant, by the Premier's Research Excellence Award, by the Communications and Information Technology Ontario, by the Canada Foundation for Innovation, by the Ontario Distinguished Research Award, and by the Canada Research Chairs Program.

²The authors are with the Department of Electrical and Computer Engineering, University of Waterloo, 200 University Avenue West, Waterloo, Ontario N2L 3G1, Canada. E-mails: ehyang@uwaterloo.ca, dhe@multicom.uwaterloo.ca.

³The author is with the Department of Electrical and Computer Engineering, University of Minnesota, 200 Union St. SE, Minneapolis, MN 55455, USA. E-mail: kieffer@ece.umn.edu.

¹Unless otherwise specified, throughout this paper we focus our discussion on grammar-based codes that transform each data sequence into an irreducible context-free grammar, and will simply refer to them as grammar-based codes.

²log stands for the logarithm to base 2 throughout this paper.

compression performance of grammar-based codes for sources with a countably infinite alphabet. In Section III, we will discuss the universality and redundancies of grammar-based codes in the countably infinite alphabet case. The results in Section III are available with more details in [7].

We now turn our attention to the basic concept of grammar-based coding itself. As described at the beginning of this section, to compress a sequence x a grammar-based code always transforms x into a context-free grammar G , whose language $L(G)$ consists uniquely of the sequence x , i.e.

$$L(G) = \{x\}. \quad (1)$$

The constraint (1) limits us to use only a subclass of the class of all context-free grammars (see [1] for details). To involve more context-free grammars into the design of lossless data compression algorithms, it is logical to investigate if the constraint (1) could be relaxed, for example, to

$$x \in L(G). \quad (2)$$

This motivates us to look at grammar-based coding from a completely new perspective: Given a context-free grammar G whose language $L(G)$ contains the sequence x to be compressed, how can we design an efficient lossless data compression algorithm to compress x ? In this setting $L(G)$ is allowed to contain more than one sequence. For instance, $L(G)$ could be the set of application-specific files such as the set of all HTML files, the set of all Microsoft Word document files, the set of all executable files, etc. Each sequence $x \in L(G)$ is generated by the application of a sequence of production rules of G . In this sense, it is natural to regard G as a source generating model. Note that even though the exact sequence of production rules which generates x is unknown, it does not seem justifiable to regard x as a sample sequence of a sequential random process. This new concept of using G as a source generating model enables us to propose a new theoretic framework for compression. Many interesting problems, including the one raised above, arise in this new framework. We shall elaborate more on these problems in Section IV.

Throughout this paper, we shall use the following notation. $\mathcal{A} = \{a_1, a_2, \dots, a_{|\mathcal{A}|}\}$ denotes a finite source alphabet with cardinality $|\mathcal{A}|$ greater than or equal to 2. $\mathcal{N} = \{1, 2, \dots\}$ denotes the set of all positive integers. \mathcal{A}^* denotes the set of all finite strings drawn from \mathcal{A} , including the empty string λ , and \mathcal{A}^+ denotes the set of all finite strings of positive length from \mathcal{A} . The notation $|x|$ denotes the length of x for any $x \in \mathcal{A}^*$. For any positive integer n , \mathcal{A}^n denotes the set of all sequences of length n from \mathcal{A} . Let $x = x_1x_2 \dots x_n$ be a length n sequence from \mathcal{A} . We shall sometimes write the substring $x_i \dots x_j$ ($1 \leq i \leq j \leq n$) of x as x_i^j for brevity. Similar notation will be applied to other sets and finite strings drawn from them. To avoid possible confusion, a sequence from \mathcal{A} is sometimes called an \mathcal{A} -sequence.

II. PERSPECTIVE 1: PERFORMANCE BENCHMARKS AND REDUNDANCY RESULTS

In [1], [2], the performance of grammar-based codes was analyzed by comparing with that of the best arithmetic coding algorithms with finite contexts. Let k be a positive integer. Let Θ_k denote the set of all arithmetic coding algorithms with k contexts. For any sequence $x = x_1x_2 \dots x_n$ from \mathcal{A} and any arithmetic coding algorithm $\theta \in \Theta_k$, let r^θ denote the

compression rate in bits per letter resulting from using θ to encode x . Define

$$r_k^*(x) \triangleq \min_{\theta \in \Theta_k} r^\theta(x), \quad (3)$$

i.e., $r_k^*(x)$ represents the smallest compression rate in bits per letter among all arithmetic coding algorithms with k contexts. Following the naming convention in [2], we shall call $r_k^*(x)$ the *k-context empirical entropy* of x .

Let $X = \{X_i\}_{i=1}^\infty$ denote an alphabet \mathcal{A} finite-state source with no more than k states. Then it can be shown for any individual sequence x emitted from X , $r_k^*(x)$ gives the ‘ideal’ lossless compression rate in bits per letter. Thus, $r_k^*(x)$ serves as a desirable performance benchmark in evaluating the compression performance of a lossless data compression algorithm for individual sequences emitted from finite-state sources.

Let ϕ denote an *arbitrary* grammar-based code throughout this section. Let $r^\phi(x)$ be the compression rate in bits per letter resulting from using ϕ to compress x . Define

$$R_{n,k}^\phi \triangleq \max_{x \in \mathcal{A}^n} [r^\phi(x) - r_k^*(x)].$$

The quantity $R_{n,k}^\phi$ is called the *worst-case redundancy* of ϕ against the k -context empirical entropy. The following upper bound of $R_{n,k}^\phi$ was established in [1], [2].

Result 1 There is a constant d , which depends only on $|\mathcal{A}|$ and k , such that

$$R_{n,k}^\phi \leq d \frac{\log \log n}{\log n}.$$

As alluded to in the introduction section, in this section we shall derive new redundancy results by introducing new performance benchmarks. To this end, we shall first define semi-finite-state sources; then propose context-based run-length encoding (RLE) algorithms, and show that context-based RLE algorithms are indeed superior to arithmetic coding algorithms with finite contexts; and finally evaluate the compression performance of grammar-based codes against that of the best context-based RLE algorithms. As a consequence, a redundancy result stronger than Result 1 will be established.

Before giving the definition of semi-finite-state sources, we give a brief description of run-length parsing. Let $X = \{X_i\}_{i=1}^\infty$ denote a source with alphabet \mathcal{A} . The source $X = \{X_i\}_{i=1}^\infty$ can also be represented by a sequence of symbol-run pairs $\{(Y_j, R_j)\}_{j=1}^\infty$ (also called the run-length parsing of X), where $Y_1 = x_1$, R_1 is the run length of Y_1 in the beginning of X , $Y_2 = X_{R_1+1}$, which of course is not equal to the symbol of X_{R_1} , R_2 is the run length of Y_2 in X immediately after the first run of Y_1 , and so on. For example, consider a sequence $x = a_1a_1a_1a_2a_2a_2a_3a_3$; the corresponding sequence of symbol-run pairs is given by $(a_1, 3)(a_2, 4)(a_3, 2)$. Clearly, there is a one-to-one correspondence between $\{X_i\}_{i=1}^\infty$ and $\{(Y_j, R_j)\}_{j=1}^\infty$.

Let k be a positive integer. Let \mathcal{Z} be a finite set consisting of k contexts (or states). An alphabet \mathcal{A} source $X = \{X_i\}_{i=1}^\infty$ is called a *semi-finite-state source with k states* if there exist an initial state z_0 , a transition probability function $q_0 : \mathcal{Z} \times (\mathcal{A} \times \mathcal{Z}) \rightarrow [0, 1]$ from \mathcal{Z} to $\mathcal{A} \times \mathcal{Z}$, and two transition probability functions $Q_i : (\mathcal{A} \times \mathcal{Z}) \times (\mathcal{A} \times \mathcal{N} \times \mathcal{Z}) \rightarrow [0, 1]$, $i = 0, 1$, from $\mathcal{A} \times \mathcal{Z}$ to $\mathcal{A} \times \mathcal{N} \times \mathcal{Z}$ satisfying $Q_i(a, r, z|a, z') = 0$ for any $a \in \mathcal{A}$, $r \in \mathcal{N}$, and $z, z' \in \mathcal{Z}$, such that the probability $\Pr\{X_1 \dots X_n = x_1 \dots x_n\}$ is equal to (4) for all $n \geq 1$ and

$x_1x_2 \cdots x_n$ from \mathcal{A} . In (4), $(y_1, r_1)(y_2, r_2) \cdots (y_{t_x}, r_{t_x})$, $1 \leq t_x \leq n$, denotes the sequence of symbol-run pairs associated with $x_1x_2 \cdots x_n$. For brevity, we shall write the quantity given by (4) as $\Delta_{sfs,k}(q_0, Q_0, Q_1, x, z_0)$.

For any positive integer k , let $\Lambda_{sfs}^k(\mathcal{A})$ denote the class of all alphabet \mathcal{A} semi-finite-state sources with k states. It is easy to check that $\Lambda_{sfs}^1(\mathcal{A})$ includes all renewal processes [9] when \mathcal{A} is binary. We are also interested in the relationship between semi-finite-state sources and finite-state sources. Since for any source X from $\Lambda_{sfs}^k(\mathcal{A})$, with probability one X does not have a run of infinite length, we here define the so-called *degenerate finite-state sources*.

Let $X = \{X_i\}_{i=1}^\infty$ denote an alphabet \mathcal{A} finite-state source with transition probability function p from \mathcal{Z} to $\mathcal{A} \times \mathcal{Z}$. Then X is called a *degenerate finite-state source with k states* if

$$\begin{aligned} & \Pr\{X_2 = a, X_3 = a, \dots, X_n = a, \dots | z_1\} \\ &= \sum_{z_2 \in \mathcal{Z}, \dots, z_n \in \mathcal{Z}, \dots} \prod_{i=2}^{\infty} p(a, z_i | z_{i-1}) \\ &> 0 \end{aligned} \quad (5)$$

for some $(z_1, a) \in \mathcal{Z} \times \mathcal{A}$. Otherwise, X is called a *non-degenerate finite-state source with k states*.

Let $\tilde{\Lambda}_{fs}^k(\mathcal{A})$ denote the class of all alphabet \mathcal{A} non-degenerate finite-state sources with k states. The following theorem shows that $\Lambda_{sfs}^k(\mathcal{A})$ includes $\tilde{\Lambda}_{fs}^k(\mathcal{A})$ as a strict subclass. The proof of Theorem 1 along with proofs of other theorems in this section are provided in [6].

Theorem 1 Let k be a positive integer. Then,

- i) any alphabet \mathcal{A} non-degenerate finite-state source with k states is a semi-finite-state source with k states;
- ii) there exist semi-finite-state sources $X = \{X_i\}_{i=1}^\infty$ with alphabet \mathcal{A} such that $X \notin \Lambda_{sfs}^k(\mathcal{A})$ no matter how large k is.

The definition of semi-finite-state sources implies that in order to efficiently compress a sequence emitted from a semi-finite-state source, one can encode the sequence indirectly by encoding the sequence of symbol-run pairs associated with the sequence. In the literature, such an algorithm is called a RLE algorithm [8]. Traditionally, a RLE algorithm uses Huffman coding to encode the sequence of symbol-run pairs. As a result, it may not be optimal since the length of each codeword must be an integer. One way to improve its compression performance is to replace Huffman coding with arithmetic coding or more specifically multilevel arithmetic coding [13]. To avoid possible confusion, we shall refer to such a modified RLE algorithm as an arithmetic RLE algorithm, and the original RLE algorithm as a Huffman RLE algorithm. If the arithmetic coding algorithm used in an arithmetic RLE algorithm is context-based with k contexts, then such an arithmetic RLE algorithm will be called a *RLE algorithm with k contexts*.

We next show that in comparison with arithmetic coding algorithms with k contexts, it is desirable to use RLE algorithms with k contexts to evaluate the compression performance of universal source coding algorithms including grammar-based codes.

Let \mathcal{Z} be a finite set consisting of k elements; each element $z \in \mathcal{Z}$ is regarded as an abstract context. Let $q_0 : \mathcal{Z} \times (\mathcal{A} \times \mathcal{Z}) \rightarrow [0, 1]$ be a transition probability function from \mathcal{Z} to

$\mathcal{A} \times \mathcal{Z}$; let $Q_i : (\mathcal{A} \times \mathcal{Z}) \times (\mathcal{A} \times \mathcal{N} \times \mathcal{Z}) \rightarrow [0, 1]$, $i = 0, 1$, be two transition probability functions from $\mathcal{A} \times \mathcal{Z}$ to $\mathcal{A} \times \mathcal{N} \times \mathcal{Z}$ satisfying $Q_i(a, r, z | a, z') = 0$ for any $a \in \mathcal{A}$, $r \in \mathcal{N}$, and $z, z' \in \mathcal{Z}$. For any sequence $x = x_1x_2 \cdots x_n$ from \mathcal{A} , the compression rate in bits per letter resulting from using the k -context RLE algorithm with functions q_0 , Q_0 , and Q_1 and an initial state z_0 to encode x is given by

$$-\frac{1}{n} \log \Delta_{sfs,k}(q_0, Q_0, Q_1, z_0, x).$$

Define

$$r_{sr,k}^*(x) \triangleq -\frac{1}{n} \log \max_{q_0, Q_0, Q_1} \max_{z_0 \in \mathcal{Z}} \Delta_{sfs,k}(q_0, Q_0, Q_1, z_0, x) \quad (6)$$

where the outer maximization varies over all possible combinations of transition probability functions q_0 , Q_0 , and Q_1 . The quantity $r_{sr,k}^*(x)$ represents the smallest compression rate in bits per letter among all RLE algorithms with k contexts. It should be emphasized that there is no single RLE algorithm with k contexts that can achieve the compression rate $r_{sr,k}^*(x)$ for every sequence $x = x_1x_2 \cdots x_n$. Since symbols and runs are encoded by arithmetic coding with k contexts, we shall call $r_{sr,k}^*(x)$ the *k -context symbol-run empirical entropy* of the sequence x .

The following two theorems together show that it is more desirable to use the k -context symbol-run empirical entropy $r_{sr,k}^*(x)$ as the benchmark than to use $r_k^*(x)$ to evaluate the compression performance of universal source coding algorithms.

Theorem 2 Let k be a positive integer.

- i) Let $X = \{X_i\}_{i=1}^\infty$ be an alphabet \mathcal{A} semi-finite-state source with k states. Then

$$r_{sr,k}^*(X_1^n) \leq -\frac{1}{n} \log \Pr\{X_1^n\}. \quad (7)$$

- ii) For any alphabet \mathcal{A} source $X = \{X_i\}_{i=1}^\infty$, there exists a constant $d > 0$ depending only on k and \mathcal{A} such that for any sufficiently large n ,

$$r_{sr,k}^*(X_1^n) \geq -\frac{1}{n} \log \Pr\{X_1^n\} - d \frac{1}{\sqrt{n}} \quad (8)$$

with probability greater than or equal to $1 - 1/n^2$.

Theorem 3 Let k be a positive integer. Then, for any sequence $x \in \mathcal{A}^+$,

$$r_{sr,k}^*(x) \leq r_k^*(x). \quad (9)$$

Furthermore, there exist sequences x for which the above inequality becomes a strict inequality.

Remark 1 Fix a positive integer k . Theorem 3 shows that $r_{sr,k}^*(x)$ is no greater than $r_k^*(x)$ for any individual sequence x from a finite alphabet, and there exist sequences for which (9) becomes a strict inequality. However, being smaller than $r_k^*(x)$ alone does not mean that $r_{sr,k}^*(x)$ is a useful quantity from an information theoretic point of view; one has to guarantee that $r_{sr,k}^*(x)$ can not be essentially less than the self-information. That is why in Theorem 2, we relate $r_{sr,k}^*(X_1 \cdots X_n)$ to the self-information $-\log \Pr\{X_1 \cdots X_n\}$ for any source $X = \{X_i\}_{i=1}^\infty$ with a finite alphabet.

$$\Pr\{X_1^n = x_1^n\} = \begin{cases} \sum_{z_1, z_2 \in \mathcal{Z}} q_0(y_1, z_1|z_0) \sum_{a \in \mathcal{A}} \sum_{l=r_1}^{\infty} Q_0(a, l, z_2|y_1, z_1) & \text{if } t_x = 1 \\ \sum_{z_1, \dots, z_{t_x} \in \mathcal{Z}} \left[q_0(y_1, z_1|z_0) Q_0(y_2, r_1, z_2|y_1, z_1) \right. \\ \left. \times \prod_{j=2}^{t_x-1} Q_1(y_{j+1}, r_j, z_{j+1}|y_j, z_j) \sum_{a \in \mathcal{A}} \sum_{l=r_{t_x}}^{\infty} \sum_{z \in \mathcal{Z}} Q_1(a, l, z|y_{t_x}, z_{t_x}) \right] & \text{if } t_x > 1 \end{cases}. \quad (4)$$

Finally in this section we revisit grammar-based codes by evaluating their compression performance against the k -context symbol-run empirical entropy for any individual sequence from a finite alphabet. In particular, we are interested in the difference between $r^\phi(x)$ and $r_{sr,k}^*(x)$ for any individual sequence $x \in \mathcal{A}^+$. Define

$$R_{n, sr, k}^\phi \triangleq \max_{x \in \mathcal{A}^n} [r^\phi(x) - r_{sr,k}^*(x)].$$

The quantity $R_{n, sr, k}^\phi$ is called the *worst-case redundancy* of the grammar-based code ϕ against the k -context symbol-run empirical entropy. In the following theorem, we upper-bound $R_{n, sr, k}^\phi$.

Theorem 4 There is a constant d , which depends only on \mathcal{A} and k , such that

$$R_{n, sr, k}^\phi \leq d \frac{\log \log n}{\log n}.$$

Theorems 4, together with Theorem 3, implies Result 1. Hence, the new redundancy result is stronger than Result 1.

III. PERSPECTIVE 2: UNIVERSALITY FOR SOURCES WITH COUNTABLY INFINITE ALPHABETS

As mentioned in the introduction section, grammar-based codes were originally designed and analyzed for sources with *finite alphabets*. In this section, we show how grammar-based codes can be extended to encode sources with countably infinite alphabets, and analyze their compression performance.

Let \mathcal{E} be a countably infinite source alphabet throughout this section. Let $x \in \mathcal{E}^n$ denote a sequence to be compressed. For any x , let \mathcal{E}_x denote the set that consists of all distinct symbols appearing in x . Since x is of finite length, \mathcal{E}_x is a finite subset of \mathcal{E} . However, \mathcal{E}_x may grow without bound as x gets longer and longer. Because \mathcal{E}_x is unknown to the decoder, it is clear that in order to encode x efficiently, the set \mathcal{E}_x has to be encoded and transmitted to the decoder. Note that such issue does not exist in the case of finite alphabet. Having the knowledge of \mathcal{E}_x , one can then regard x as a sequence from \mathcal{E}_x , which is finite if the length n of x is finite. Thus, by incorporating the encoding of \mathcal{E}_x , any grammar-based code can be applied to encode x as if it is from \mathcal{E}_x .

Since \mathcal{E} can be an arbitrary collection of symbols, it is generally not efficient to encode \mathcal{E}_x directly. Nonetheless, because \mathcal{E} is known to both encoder and decoder, encoding \mathcal{E}_x can be simplified as encoding integers. To facilitate our discussion, let $L: \mathcal{E} \rightarrow \mathcal{N}$ denote a known one-to-one and onto deterministic mapping from \mathcal{E} to \mathcal{N} . For example, a labeling method that assigns a unique positive integer index to each symbol e in \mathcal{E} gives rise to a mapping L . It is then easy to see that every symbol $e \in \mathcal{E}_x$ can be encoded indirectly by encoding the integer $L(e)$. For simplicity, we choose the Elias universal doubly compound representation of integers or simply the Elias code described in [10] to encode integers.

Remark 2 It should be pointed out that even though the modification of grammar-based codes needed for encoding sources with countably infinite alphabets is minor, it does not mean that the extension of the performance analysis results of grammar-based codes for sources with finite alphabets to the case of sources with infinite alphabets is straightforward. To the contrary, it is hardly overstated that due to the fact that \mathcal{E}_x grows without bound for most sequences as $n \rightarrow \infty$, the problem becomes more difficult.

Before analyzing the performance of grammar-based codes for sources with alphabet \mathcal{E} , it is necessary to review a result obtained by Kieffer over 25 years ago. Let Λ denote a class of alphabet \mathcal{E} stationary, ergodic sources. Given Λ , it was proved in [11] that a universal code for Λ exists if and only if there is a countable set $\mathcal{P} = \{p_1, p_2, \dots\}$ of probability distributions on \mathcal{E} such that $\Lambda \subseteq \Lambda_{\mathcal{P}}$, where $\Lambda_{\mathcal{P}}$ consists of all stationary, ergodic alphabet \mathcal{E} sources $X = \{X_i\}_{i=1}^{\infty}$ for which there exists a $p_X^* \in \mathcal{P}$, which depends on $\{X_i\}_{i=1}^{\infty}$, such that

$$E[-\log p_X^*(X_1)] < \infty. \quad (10)$$

Therefore, unlike the case of finite source alphabet, a universal code does not exist for the class of all stationary, ergodic sources with a countably infinite alphabet or even for the class of all memoryless sources with a countably infinite alphabet [12].

In light of the above Kieffer's result, the most general question one could possibly ask about the the universality of grammar-based codes for sources with alphabet \mathcal{E} is as follows:

Q1 Given any Λ for which a universal code exists, are grammar-based codes with the modification described above universal with respect to Λ ?

In the following we shall settle the above question in the affirmative. Moreover, we shall also investigate the compression performance of grammar-based codes for individual sequences from alphabet \mathcal{E} .

Let x be a sequence from \mathcal{E} . Let ϕ denote an arbitrary grammar-based code that encodes \mathcal{E}_x as described above. Let $r^\phi(x)$ be the compression rate in bits per letter resulting from using ϕ to compress x . To evaluate $r^\phi(x)$, we define $r_b^*(x)$ as the smallest compression rate in bits per letter among all b th-order arithmetic coding algorithms, where b is a finite integer. In view of that $r_b^*(x)$ is equal to the empirical entropy of x when $b = 0$, we call $r_b^*(x)$ the *b th-order empirical entropy* of x . Define

$$R_{\mathcal{F}(\mathcal{E}^n), b}^\phi \triangleq \max_{x \in \mathcal{F}(\mathcal{E}^n)} [r^\phi(x) - r_b^*(x)],$$

where $\mathcal{F}(\mathcal{E}^n)$ is a subset of \mathcal{E}^n to be specified later. The reason for taking the maximization over a subset of \mathcal{E}^n , not \mathcal{E}^n itself, lies in the fact that there is no universal code for all

alphabet \mathcal{E} stationary, ergodic sources. The quantity $R_{\mathcal{F}(\mathcal{E}^n),b}^\phi$ is called the *worst-case redundancy* of the grammar-based code ϕ against the b th-order empirical entropy among $\mathcal{F}(\mathcal{E}^n)$.

Recall that in Result 1 the k -context empirical entropy $r_k^*(x)$ is used as the performance benchmark. The reason for using $r_b^*(x)$ instead of $r_k^*(x)$ here is that there exists a sequence emitted from a first-order Markov source with alphabet \mathcal{E} for which the first-order empirical entropy is strictly less than the k -context empirical entropy for any finite k .

The next two theorems are the main results of this section. Their proofs are provided in [7].

Theorem 5 Suppose that in ϕ , the encoding of \mathcal{E}_x for each $x \in \mathcal{E}^+$ is based on a known one-to-one and onto deterministic mapping $L : \mathcal{E} \rightarrow \mathcal{N}$.

1) Let

$$\mathcal{E}^n(L^\alpha, K_1) = \{x \in \mathcal{E}^n : \sum_{i=1}^n [L(x_i)]^\alpha < K_1 n\},$$

where α and K_1 are both positive constants. Then there is a constant d , which depends only on b , α , and K_1 , such that

$$R_{\mathcal{E}^n(L^\alpha, K_1), b}^\phi \leq d \frac{\log \log n}{\log n}. \quad (11)$$

2) Let

$$\mathcal{E}^n(\log^\gamma L, K_2) = \{x \in \mathcal{E}^n : \sum_{i=1}^n (\lfloor \log^\gamma L(x_i) \rfloor + 1) < K_2 n\},$$

where K_2 is a positive constant. If $\gamma > 1$, then there is a constant d' , which depends only on b and K_2 , such that

$$R_{\mathcal{E}^n(\log^\gamma L, K_2), b}^\phi \leq d' \frac{1}{\log^{1-\frac{1}{\gamma}} n}. \quad (12)$$

Remark 3 (11) in Theorem 5 is stronger than the result in [14] which shows that the incremental parsing Lempel-Ziv code [3] with modifications is asymptotically optimal for any stationary, ergodic source $\{X_i\}_{i=1}^\infty$ with alphabet \mathcal{N} satisfying³

$$EX_1^\alpha < \infty, \quad (13)$$

where α is a positive constant. (A discussion on the relationship between the defining condition of $\mathcal{E}^n(L^\alpha, K_1)$ and (13) is provided in [7] for interested readers.) The upper bound in (12), which approaches one as γ goes to 1, is due to considering the worst case scenario, in which the encoding of individual symbols that may appear in the sequence to be compressed can be extremely inefficient. On the other hand, the quantity $r_b^*(x)$ is defined in a way that does not take into account the overhead of encoding \mathcal{E}_x . It can be shown that by imposing some slightly stronger restrictions, we can obtain better upper bounds of $R_{\mathcal{F}(\mathcal{E}^n),b}^\phi$ for some $\mathcal{F}(\mathcal{E}^n) \subset \mathcal{E}^n(\log^\gamma L, K_2)$.

Remark 4 Theorem 5 suggests that the performance of the modified grammar-based code ϕ depends on the behavior of the mapping L chosen to encode individual symbols in \mathcal{E}_x . This is particularly evident in (12), which shows that a poor

selection of L will result in a large worst-case redundancy among a certain set of sequences. Clearly, to design a good mapping L , one needs some prior information about the data sequences to be compressed. Fortunately, in practice, for example, in applications like audio compression and image compression, one does have such prior knowledge that can be utilized in the design of L .

Theorem 6 Let $\mathcal{P} = \{p_1, p_2, \dots\}$ be a countable set of probability distributions on \mathcal{E} . Then for any stationary, ergodic source $\{X_i\}_{i=1}^\infty \in \Lambda_{\mathcal{P}}$,

$$r^\phi(X_1 X_2 \cdots X_n) \rightarrow H(X)$$

with probability one as $n \rightarrow \infty$.

Clearly, Theorem 6 provides a positive answer to Question Q1 posed earlier in this section.

IV. PERSPECTIVE 3: USING GRAMMARS AS SOURCE MODELS

In this section, we elaborate on the idea of using grammars as source generating models. From this perspective, we propose a new theoretic framework for compression, and point out some open problems in the framework.

As mentioned in the introduction section, we shall allow our context-free grammars to have more than one sequence in their languages. This indicates that a single variable may be associated with more than one production rule in a grammar. To better illustrate this, let us look at the following example.

Example 1: Let G be a context-free grammar with variable set $\{s\}$ and terminal set $\{a, b\}$ consisting of the following four production rules:

1. $s \rightarrow bss$
2. $s \rightarrow ss$
3. $s \rightarrow a$
4. $s \rightarrow ab$

Thus, the language $L(G)$ of the grammar G contains both the $\{a, b\}$ -sequence

$baab$

generated by invoking production rules 1, 3, and 4 in the specified sequential order, and the $\{a, b\}$ -sequence

$baababa$

generated by invoking production rules 1, 2, 2, 3, 4, 4, and 3, or production rules 2, 1, 2, 3, 4, 4, and 3 sequentially.

Let $x = x_1 x_2 \cdots x_n$ be a sequence in the language $L(G)$ of a context-free grammar G . Then there exists a sequence of production rules in G such that x can be generated by invoking these rules sequentially. Denote such a sequence of rules by $u = u_1 u_2 \cdots u_m$, where each $u_i \in \mathcal{N}$ is the index of a production rule in G . To facilitate our discussion, we shall write $u \xrightarrow{G} x$ to indicate that x can be generated by invoking the sequence of production rules denoted by u . Suppose that to each sequence u from \mathcal{N} , the grammar G assigns a probability $p_G(u)$. That is, G is a *stochastic grammar*. Then the probability $p(x|G)$ that x is generated by G is

$$p(x|G) = \sum_{u \in \mathcal{N}^+ : u \xrightarrow{G} x} p_G(u).$$

³Inspired by [7], the condition (13) was weakened to be $E[\log X_1] < \infty$ in the recent preprint of Uyematsu and Kanaya [15] with the same title as [14].

Equivalently we have a source generating model that emits sequences x with probability $p(x|G)$. This explains why we can use a grammar as a source generating model. Observe also that though to generate x we invoke the production rules in u sequentially, it does not seem justifiable to regard x as a sample sequence of a sequential random process.

Using grammars as source generating models, we can now propose our theoretic framework for compression. Let us first consider the coding of a sequence x in the language $L(G)$ generated by a context-free grammar G that is known to both the encoder and the decoder. Clearly, one method to encode x is to encode a sequence u such that $u \xrightarrow{G} x$. When the set $\{u \in \mathcal{N}^+ : u \xrightarrow{G} x\}$ consists of multiple sequences, one can encode the sequence u^* among $\{u \in \mathcal{N}^+ : u \xrightarrow{G} x\}$ such that $p_G(u^*)$ is a maximum. However, to justify this approach, the following problems need to be solved. At the moment of writing this paper, these problems remain open.

Problem 1 Under what condition does the following equality hold?

$$-\log p(x|G) = -\log p_G(u^*) + o(|x|).$$

Problem 2 How can we find the sequence u^* efficiently for any individual sequence in $L(G)$?

Problem 3 How can we encode the index sequence u^* efficiently?

In a more general case, we assume that G is an unknown grammar in a set \mathcal{G} . Thus to encode a sequence x , we need to first find the grammar G such that $x \in L(G)$, and then encode x as a member sequence in $L(G)$. Following this approach, the compression rate in bits per symbol is always lower bounded by

$$\rho_{\mathcal{G}}^*(x) \triangleq \min_{G \in \mathcal{G}: x \in L(G)} [r_G(x)], \quad (14)$$

where $r_G(x)$ denotes the compression rate in bits per symbol resulting from encoding x as a sequence in $L(G)$. It is easy to see that when the set \mathcal{G} is equal to the set of all Turing machines, the quantity $\rho_{\mathcal{G}}^*(x)$ becomes the Kolmogorov complexity of the sequence x , and when \mathcal{G} consists of all context-free grammars, $\rho_{\mathcal{G}}^*(x)$ is no greater than the finite-state compressibility of x [3]. For this reason, we call $\rho_{\mathcal{G}}^*(x)$ the *grammar compressibility* of the sequence x given \mathcal{G} . Note that in the cases when \mathcal{G} is a subset of the set of all context-free grammars, the quantity $\rho_{\mathcal{G}}^*(x)$, in contrast to the incomputable Kolmogorov complexity, is computable for any individual sequence x .

In this general case, many more interesting problems arise. Focusing on the grammar compressibility, we present the following two open problems.

Problem 4 How to characterize the grammar compressibility and develop corresponding coding theorems.

Problem 5 How to develop lossless data compression algorithms that can outperform asymptotically $\rho_{\mathcal{G}}^*(x)$ for any x .

We conclude this section with a comment on Problem 5 above. For any grammar set \mathcal{G}_1 , one approach to developing algorithms that outperform asymptotically $\rho_{\mathcal{G}_1}^*(x)$ for any x is to look at a larger set \mathcal{G}_2 that includes \mathcal{G}_1 as a strict subset. Therefore, we may envision a hierarchy of grammar sets,

$$\mathcal{G}_1 \subset \mathcal{G}_2 \subset \dots \subset \mathcal{G}_m \subset \dots$$

such that for each $i \geq 1$, \mathcal{G}_{i+1} gives rise to a lossless data compression algorithm that can outperform asymptotically $\rho_{\mathcal{G}_i}^*(x)$ for any x . This hierarchy may be regarded as a refined version of the well-known Chomsky hierarchy, at the top of which is the set of all Turing machines.

V. CONCLUSION

This paper has presented a few new perspectives to investigate grammar-based coding. As we have seen in Section IV above, many interesting problems remain open. We hope that this paper will inspire more work on grammar-based coding either along the lines discussed here or from completely new perspectives.

REFERENCES

- [1] J. C. Kieffer and E.-H. Yang, "Grammar based codes: A new class of universal lossless source codes," *IEEE Trans. Inform. Theory*, vol. 46, pp. 737–754, 2000.
- [2] E.-H. Yang and J. C. Kieffer, "Efficient universal lossless data compression algorithms based on a greedy sequential grammar transform—Part one: Without context models," *IEEE Trans. Inform. Theory*, vol. 46, pp. 755–788, 2000.
- [3] J. Ziv and A. Lempel, "Compression of individual sequences via variable rate coding," *IEEE Trans. Inform. Theory*, vol. 24, pp. 530–536, 1978.
- [4] J. C. Kieffer, E.-H. Yang, G. Nelson, and P. Cosman, "Universal lossless compression via Multilevel Pattern Matching," *IEEE Trans. Inform. Theory*, vol. 46, pp. 1227–1245, 2000.
- [5] E.-H. Yang and D.-K. He, "Efficient universal lossless data compression algorithms based on a greedy sequential grammar transform—Part two: With context models," *IEEE Trans. Inform. Theory*, vol. 49, pp. 2874–2894, 2003.
- [6] D.-K. He and E.-H. Yang, "Performance analysis of grammar-based codes revisited," *IEEE Trans. Inform. Theory*, vol. 50, pp. 1524–1535, 2004.
- [7] D.-K. He and E.-H. Yang, "The universality of grammar-based codes for sources with countably infinite alphabets," revised and submitted to *IEEE Trans. Inform. Theory* for second round review, 2004.
- [8] S. W. Golomb, "Run-length encodings," *IEEE Trans. Inform. Theory*, vol. 12, pp. 399–401, 1966.
- [9] I. Csiszár and P. C. Shields, "Redundancy rates for renewal and other processes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 2065–2072, 1996.
- [10] P. Elias, "Universal codeword sets and representations of the integers," *IEEE Trans. Inform. Theory*, vol. 21, pp. 193–203, 1975.
- [11] J. C. Kieffer, "A unified approach to weak universal source coding," *IEEE Trans. Inform. Theory*, vol. 24, pp. 674–682, 1978.
- [12] L. Györfi and I. Páli, "There is no universal code for an infinite source alphabet," *IEEE Trans. Inform. Theory*, vol. 40, pp. 267–271, 1994.
- [13] E.-H. Yang and Y. Jia, "Universal lossless coding of sources with large or unbounded alphabets," In *Numbers, Information and Complexity*, Ingo Althöfer, et al., Eds. Kluwer Academic Publishers, 2000, pp. 421–442.
- [14] T. Uyematsu and F. Kanaya, "Asymptotical optimality of two variations of Lempel-Ziv codes for sources with countably infinite alphabet," In *Proc. of the 2002 IEEE International Symposium on Information Theory*, Lausanne, Switzerland, 2002.
- [15] T. Uyematsu and F. Kanaya, "Asymptotical optimality of two variations of Lempel-Ziv codes for sources with countably infinite alphabet," *Preprint*.