RePair and All Irreducible Grammars Are Upper Bounded by High-Order Empirical Entropy

Carlos Ochoa and Gonzalo Navarro

***Abstract*—Irreducible grammars are a class of context-free grammars with well-known representatives, such as REPAIR (with a few tweaks), LONGEST MATCH, GREEDY, and SEQUENTIAL. We show that a grammar-based compression method described by Kieffer and Yang (2000) is upper bounded by the high-order empirical entropy of the string when the underlying grammar is irreducible. Specifically, given a string S over an alphabet of size σ, we prove that if the underlying grammar is irreducible, then the length of the binary code output by this grammar-based compression method is bounded by $|S|H_k(S) + o(|S|\log\sigma)$ for any $k \in o(\log_\sigma |S|)$, where $H_k(S)$ is the k-order empirical entropy of S. This is the first bound encompassing the whole class of irreducible grammars in terms of the high-order empirical entropy, with coefficient 1.**

***Index Terms*—Data compression, empirical entropy, grammar-based compression, irreducible grammars, RePair.**

I. INTRODUCTION

GRAMMAR-BASED compression [1] is a technique that, given a string to compress S, builds a context-free grammar G_S that generates S and only S. The intuition is that, if S has redundancies, then G_S can be significantly smaller than S. The actual output of the compressor is a prefix-free binary code $B(G_S)$, that is, no other string S' can generate a code $B(G_{S'})$ that is a prefix of $B(G_S)$.

Grammar-based compression methods are attractive in comparison with entropy-based compression methods because they can be implemented more efficiently, especially at decompression. They are also preferable over other popular dictionary-based compression methods like the Lempel-Ziv family [2] because they allow accessing substrings of S directly in compressed form in logarithmic time [3]. This makes them particularly useful to implement compact data structures [4].

Kieffer and Yang [1] studied the effectiveness of grammar-based compression. In particular, they gave minimal conditions to ensure that the grammar G_S has no obvious undesirable redundancies, calling *irreducible* the grammars that satisfy those conditions. Various popular grammar-based compressors have been proved irreducible [1], [5], for example:

REPAIR (Larsson and Moffat 1999 [6]) repeatedly finds the most frequent pair XY of symbols in S, replaces it with a new variable $\mathcal{A}$, and adds the production rule $\mathcal{A} \rightarrow XY$ to G_S. The resulting grammar is not necessarily irreducible, but it is easily modified to be so.

LONGEST MATCH (Kieffer and Yang 2000 [1]) creates the production rule $\mathcal{S} \rightarrow S$, and then repeatedly finds the longest substring α that appears at least twice in the right-hand sides of the production rules, replaces it with a new variable $\mathcal{A}$, and adds the rule $\mathcal{A} \rightarrow \alpha$ to G_S.

GREEDY (Apostolico and Lonardi 2000 [7]) starts with $\mathcal{S} \rightarrow S$ and repeatedly finds the rule $\mathcal{A} \rightarrow \alpha$ that reduces $|G_S|$ the most, where $|G_S|$ is defined as the sum of the lengths of the right-hand sides of the production rules.

SEQUENTIAL (Kieffer and Yang 2000 [1]) processes S left to right. At each step, it first finds the longest prefix of what remains of S that matches the expansion of a variable $\mathcal{A}$, and appends $\mathcal{A}$ to the start rule (or appends a symbol of S if the longest prefix is empty). Second, if the last two symbols appended appear elsewhere in G_S (without overlap), it creates a rule to replace all those pairs. Third, if after the second step some variable occurs only once, its occurrence is replaced by its definition and the variable is deleted.

Kieffer and Yang [1] described an encoding $B(G_S)$ that, when G_S is irreducible, asymptotically matches the entropy of any finite-state source (see their Thm. 8). This gives a theoretical basis to the good empirical performance displayed by the methods above [6], [7].

In this article we are interested in bounding the size of the output of irreducible grammar-based compressors in terms of the compressibility of the *individual sequences*, instead of that of information sources. This is useful for deriving *worst-case* bounds on the size of compressed string representations that depend solely on the string S being represented, without having to assume that S comes from a particular information source. Such a measure is stronger than deriving a plausible information source from the observed frequencies in S and then obtaining an *expected-case* result for strings generated from that source [8], [9].

The ideal measure of compressibility of individual sequences is the Kolmogorov complexity [10], however this is unfortunately not computable. A popular alternative is then the so-called *k-order empirical entropy of S* [8], $H_k(S)$, which is aimed to lower-bounding the performance of statistical compressors: any encoder that encodes each symbol of S as a function of its k preceding symbols only, cannot output less than $|S|H_k(S)$ bits. This is then a lower

Manuscript received May 30, 2018; revised September 3, 2018; accepted September 8, 2018. Date of publication September 20, 2018; date of current version April 19, 2019. This work was supported in part by Fondecyt under Grant 1-170048 and in part by Basal Funds FB0001, Conicyt, Chile.

The authors are with the Center for Biotechnology and Bioengineering, Department of Computer Science, University of Chile, Santiago 8370456, Chile (e-mail: cochoa@dcc.uchile.cl; gnavarro@dcc.uchile.cl).

Communicated by V. Vaishampayan, Associate Editor for Source Coding.

Digital Object Identifier 10.1109/TIT.2018.2871452

bound to the output size of any static statistical k-th order compressor on S.

Various families of compressors, both statistical and dictionary-based, are known to approach $H_k(S)$. In the former family, Debowski [11] shows that the output of Prediction by Partial Matching (PPM) [12] compressors of order k can be upper bounded by $|S|H_k(S) + o(|S| \log \sigma)$ on any string S with alphabet size σ, a bound that holds for any $k < (1-\epsilon)\log_\sigma |S| - 1$ and any constant $0 < \epsilon < 1$. A similar result [13] is obtained for adequate encodings of the output of the Burrows-Wheeler Transform (BWT) [14]. With respect to dictionary-based compressors, Kosaraju and Manzini [15] proved that the output size of several members of the Lempel-Ziv family, using simple encodings, is upper bounded by $|S|H_k(S) + o(|S| \log \sigma)$ bits for any $k \in o(\log_\sigma |S|)$. The situation, however, is less well understood on grammar-based compressors. Navarro and Russo [16] described an encoding for the grammar obtained by RePair, and bounded the resulting binary code by $2|S|H_k(S) + o(|S| \log \sigma)$ bits. Recently, Gańczorz [17] described a grammar encoding and proved that, built on any irreducible grammar, it is also of length at most $2|S|H_k(S) + o(|S| \log \sigma)$ bits. Further, he proved that, built on a variant of RePair where the pair generation process is preempted at a certain moment, the output size is bounded by $|S|H_k(S) + o(|S| \log \sigma)$ bits.

In this paper we improve and simplify the results of Gańczorz, obtaining for all irreducible grammars a general result analogous to that of Kosaraju and Manzini [15]: we use a result of Gagie [9] to show that the original encoding $B(G_S)$ of Kieffer and Yang [1] can be bounded in terms of the k-th order empirical entropy of S with coefficient 1, that is, for any $k \in o(\log_\sigma |S|)$, it holds that

$$|B(G_S)| \le |S|H_k(S) + o(|S| \log \sigma)$$

without any further condition on G_S. Since the empirical entropy of a string coming from a stationary ergodic source converges almost surely to the conditional entropy of such source, our bound also provides an alternative proof of the universality of irreducible grammar-based codes.

II. Basic Concepts

As of notation, all our logarithms will be in base 2 unless otherwise stated. In our complexity formulas, as it is customary, we take σ as a (constant or not) function of $|S|$.

A. Definitions

We start by defining admissible grammars, which generate exactly one string and avoid some kinds of useless variables.

Definition 1: A grammar G is *admissible* iff

1) each variable is the left-hand side of exactly one rule;
2) the empty string is not the right-hand side of any rule;
3) the generated language contains at least one string; and
4) every variable appears in some derivation of a string of the language.

Irreducible grammars, on the other hand, are stricter in avoiding unnecessary inefficiencies.

Definition 2: A grammar G is *irreducible* iff it is admissible and

1) distinct variables derive different strings;
2) every variable other than the start symbol appears more than once in the right-hand sides of the production rules of G; and
3) no pair X, Y of symbols exists such that the string XY appears more than once in nonoverlapping positions in the right-hand sides of the production rules of G (nonoverlapping means that we allow the pair X, X to appear twice in XXX).

Definition 3: By G_S we denote a grammar that generates S and only S, and by $|G_S|$ we denote the sum of the lengths of the right-hand sides of all the production rules in G_S.

We now define our measure of entropy for individual strings, which is the the conditional entropy of the empirical distribution induced by the string.

Definition 4: The *empirical entropy of order k* of a string S is defined as

$$H_k(S) = \begin{cases} \sum_{a \in A} p_a \log \frac{1}{p_a} & \text{if } k = 0, \\ \frac{1}{|S|} \sum_{|\alpha|=k} |S_\alpha| H_0(S_\alpha) & \text{if } k > 0, \end{cases}$$

where k is an integer, A is the alphabet of S, p_a is the relative frequency of the character a in S (i.e., the number of occurrences of a in S divided by the length of S), S_α is the string obtained by concatenating the single characters that immediately follow the occurrences of the substring α in S, and $|x|$ denotes the length of the string x.

Finally, we conclude with some definitions on the finite-state sources we will use to prove our results.

Definition 5: A *k-th order Markov process* is a string of random variables in which each variable depends only on its k immediate predecessors. If the domain of all the random variables are the characters of an alphabet, then the k-th order Markov process represents an information source where the probability of choosing a character depends on the k preceding characters.

Definition 6: Given a string $S = a_1 a_2 \dots a_{|S|}$ over an alphabet A assume, without loss of generality, that a k-th order Markov process Q emits the first k characters of S with probability 1. For every string $\gamma \in A^k$ (i.e., γ is a string of length k over the alphabet A), let $q_{\gamma,a}$ be the probability that Q emits the character a after an occurrence γ. Then, Q *emits* S with probability $Pr[Q \text{ emits } S] = \prod_{i=k+1}^{|S|} q_{a_{i-k}\dots a_{i-1}, a_i}$.

B. Grammar Encoding

Once a grammar G_S deriving S and only S has been obtained, the final phase of a grammar-based compression method is to *encode* G_S, that is, assign it a binary code $B(G_S)$ so that, for any two different grammars, the code assigned to one of them is not a prefix of the code assigned to the other. Given a string S over an alphabet of size σ, and a context-free grammar G_S in canonical form[1] that generates (only) S,

[1] A grammar is in canonical form if its variables are enumerated increasingly according to the first time they are found in a levelwise top-down left-to-right traversal of the derivation tree of S.

Kieffer and Yang [1] described a grammar encoding $\mathcal{G}$ that assigns a binary code $B(G_S)$ to G_S in the following way:

1) the number of variables in G_S is encoded using unary code;
2) the characters that appear in S are encoded using a binary string of size σ (i.e., each bit indicates whether the character appears or not in S);
3) the number of occurrences of the symbols (variables and alphabet symbols) in G_S are encoded in unary;
4) the lengths of the right-hand sides of the production rules are encoded in unary;
5) letting α be the string that results from concatenating the right-hand sides of the production rules in G_S, a binary string of length $|\alpha| = |G_S|$ encodes the first occurrence of each variable in α; and
6) letting β be the string obtained from α by removing the first occurrence of each variable, β is encoded using Huffman code [18].

The codes in the points 3–6 assume the canonical ordering of the variables in G_S. The grammar G_S can then be reconstructed from $B(G_S)$. For this encoding it holds that

$$B(G_S) \leq \lceil H_0(\beta) \rceil + 4|G_S| + \sigma,$$

where the dominant term is $H_0(\beta)$.

There are two noteworthy advantages of this grammar encoding, which makes it more efficient than the one used by Gańczorz [17]: (i) the right-hand side of the start symbol and the rest of the production rules in G_S are encoded together, and (ii) when encoding the right-hand sides of the production rules, it removes one occurrence of each variable $\mathcal{A}$ in G_S.

C. Markov Processes and Empirical Entropy

Gagie [9] proved a relation between the k-th order empirical entropy of a string S and the probability that a k-th order Markov process will emit S:

Lemma 1 [9, Thm. 1]: For any string S and any integer $k \geq 0$, $H_k(S) = \frac{1}{|S|} \min\{\log(1/Pr[Q \text{ emits } S])\}$, where the minimum is taken over all k-th order Markov processes Q.

III. K-th Order Empirical Entropy Bound

We now obtain our main result. The key of our development is to modify a proof in Kieffer and Yang [1] in order to bound the size of $B(G_S)$ in terms of the probability that the string S is emitted by a k-th order Markov process, when the underlying grammar is irreducible. Combining this result with Lemma 1, we obtain in Theorem 1 the promised bound for the binary code $B(G_S)$ in terms of the k-th order empirical entropy of S.

Given a string S, regardless the context-free grammar G_S that generates (only) S, Kieffer and Yang [1] proved a bound for the binary code $B(G_S)$ resulting from the grammar encoding $\mathcal{G}$ in function of the probability that an ℓ-order finite state source[2] will emit S. They divided this result in two steps: first, they bounded the size of $B(G_S)$ by the entropy $H(\pi)$ of a parsing π of S, and then bounded $H(\pi)$ in terms of the probability that an ℓ-order finite state source will emit S. We modify the second proof so as to state the bound in terms of the probability that a k-th order Markov process will emit S.

Definition 7: A *parsing* $\pi = \langle u_1, \ldots, u_t \rangle$ of a string S is a sequence of strings $u_1, \ldots, u_t$ such that S is the concatenation of $u_1, \ldots, u_t$. The length t of π is denoted by $|\pi|$. The *entropy* of π is

$$H(\pi) = \sum_{i=1}^{t} \log \frac{1}{p_{u_i}},$$

where p_u is the relative frequency of u in π (i.e., the number of occurrences of u divided by $|\pi|$).

Kieffer and Yang [1] bounded the size of the binary code $B(G_S)$ resulting from the grammar encoding $\mathcal{G}$ by the entropy $H(\pi)$ of a parsing π of S:

Lemma 2 [1, Lemma 8]: Given a string S over an alphabet of size σ, a context-free grammar G_S that generates (only) S, and the binary code $B(G_S)$ resulting from the grammar encoding $\mathcal{G}$, there is a parsing π of S of length at most $|G_S|$ satisfying $B(G_S) \leq H(\pi) + 5|G_S| + \sigma$.

The following lemma bounds the entropy $H(\pi)$ of any parsing π of a string S in terms of the probability that a k-th order Markov process will emit S. We modify a proof by Kieffer and Yang [1] to state the bound in terms of the probability that a k-th order Markov process will emit S instead of the probability that an ℓ-order finite state source will emit S.

Lemma 3 [1, Lemma 1]: Given a string S over an alphabet A of size σ, and any integer $k \geq 0$, any parsing π of S satisfies $H(\pi) \leq \log(1/Pr[Q \text{ emits } S]) + t(1 + 2k \log \sigma) + 2t \log \frac{|S|}{t}$, where Q is any k-th order Markov process, and $t = |\pi|$.

Proof: Consider the probability that a k-th order Markov process Q emits S. Let $\tau : A^+ \to [0,1]$ be the function $\tau(u) = \max_{\gamma \in A^k} Pr[Q \text{ emits } u|_\gamma]$, where $u|_\gamma$ is the string that results from replacing the first k characters of u by γ. If the length of u is less than k, then we assume that $\tau(u) = 1$. Given a parsing $\pi = \langle u_1, \ldots, u_t \rangle$ of S, τ satisfies the following relation:

$$\tau(S) \leq \tau(u_1)\tau(u_2)\cdots\tau(u_t),$$

since each $\tau(u_i)$ assumes the prefix γ_i that maximizes the probability that Q emits $u_i|_{\gamma_i}$.

Consider all the strings $u \in A^r$. Each value $\tau(u)$ is $Pr[Q \text{ emits } u|_\gamma]$ for the prefix γ that maximizes it. Let $C(\delta, \gamma) \subseteq A^r$ be the set of strings u prefixed by δ for which $\tau(u)$ chooses to use instead the prefix γ in the maximization. Then

$$\sum_{u \in A^r} \tau(u) = \sum_{\delta, \gamma \in A^k} \sum_{u \in C(\delta,\gamma)} Pr[Q \text{ emits } u|_\gamma].$$

Since the inner sum adds up probabilities of disjoint events, it holds that $\sum_{u \in C(\delta,\gamma)} Pr[Q \text{ emits } u|_\gamma] \leq 1$ and therefore $\sum_{u \in A^r} \tau(u) \leq \sigma^{2k}$.

We can then define a probability distribution p on A^+, that is, $p : A^+ \to [0,1]$ and $\sum_{u \in A^+} p(u) = 1$, such that for every

[2] Markov processes are a subclass of finite state sources. In particular, a k-th order Markov process can be emulated with an $O(\sigma^k)$-th order finite state source. We use the knowledge about the restricted model for deriving more specific bounds.

integer $r > 0$ and every $u \in A^r$, it holds that

$$p(u) = c_k \, \sigma^{-2k} r^{-2} \, \tau(u),$$

where $c_k \geq \frac{1}{2}$ depends only on k. This follows from the previous fact: $\sum_{u \in A^+} p(u) = \sum_{r \geq 1} \sum_{u \in A^r} c_k \, \sigma^{-2k} r^{-2} \, \tau(u) = c_k \sum_{r \geq 1} r^{-2} \sigma^{-2k} \sum_{u \in A^r} \tau(u) \leq c_k \sum_{r \geq 1} r^{-2} < 1.65 \, c_k$, so there is an appropriate constant c_k.

By the information inequality [19, Th. 2.6.3], it holds that

$$H(\pi) = \min_q \sum_{i=1}^{t} \log \frac{1}{q(u_i)} \leq \sum_{i=1}^{t} \log \frac{1}{p(u_i)},$$

where the minimum is taken over all probability distributions q on A^+.

Because $Pr[Q \text{ emits } S] \leq \tau(S) \leq \tau(u_1) \cdots \tau(u_t) \leq \prod_{i=1}^{t} p(u_i) \prod_{i=1}^{t} 2\sigma^{2k} |u_i|^2$, taking the binary logarithm of both sides, multiplying by -1, and using the aforementioned bound for $H(\pi)$, we have

$$\begin{aligned} H(\pi) &\leq \log(1/Pr[Q \text{ emits } S]) \\ &\qquad + t\,(1 + 2k \log \sigma) + 2 \sum_{i=1}^{t} \log |u_i|. \end{aligned}$$

Using the concavity of the logarithm, we bound $\sum_{i=1}^{t} \log |u_i| \leq t \log \frac{S}{t}$, and the result follows. □

So far, the results only require that G_S is a context-free grammar in canonical form, and that it generates (only) S. The final piece of the puzzle is a bound for the ratio $\frac{|G_S|}{|S|}$ that holds when G_S is irreducible:

Lemma 4 [1, cf. Lemma 2]: Given a string S over an alphabet of size σ and an irreducible grammar G_S that generates (only) S, it holds that $\frac{|G_S|}{|S|} = O(1/\log_\sigma |S|)$.

Proof: Kieffer and Yang [1, Lemma 2] prove that $\frac{|G_S|}{|S|} \leq \frac{\sigma}{|S|} + \frac{12 \log \sigma}{\log |S| - 8 \log \sigma - 8}$ holds for any $\sigma \leq |S|^{1/34}$. The bound $\frac{|G_S|}{|S|} = O(1/\log_\sigma |S|)$ also holds if $\sigma > |S|^{1/34}$, since in this case $\log_\sigma |S| \leq 34 = O(1)$, and no irreducible grammar can be asymptotically larger than $|S|$. [3] □

Combining the previous results, we obtain the bound for the binary code $B(G_S)$ resulting from the grammar encoding $\mathcal{G}$, when the underlying grammar G_S is irreducible:

Theorem 1: Given a string S over an alphabet of size $\sigma = O(|S|)$, an irreducible grammar G_S that generates (only) S, and an integer $k \geq 0$, the binary code $B(G_S)$ resulting from the grammar encoding $\mathcal{G}$ satisfies $|B(G_S)| \leq |S| H_k(S) + O\left(|S| \frac{k \log \sigma + \log \log_\sigma |S|}{\log_\sigma |S|}\right)$. If $k \in o(\log_\sigma |S|)$, this simplifies to $|B(G_S)| \leq |S| H_k(S) + o(|S| \log \sigma)$.

Proof: Combining Lemmas 1 to 3, and dividing by $|S|$, we obtain the following relation:

$$\frac{|B(G_S)|}{|S|} \leq H_k(S) + O\left(\frac{\sigma + |G_S| k \log \sigma}{|S|} + \frac{|G_S|}{|S|} \log \frac{|S|}{|G_S|}\right).$$

Using $\sigma = O(|S|)$ and Lemma 4, we then obtain

$$\frac{|B(G_S)|}{|S|} \leq H_k(S) + O\left(\frac{k \log \sigma + \log \log_\sigma |S|}{\log_\sigma |S|}\right),$$

where we used that $\frac{|G_S|}{|S|} \log \frac{|S|}{|G_S|}$ is increasing with $\frac{|G_S|}{|S|}$ for $\frac{|G_S|}{|S|} \leq \frac{1}{e}$. Otherwise, it holds that $\frac{|G_S|}{|S|} = \Theta(1)$, and thus by Lemma 4 it must be that $\log_\sigma |S| = \Theta(1)$; therefore it is also true that $\frac{|G_S|}{|S|} \log \frac{|S|}{|G_S|} = O\left(\frac{\log \log_\sigma |S|}{\log_\sigma |S|}\right)$.

Multiplying by $|S|$, and if $k \in o(\log_\sigma |S|)$, we have

$$|B(G_S)| \leq |S| H_k(S) + o(|S| \log \sigma).$$

□

Note that we need that $\log \sigma = o(\log |S|)$ in order to obtain a meaningful result for $k > 0$. The limitation $k \in o(\log_\sigma |S|)$ is not arbitrary, and appears in Lempel-Ziv based schemes as well [15, Th. A.4]. It is a consequence of the same per-symbol redundancy term that appears in both cases:

$$O\left(\frac{k \log \sigma + \log \log_\sigma |S|}{\log_\sigma |S|}\right).$$

The schemes based on PPM or on the BWT, instead, are more permissive, reaching $H_k(S)$ for any $k \leq (1 - \epsilon) \log_\sigma |S| - 1$, for any constant $0 < \epsilon < 1$ [11, eq. (A34), Th. A4], [13, Th. 12]. This limit is a consequence of the lower redundancy reached in those schemes:

$$O\left(\frac{\sigma^{k+1} \log |S|}{|S|}\right).$$

The difference is especially relevant on finite sequences. Given a concrete string S, its length $|S|$ determines the entropy order up to which different methods can compress it. Alternatively, the redundancy formulas show how well can the finite string be compressed, as a consequence of the rate of convergence of the different methods. The redundancies show that statistical or BWT-based methods can perform better than those based on Lempel-Ziv or on grammar compression, especially on strings whose only source of compressibility are the symbol frequencies (that is, $|S| H_k(S)$ is close to their Kolmogorov complexity). As discussed in the Introduction, Lempel-Ziv or grammar compression may be preferable for other reasons, like speed of decoding or direct text access, but also because they can perform better in strings that are compressible for other reasons, such as high repetitiveness. Consider, for example, a sequence S' formed by t copies of a random sequence S, where S is terminated with a special symbol. For any $k \ll |S|$, it holds that $|S| H_k(S) \approx |S| \log \sigma$ and $|S'| H_k(S') = t |S| H_k(S) \approx t |S| \log \sigma$ [4, Sec. 13.2]. Instead, a Lempel-Ziv compressor can output $|S| H_k(S) + o(|S| \log \sigma) + O(\log(t|S|))$ bits, whereas a grammar compressor can output $|S| H_k(S) + o(|S| \log \sigma) + O(\log(|S| + \log t) \log t)$ bits.

Indeed, the limit $k < \log_\sigma |S| - 1$ is a fundamental one: Gagie [9, Th. 6] proved that, for any constant $0 < \epsilon < 1$, if $\sigma^{k+1-\epsilon} = \Omega(|S|)$, then the Kolmogorov complexity of S exceeds $|S| H_k(S) + \frac{\epsilon}{3} |S| \log \sigma$, for most strings S. In this case, $|S| H_k(S)$ becomes useless as a lower bound. In particular, it becomes impossible to encode S in close to $|S| H_k(S)$ bits.

[3] This limit on σ is just a crude upper bound used to obtain a simple formula. For example, in the proof of [1, Lemma 2], we can show by induction that $N_r(x) \leq 8(r(x) + 1) r(x) \sigma^{r(x)+1}$, thus the bound $N_{r(x)} \leq 16 \, r(x)^2 \sigma^{r(x)+1}$ follows. Using this bound, the relation (9.44) can be substituted by $r(x) - 3 \geq \frac{\log \frac{x}{4} - 2 \log \log \frac{x}{4} - 4 \log \sigma - 4}{\log \sigma}$. If $|S| = x > \sigma^{18}$, then $\log \frac{x}{4} - 2 \log \log \frac{x}{4} - 4 \log \sigma - 4 > 0$, and $r(x) - 3 = \Omega(\log_\sigma x)$.

IV. Conclusion

We have shown that, using a proper encoding [1], a large class of popular grammar compression methods reaches the empirical k-th order entropy of the string they compress, plus a sublinear redundancy. We therefore extend previous results that hold only for information sources, but not for individual sequences [1], and improve and/or simplify previous results [16], [17] that either do not obtain constant 1 multiplying the entropy term, do not hold for all irreducible grammars, and/or require some further adjustments in order to hold for some particular grammars.

The grammars considered are the so-called irreducible grammars, which enforce a few rules to avoid some undesired redundancies in the grammar. Charikar *et al*. [5] show that certain kinds of algorithms always produce irreducible grammars. Kieffer and Yang [1] give a set of rules that can be applied to any context-free grammar G in order to make it irreducible without altering its output. Therefore, every grammar can be transformed into an equivalent irreducible grammar.

A bound in terms of the empirical entropy is stronger than a bound on an information source, because it allows us derive *worst-case* bounds on the size of representations and data structures built on a string, as a function of the compressibility of that particular string. Our result completes a family of bounds for statistical and non-statistical compressors in terms of the empirical entropy, which had already been obtained for compressors based on Prediction by Partial Matching [11], the Burrows-Wheeler Transform [8], [13], and on Lempel-Ziv parsings [15].

References

[1] J. C. Kieffer and E.-H. Yang, “Grammar-based codes: A new class of universal lossless source codes,” *IEEE Trans. Inf. Theory*, vol. 46, no. 3, pp. 737–754, May 2000.

[2] A. Lempel and J. Ziv, “On the complexity of finite sequences,” *IEEE Trans. Inf. Theory*, vol. IT-22, no. 1, pp. 75–81, Jan. 1976.

[3] P. Bille, G. M. Landau, R. Raman, K. Sadakane, S. S. Rao, and O. Weimann, “Random access to grammar-compressed strings and trees,” *SIAM J. Comput.*, vol. 44, no. 3, pp. 513–539, 2015.

[4] G. Navarro, *Compact Data Structures: A Practical Approach*. Cambridge, U.K.: Cambridge Univ. Press, 2016.

[5] M. Charikar *et al.*, “The smallest grammar problem,” *IEEE Trans. Inf. Theory*, vol. 51, no. 7, pp. 2554–2576, Jul. 2005.

[6] N. J. Larsson and A. Moffat, “Off-line dictionary-based compression,” *Proc. IEEE*, vol. 88, no. 11, pp. 1722–1732, Nov. 2000.

[7] A. Apostolico and S. Lonardi, “Off-line compression by greedy textual substitution,” *Proc. IEEE*, vol. 88, no. 11, pp. 1733–1744, Nov. 2000.

[8] G. Manzini, “An analysis of the Burrows—Wheeler transform,” *J. ACM*, vol. 48, no. 3, pp. 407–430, May 2001.

[9] T. Gagie, “Large alphabets and incompressibility,” *Inf. Process. Lett.*, vol. 99, no. 6, pp. 246–251, 2006.

[10] A. N. Kolmogorov, “Three approaches to the quantitative definition of information,” *Int. J. Comput. Math.*, vol. 2, nos. 1–4, pp. 157–168, 1968.

[11] L. Dębowski, “Is natural language a perigraphic process? The theorem about facts and words revisited,” *Entropy*, vol. 20, no. 2, p. 85, 2018.

[12] J. G. Cleary and I. Witten, “Data compression using adaptive coding and partial string matching,” *IEEE Trans. Commun.*, vol. COM-32, no. 4, pp. 396–402, Apr. 1984.

[13] V. Mäkinen and G. Navarro, “Dynamic entropy-compressed sequences and full-text indexes,” *ACM Trans. Algorithms*, vol. 4, no. 3, 2008, Art. no. 32.

[14] M. Burrows and D. Wheeler, “A block sorting lossless data compression algorithm,” Digit. Equip. Corp., Maynard, MA, USA, Tech. Rep. 124, 1994.

[15] S. R. Kosaraju and G. Manzini, “Compression of low entropy strings with Lempel-Ziv algorithms,” *SIAM J. Comput.*, vol. 29, no. 3, pp. 893–911, 1999.

[16] G. Navarro and L. Russo, “Re-pair achieves high-order entropy,” in *Proc. Data Compress. Conf. (DCC)*, 2008, p. 537.

[17] M. Gańczorz. (Apr. 2018). “Entropy bounds for grammar compression.” [Online]. Available: https://arxiv.org/abs/1804.08547

[18] D. A. Huffman, “A method for the construction of minimum-redundancy codes,” *Proc. Inst. Radio Eng.*, vol. 40, no. 9, pp. 1098–1101, Sep. 1952.

[19] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd ed. Hoboken, NJ, USA: Wiley, 2006.

Carlos Ochoa is a PhD student at the Department of Computer Science, University of Chile. His research centers on data compression, algorithms and data structures, and computational geometry. He earned his bachelor degree in computer science from University of Havana, where he was conferred *cum laude*.

Gonzalo Navarro is full professor at the Department of Computer Science, University of Chile. His areas of interest include compression, algorithms and data structures, and text searching. He authored a book on Compact Data Structures, published by Cambridge University Press. He is the Editor in Chief of the *ACM Journal of Experimental Algorithmics* and a member of the Editorial Board of *Information Systems* and *Information Retrieval*. He is the Deputy Director of the Center for Biotechnology and Bioengineering (CeBiB) and an Associate Researcher of the Millennium Institute for Fundamental Research on Data.