Entropy bounds for grammar compression

Michał Gańczorz
Institute of Computer Science, University of Wrocław, Poland
mga@cs.uni.wroc.pl

Abstract
In grammar compression we represent a string as a context free grammar. This model is popular both in theoretical and practical applications due to its simplicity, good compression rate and suitability for processing of the compressed representations. In practice, achieving compression requires encoding such grammar as a binary string, there are a few commonly used. We bound the size of such encodings for several compression methods, along with well-known Re-Pair algorithm. For Re-Pair we prove that its standard encoding, which is a combination of entropy coding and special encoding of a grammar, achieves $1.5|S|H_k(S)$. We also show that by stopping after some iteration we can achieve $|S|H_k(S)$. The latter is particularly important, as it explains the phenomenon observed in practice, that introducing too many nonterminals causes the bit-size to grow. We generalize our approach to other compressions methods like Greedy or wide class of irreducible grammars, and other bit encodings (including naive, which uses fixed-length codes). Our approach not only proves the bounds but also partially explains why Greedy and Re-Pair are much better in practice than the other grammar based methods. At last, we show that for a wide family of dictionary compression methods (including grammar compressors) $\Omega(\frac{n k \log \sigma}{\log n})$ bits of redundancy are required. This shows a separation between context-based/BWT methods and dictionary compression algorithms, as for the former there exists methods where redundancy does not depend on $n$, but only on $k$ and $\sigma$.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases grammar compression, compression, empirical entropy, parsing, RePair

1 Introduction
Grammar compression is a type of dictionary compression, in which we represent the input as a (context free) grammar that produces exactly the input string. Variants of grammar compression achieve competitive compression ratios [27]. Its simple inductive structure makes it particularly suitable for analysis and algorithmic design. Its close ties to Lempel-Ziv type of compression methods makes grammar compression a good abstraction and an intermediate interface for those type of algorithms. Recently, there is a strong trend in algorithmic design to develop algorithms that work directly on the compressed data without the need of a full decompression; among compression methods, grammar compression is particularly suitable for such an approach. Lastly, algorithms for grammar-compressed data can be used in a compress and compute paradigm, in which we compute the grammar compressed representation of data and then process it in this succinct representation, see [28] for a recent survey.

The problem of computing the smallest grammar (in terms of the number of symbols in the productions) is known to be NP-complete [5, 36, 4]. This led to the development of approximation algorithms [33, 35, 22, 35] as well as heuristical algorithms [27, 32, 3, 24, 41, 10]. From the practical point of view, the approximation algorithms have their drawbacks: they achieve only logarithmic approximation, in practice, they are inferior to heuristics, and it seems that minimizing the bit-size and symbol size are not the same thing. On the other hand, heuristics perform comparatively to other dictionary based methods [27]. Note
that heuristical algorithms routinely apply Huffman coding to their output [27, 2]. This is standard also for many other compression methods.

The apparent success of heuristical algorithms fuelled theoretical research that tried to explain or evaluate their performance. One branch of such approach tried to establish their approximation ratios when the size is calculated as the number of symbols [5, 20]. On the other hand, in the spirit of compressors, attempts were made to estimate the bit-size of their output. In this way, grammar compressors could be compared not only among themselves but also with other compressors.

In general, estimating precisely the bit-size of any compressor is hard. However, this is not the case for compressors based on the (higher-order) entropy, which includes Huffman coding, arithmetical coding, PPM, and others. In their case, the bit-size of the output is very close to the \((k\)-order\) empirical entropy of the input string. Thus instead of comparing to those compressors, we can compare the size of the output with the \(k\)-order entropy of the input string. Moreover, in practice, it seems that higher-order entropy is a good estimation of possible compression rate of data and entropy-based compression is widely used. Such analysis was carried out for BWT [29], LZ78 [25], LZ77 [37] compression methods. In some sense, this approach generalised classic results from information theory on LZ-algorithms coding for ergodic sources [42, 44], a similar work was performed also for a large class of grammar compressors [24].

Despite wide popularity of grammar-based methods, not many results that linked their performance to \(k\)-order entropy were known, with the notable exception: Re-Pair was shown to achieve \(2|S|H_k(S) + o(|S| \log \sigma)\) for \(k = o(\log \sigma n)\) [30]. Note that this result holds without the Huffman coding of the output, which is used in practice.

**Our contribution** We start by proving the bounds for Re-Pair [27] and Greedy [2] compressors in terms of \(k\)-order empirical entropy. Then we show that our methods can be generalized for a wide family of so-called irreducible grammars [24]. Our results extend to other grammars that have similar properties. We consider several encodings of the output, which are exactly those (or closely related) used in practice; in particular, we consider the Huffman coding.

The main technical tool is a generalization of result by Kosaraju and Manzini [25, Lemma 2.3], which in turn generalizes Ziv’s Lemma [8, Section 13.5.5]. For any parsing \(Y_S = y_1 y_2 \ldots y_c\) of \(S\):

\[
cH_0(Y_S) \leq |S|H_k(S) + ck \log \sigma + |L|H_0(L),
\]

(1)

where \(L\) is string of lengths \(|y_1| |y_2| \ldots |y_c|\) of consecutive phrases in \(Y_S\). Comparing to [25 Lemma 2.3], our proof is more intuitive, simpler and removes the greatest pitfall of the previous result: the dependency on the highest frequency of a phrase in the parsing. Furthermore, it can be used to estimate the size of the Huffman-coded output, i.e. what is truly used in practice, which was not possible using previously known methods.

Using (1) we show that Re-Pair stopped at the right moment achieves \(|S|H_k(S) + o(|S| \log \sigma)\) bits. Moreover, at this moment the size of the dictionary is \(O(n^c), n = |S|, c < 1,\) where \(c\) depends on the constant in the expression hidden under \(o(\cdot)\). This implies that strings produced by Re-Pair and related methods have a small alphabet. On the other hand, many compression algorithms, like LZ78, do not have this property [13]. Then we prove that in general Re-Pair’s output size can be bounded by \(1.5|S|H_k(S) + o(|S| \log \sigma)\) bits. One of the crucial steps is to use (1) to lower bound the entropy of the string at certain iteration.

Stopping the compressor during its runtime seems counter-intuitive but it is consistent with empirical observations [15, 13] and our results shed some light on the reasons behind
this phenomenon. Furthermore, there are approaches that suggest partial decompression of grammar-compressors in order to achieve better (bit) compression rate [3].

For Greedy we give the same bounds: it achieves $1.5|S|H_k(S) + o(|S|\log \sigma)$ bits using entropy coder, $2|S|H_k(S) + o(|S|\log \sigma)$ without Huffman coding and if stopped after $O(n^c)$ iterations it achieves $|S|H_k(S) + o(|S|\log \sigma)$ bits. The last result seems of practical importance, as each iteration of Greedy requires $O(n)$ times, and so we can reduce the running time from $O(n^2)$ to $O(n^{1+c})$ and should obtain comparable if not better compression rate. No such results were known before.

Then we apply our methods to general class of irreducible grammars [21] and show that Huffman coding of an irreducible grammar uses at most $2|S|H_k(S) + o(|S|\log \sigma)$ bits and at most $6|S|H_k(S) + o(|S|\log \sigma)$ without this encoding. No such general bounds were known before.

In a sense, the upper-bound from [11] can be made constructive: for any $S$ we show how to find a parsing $Y_S$ into phrases of length $l$ such that:

$$|Y_S|H_0(Y_S) \leq |S|\sum_{i=0}^{l-1} H_i(S) + O(\log |S|) .$$

This has direct applications to text encodings with random access based on parsing the string into equal phrases [11] and improves their performance guarantee from $|S|H_k(S) + O(nk \log \sigma/ \log_\sigma n) + o(n) = |S|H_k(S) + o(n \log \sigma)$ to $|S|\sum_{i=0}^{l-1} H_i(S) + O(\log |S|) + o(|S|)$ for $k = o(\log_\sigma n)$.

Finally, we present lower bounds that apply to algorithms that parse the input in a “natural way”, this includes not only considered grammar compressors and compressed text representations [11] [11] [17], but also most of the dictionary compression methods. The main idea is the observation that for some inputs [2] is indeed tight. We construct a family of strings, which can be viewed as a generalization of de Bruijn strings, for which any such algorithm cannot perform better than $\beta|S|H_k(S) + \Omega(nk \log \sigma/ \log_\sigma n)$ in several meanings: the constant at $|S|H_k(S)$ cannot be improved to be lower than 1, the additive term cannot be made smaller, and lifting the assumption that $k = o(\log_\sigma n)$ implies that the coefficient at $|S|H_k(S)$ must be larger than 1. The constructed family of strings has interesting properties in terms of entropy and can be of independent interest.

## 2 Strings and their parsing

A string is a sequence of elements, called letters, from a finite set, called alphabet, and it is denoted as $w = w_1w_2\ldots w_k$, where each $w_i$ is a letter, the length $|w|$ of such a $w$ is $k$; the size of the alphabet is usually denoted as $\sigma$, the alphabet is usually not named explicitly as it is clear from the context or not needed, $\Gamma$ is used when some name is needed. We often analyse words over different alphabets. For any two words $w, w'$ the $ww'$ denotes their concatenation. By $w[i \ldots j]$ we denote $w_iw_{i+1}\ldots w_j$, this is a subword of $w$. By $\epsilon$ we denote the empty word. For a pair of words $v, w$ the $|w|_v$ denotes the number of different (possibly overlapping) subwords of $w$ equal to $v$; if $v = \epsilon$ then for uniformity of presentation we set $|w|_\epsilon := |w|$. Usually $S$ denotes the input string and $n$ its length.

A grammar compression represents an input string $S$ as a context free grammar that generates a unique string $S$. The right-hand side of the start nonterminal is called a starting string (often denoted as $S'$) and by the grammar $(G)$ we mean the collection of other rules, together they are called the full grammar and denoted as $(S', G)$. For a nonterminal $X$ we denote its rule right-hand side by $\text{rhs}(X)$. For a grammar $G$ or full grammar $(S, G)$
their right-hand sides, denoted as \( \text{rhs}(G) \) and \( \text{rhs}(S, G) \), are the concatenations of strings that are the right-hand sides of all productions in \( G \), for \( (S, G) \) we also concatenate \( S \). By \( |\mathcal{N}(G)| \) we denote the number of nonterminals. If all right-hand sides of grammar consist of two symbols, then this grammar is in CNF. The expansion \( \exp(X) \) of a nonterminal \( X \) in a grammar \( G \) is the string generated by \( X \) in this grammar. All reasonable grammar compressions guarantee that no two nonterminals have the same expansion, we implicitly assume this in the rest of the paper. We say that a grammar (full grammar) \( G \) \((S, G)\), respectively) is small, if \( |\text{rhs}(G)| = O\left(\frac{n}{\log_2 n}\right) \) \( (|\text{rhs}(S, G)| = O\left(\frac{n}{\log_2 n}\right) \), respectively. This matches the folklore information-theoretic lower bound on the size of the grammar for a string of length \( n \).

In practice, the starting string and the grammar may be encoded in different ways, especially when \( G \) is in CNF, hence we make a distinction between these two. Note that for many (though not all) grammar compressors both theoretical considerations and proofs as well as practical evaluation show that the size of the grammar is considerably smaller than the size of the starting string.

A parsing of a string \( S \) is any representation \( S = y_1 y_2 \cdots y_c \), where each \( y_i \) is nonempty and is called a phrase. We denote a parsing as \( Y_S = y_1, \ldots, y_c \) and treat it as a word of length \( c \) over the alphabet \( \{y_1, \ldots, y_c\} \); in particular \( |Y_S| = c \) is its size. Then \( \text{Lengths}(Y_S) = |y_1|, |y_2|, \ldots, |y_c| \in \mathbb{N}^* \) and we treat it as a word over the alphabet \( \{1, 2, \ldots, n\} \).

The idea of parsing is strictly related to dictionary compression method, as most of this methods pick some substring and replace it with a new symbol, thus creating a phrase in a parsing. Examples include Lempel-Ziv algorithms. In grammar methods, which are a special case of dictionary compression, parsing is often induced by starting string.

Given a word \( w \) its \( k \)-order empirical entropy is

\[
H_k(w) = -\frac{1}{|w|} \sum_{v: |v|=k} \log \frac{|w|_v}{|w|},
\]

with the convention that the summand is 0 whenever \( |w|_v = 0 \) or \( |w|_v = 0 \). We are mostly interested in the \( H_k \) entropy of the input string \( S \) and in the \( H_0(Y_S) \) for parsing \( Y_S \) of \( S \).

The former is a natural measure of the input, to which we shall compare the size of the output of the grammar compressor, and the latter corresponds to the size of the entropy coding of the starting string returned by the grammar compressor.

3 Entropy bound for string parsing

In this Section, we make a connection between the entropy of the parsing of a string \( S \), i.e. \( |Y_S|H_0(Y_S) \) and the \( k \)-order empirical entropy \( |S|H_k(S) \); this can be seen as a refinement and strengthening of results that relate \( |Y_S| \) and \( |S|H_k(S) \) to \( H_k(S) \) \cite{25} Lemma 2.3], i.e. our result establishes upper bounds when phrases of \( Y_S \) are encoded using entropy coder while previous one use trivial encoding of \( Y_S \), which assigns to each letter \( \log |Y_S| \) bits.

Theorem 1 yields that entropy of any parsing is bounded by \( H_k \) plus some additional summands, which depend on the size of the parsing and entropy of lengths of the parsing. In particular, it eliminates the main drawback of previous results \cite{25} Lemma 2.3], which also had a dependency on the frequency of the most frequent phrase.

\[\text{Theorem 1 (cf. \cite{25} Lemma 2.3)} \] Let \( S, |S| = n \) be a string, \( Y_S \) its parsing, and \( L = \text{Lengths}(Y_S) \). Then: \( |Y_S|H_0(Y_S) \leq |S|H_k(S) + |Y_S|k \log \sigma + |L|H_0(L) \). Moreover, if \( k = o(\log \sigma n) \) and \( |Y_S| \leq \frac{\alpha n}{\log_2 n} \) for some constant \( \alpha \) then: \( |Y_S|H_0(Y_S) \leq |S|H_k(S) + \)
Lemma 3. Theorem 2 implies that any parsing $|Y_S|H_0(Y_S)$ is within $|L|H_0(L)$ summand of $|S|H_k(S)$. This also gives upper bound on entropy increase of any parsing. Interestingly, the second bound holds for any parsing $Y_S$, assuming it is small enough.

When we can choose the parsing, the upper bound can be improved, even when we are restricted to parsings with phrases of (almost) fixed phrase length.

**Theorem 2.** Let $S$ be a string over alphabet $\sigma$. Then for any integer $l$ we can construct a parsing $Y_S$ of size $|Y_S| \leq \left\lceil \frac{|S|}{l} \right\rceil + 1$ satisfying: $|Y_S|H_0(Y_S) \leq |S|\sum_{k=1}^{l-1} H_k(S) + O(|\log|S||).

All phrases except the first and last one have length $l$.

Note that Theorem 2, unlike Theorem 1, does not hold for every parsing, it only claims that a carefully chosen parsing can have smaller entropy than a naive one.

The proofs follow a couple of simple steps. First, we recall a strengthening of the known fact that entropy lower-bounds the size of the encoding of the string using any prefix codes: instead of assigning natural lengths (of codes) to letters of $Y_S$ we can assign them some probabilities (and think that $- \log(p)$ is the “length” of the prefix code). Then we define the probabilities of phrases in the parsing $Y_S$ in such way that they relate to higher order entropies. Depending on the result we either look at fixed, $k$-letter context, or $(i-1)$-letter context for $i$-th letter of phrase. This already yields the first claim of Theorem 3 to obtain the second we substitute the estimation on the parsing size and make some basic estimation on the entropy of the lengths, which is a textbook knowledge [8 Lemma 13.5.4].

**Entropy estimation** The following technical Lemma strengthens the well-known fact that entropy lower-bounds the size of the prefix code encoding, it is a simple corollary from Gibbs’ inequality, see [1] for a proof.

**Lemma 3 ([1]).** Let $w$ be a string over alphabet $\Gamma$ and $p : \Gamma \to \mathbb{R}^+$ be a function such that $\sum_{s \in \Gamma} p(s) \leq 1$. Then: $|w|H_0(w) \leq - \sum_{s \in \Gamma} |w_s| \log p(s)$.  

To use Lemma 3 to prove Theorems 1, 2 we need to devise appropriate valuation $p$ for phrases of the parsing. Instead of assigning single $p$ value to each phrase we assign it to each individual letter in each phrase, then $p(y)$ is a product of values of consecutive letters of $y$.

In the case of Theorem 2 for $j$-th letter of a phrase $y$ we assign the probability of this letter occurring in $j-1$ letter context in $S$, i.e. we assign $\frac{|S_{|j-1|}}{|S|}$. Thus we can think that we encode the letter using $(j-1)$-th order entropy. The difference in the case of Theorem 2 is that we assign to first $k$ letters of the phrase, the remaining ones are assigned values as in the first case. The phrase costs are simply logarithmed values of phrase probabilities, which can be viewed as a cost of bit-encoding of a phrase.

The idea of assigning values to phrases was used before, for instance by Kosaraju and Manzini in estimations of entropy of LZ77 and LZ78 [25], yet their definition depends on $k$ symbols preceding the phrase. This idea was adapted from methods used to estimate entropy of the source model [12], see also [8 Sec. 13.5]. Similar idea was used in [14] in construction of compressed text representations. Their solution used constructive argument:
Entropy bounds for grammar compression

it calculated bit strings for each phrase using arithmetic coding and context modeller. Later it was observed that arithmetic coder and context modeller can be replaced with Huffman encoding [11]. Still, both these representations were based on assumption that text is parsed into short (i.e \( \alpha \log \sigma n, \alpha < 1 \)) phrases of equal length and used specific compression methods in the proof.

Definition 4 (Phrase probability, parsing cost). Given a string \( S \) and its parsing \( Y_S = y_1 y_2 \ldots y_c \), the phrase probability \( \mathcal{P}(y_i) \) and \( k \)-bounded phrase probability are:

\[
\mathcal{P}(y_i) = \prod_{j=1}^{y_i} \frac{|S|_{y_i[j-1:j]} |S|_{y_i[j+1:i-1]}}{|S|_{y_i[j-1:j]} |S|_{y_i[j+1:i-1]}} \quad \text{and} \quad \mathcal{P}_k(y_i) = \frac{1}{\alpha \min(|y_i|, k)} \prod_{j=k+1}^{y_i} \frac{|S|_{y_i[j-k:j]} |S|_{y_i[j+k:j+1]}}{|S|_{y_i[j-k:j]} |S|_{y_i[j+k:j+1]}}
\]

where those are 0 if, respectively, some \( |S|_{y_i[1:j]} = 0 \) or \( |S|_{y_i[j-k:j]} = 0 \). Observe that the definition also holds for \( k = 0 \), as we assumed that \( w[i \ldots j] = \epsilon \) when \( i > j \), and \( |S|_c = |S| \).

The phrase cost and parsing cost are \( C(y_i) = -\log \mathcal{P}(y_i) \) and \( C(Y_S) = \sum_{i=1}^{|Y_S|} C(y_i) \). Similarly the \( k \)-bounded phrase cost and \( k \)-bounded parsing cost are: \( C_k(y_i) = -\log \mathcal{P}_k(y_i) \) and \( C_k(Y_S) = \sum_{i=1}^{|Y_S|} C_k(y_i) \).

When comparing the \( C_k(Y_S) \) cost and \( |S|H_k(S) \), the latter always uses \( H_k \) entropy for each symbol, while \( C_k(Y_S) \) uses \( \log \sigma \) on each first \( k \) letters of each phrase, thus intuitively it losses up to \( \log \sigma \) on each of those \( |Y_S|k \) letters.

Lemma 5. Let \( S \) be a string and \( Y_S \) its parsing. Then \( C_k(Y_S) \leq |S|H_k(S) + |Y_S|k \log \sigma \).

Lemma 6. Let \( S \) be a string. Then for any \( l \) there exist a parsing \( Y_S \leq \frac{|S|}{l} + 1 \) such that each phrase except first and last has length exactly \( l \) and \( C(Y_S) \leq |S| H(S) + \log |S| \).

Parsings and entropy Ideally, we would like to plug-in the phrase probabilities for \( Y_S \) into Lemma 3 and so obtain that the parsing cost \( C(Y_S) \) upper-bounds entropy encoding of \( Y_S \), i.e. \( |Y_S|H_0(Y_S) \). But the assumption of Lemma 3 (that the values of function \( p \) sum to at most 1) may not hold as we can have phrases of different lengths and so their probabilities can somehow mix. Thus we also take into the account the lengths of the phrases: we multiply the phrase probability by the probability of \( |y_i| \), i.e. the frequency of \( |y_i| \) in \( \text{Lengths}(Y_S) \). After simple calculations, we conclude that \( |Y_S|H_0(Y_S) \) is upper bounded by the parsing cost \( C(Y_S) \) plus the entropy of lengths: i.e. when \( L = \text{Lengths}(Y_S) \), the \( |L|H_0(L) \).

Theorem 7. Let \( S \) be a string over \( \sigma \)-size alphabet, \( Y_S \) its parsing, \( c = |Y_S| \) and \( L = \text{Lengths}(Y_S) \). Then:

\[
|Y_S|H_0(Y_S) \leq C(Y_S) + |L|H_0(L) \quad \text{and} \quad |Y_S|H_0(Y_S) \leq C_k(Y_S) + |L|H_0(L).
\]

When \( S \in \alpha^* \) (and so \( C(S) = C_k(S) = H_0(S) = 0 \)) the entropy \( |Y_S|H_0(Y_S) \) is the entropy of \( |L|H_0(L) \), thus summation \( |L|H_0(L) \) on the right-hand is necessary.

Now, the combination of above claims gives directly the proof of Theorem 1 and Theorem 2 see Appendix for the full proof. The entropy of lengths is always within \( \mathcal{O}(|S|) \) factor, and for small enough \( |Y_S| \) within \( \mathcal{O}(n \log \log \sigma n/\log \sigma n) \) factor; moreover in case of Theorem 2 it can be bounded by \( \mathcal{O}(\log |S|) \). This follows textbook arguments [8, Lemma 13.5.4].
4 Entropy upper bounds on grammar compression

In this Section we use Theorem 1 to bound the size of grammars returned by popular compressors: Re-Pair \cite{27}, Greedy \cite{2} and a general class of methods that produce irreducible grammars. We consider a couple of natural and simple bit-encodings of the grammars, those include naive encoding (which takes $\lceil \log(|N(G)| + \sigma) \rceil$ bits per grammar symbol), Huffman coding and so-called incremental coding, which is popular for grammars in CNF. Interestingly, we obtain bounds $\alpha |S| H_k(S) + o(n \log \sigma)$ for a constant $\alpha$ even for naive encoding.

4.1 Encoding of grammars

There are different ways to encode the (full) grammar thus we first discuss the possible encodings and give some estimations on their sizes. All considered encoding assume a linear order on nonterminals: if rhs($X$) contains $Y$ then $X \geq Y$. In this way we can encode the rule as a sequence of nonterminals on its right-hand side, in particular instead of storing the nonterminal names we store positions in the above ordering.

The considered encodings are simple and natural and they correspond very closely or exactly to encodings used in grammars compressors like Re-Pair \cite{27} or Sequitur \cite{31}. Some other algorithms, e.g. Greedy \cite{2}, use specialized encodings, but at some point, they still encode grammar using entropy coder with some additional information or assign codes to each nonterminal in the grammar. Thus most of this custom encodings are roughly equivalent to (or not better than) entropy coding.

Encoding of CNF grammars deserves special attention and is a problem investigated on its own. It is known that grammar $G$ in CNF can be encoded using $|N(G)| \log(|N(G)|) + 2|N(G)| + o(|N(G)|)$ bits \cite{38, 39}, which is close to the information theoretic lower bound of $\log(|N(G)|)! + 2|N(G)|$ \cite{38}. On the other hand, in heuristical compressors simpler encodings are used, for instance, Re-Pair was implemented and tested with several encodings, which were based on division of nonterminals $G$ into $z$ groups $g_1, \ldots, g_z$ where $X \rightarrow AB \in g_i$ if and only if $A \in g_{i-1}$ and $B \in g_j$ or $B \in g_{i-1}$ and $B \in g_j$, for some $j \leq i - 1$, where $g_0$ is the input alphabet. Then each group is encoded separately. Even though no theoretic bounds were given, these encodings come close to the lower bound, though some only on average.

The above encodings are difficult to analyse due to heuristical optimisations, for the sake of completeness we analyse an incremental encoding, which is a simplified version of one of the original methods used to encode Re-Pair output. It matches the theoretical lower bound except a larger constant hidden in $O(|N(G)|)$.

To be precise, we consider the following encodings:

**fully naive** We concatenate the right-hand sides of the full grammar. Then each nonterminal and letter are assigned bitcodes of the same length. We store $|\text{rhs}(X)|$ for each $X$, as it is often small, it is sufficient to store it in unary.

**naive** The starting string is entropy-coded, the rules are coded as in the fully-naive variant.

**entropy-coded** The rules are concatenated with the starting string and they are coded using an entropy coder. We also store $|\text{rhs}(X)|$ for each nonterminal $X$.

**incremental** We use this encoding only for CNF grammars, though it can be extended to general grammars. It has additional requirements on the order on letters and nonterminals: if $X \leq Y$ and $X', Y'$ are the first symbols in productions for $X, Y$ then $X' \leq Y'$. Given any grammar, this property can be achieved by permuting the nonterminals, but we must drop the assumption that right hand side of a given nonterminal $X$ occurs before $X$ in the sequence. Then grammar $G$ can be viewed as a sequence of nonterminals:
Lemma 8. Let $S$ be a string and $(S', G)$ a full grammar that generates it. Then:
- fully naive uses at most $|\text{rhs}(S', G)| (\log(\sigma + |N(G)|) + O(|\text{rhs}(S', G)|))$ bits.
- naive uses at most $|S'|H_0(S') + |\text{rhs}(G)| \log(\sigma + |N(G)|) + O(|\text{rhs}(S', G)|)$ bits.
- incremental uses at most $|S'|H_0(S') + |N(G)| \log(\sigma + |N(G)|) + O(|\text{rhs}(S', G)|)$ bits.

The proof idea of Lemma 8 is to show that $|\text{rhs}(S', G)|$ is a parsing of $S$, with at most $|N(G)|$ additional symbols, and then apply Theorem 1. The latter requires that different nonterminals have different expansions, all practical grammar compressors have this property.

Lemma 9. Let $S$ be a string over an alphabet of size $\sigma$, $k = o(\log_{\sigma} n)$ and $(S', G)$ a full grammar generating it, where no two nonterminals have the same expansion. Denote $S_G = \text{rhs}(S', G)$. If $|S_G| = O(n/\log_{\sigma} n)$ then the entropy coding of $(S', G)$ is $|S_G|H_0(S_G) \leq |S|H_k(S) + |N(G)| \log |S| + o(|S| \log \sigma)$.

4.2 Re-Pair

Re-Pair is one of the most known grammar compression heuristics. It starts with the input string $S$ and in each step replaces a most frequent pair $AB$ in $S$ with a new symbol $X$ and adds a rule $X \rightarrow AB$. Re-Pair is simple, fast and provides compression ratio better than some of the standard dictionary methods like gzip [27]. It found usage in various applications [16, 23, 3, 12, 41, 17].

We prove that Re-Pair stopped at right moment achieves $H_k$ entropy (plus some smaller terms), using any of the three: naive, incremental or entropy encoding. To this end we show that Re-Pair reduces the input string to length $\frac{\alpha n}{\log_{\sigma} n}$ for appropriate $\alpha$ and stop the algorithm when the string gets below this size. This follows by estimations on number of possible different substrings in the input string. Then on one hand the grammar constructed so far is of size $n^{c} = o(n)$, $c < 1$ and on the other side Theorem 1 yields that entropy coding of the current string is $|S|H_k(S)$ plus some smaller terms. This property gives an advantage over other methods, as it ensures small alphabet size of the string to encode and in practice encoding of a large dictionary is costly. Even though the Theorem 1 states explicitly the values of $\alpha$ and $c$, one is function of the other, see proofs in the Appendix.

We refer to the current state of $S$, i.e. $S$ with some pairs replaced, as the working string.

Theorem 10. Let $S$, $|S| = n$, be a string over $\sigma$-size alphabet, $k = o(\log_{\sigma} n)$. When the size of the working string of Re-Pair is first below $\frac{16n}{\log_{\sigma} n}$ then the number of nonterminals in the grammar is at most $\sqrt{n \log_{\sigma} n}$ and the entropy coding of the working string is at most $|S|H_k(S) + o(|S| \log \sigma)$; such a point always exists and the bit-size of Re-Pair stopped at this point is at most $|S|H_k(S) + o(|S| \log \sigma)$ for: naive, entropy and incremental encoding.

Theorem 10 says that Re-Pair achieves $H_k$-entropy, when stopped at the appropriate time. What is surprising is that continuing to the end can lead to worse compression rate. In fact, limiting the size of the dictionary for Re-Pair as well as for similar methods in practice results in worse compression.
in better compression rate for larger files [15, 13]. This is not obvious, in particular, Larsson and Moffat [27] believed that it is the other way around; this belief was supported by the results on smaller-size data. This is partially explained by Theorem 12, in which we give a \(1.5|S|H_k(S) + o(|S|\log \sigma)\) bound on Re-Pair run to the end with incremental or entropy encoding; a \(2|S|H_k(S) + o(|S|\log \sigma)\) bound for fully naive encoding was known earlier [30].

Before proving Theorem 12 we first show a simple example, see Lemma 11, that demonstrates that the 1.5 factor from Theorem 12 is tight, assuming certain encodings, even for \(k = 0\). The construction employs large alphabets, i.e. \(\sigma = \Theta(|S|)\). Observe that our results assume that \(k = o(\log \sigma \cdot n)\), which implies that for polynomial alphabets they hold only for \(k = 0\). Also, such large alphabets do not reflect practical cases when \(\sigma\) is much smaller than \(|S|\). Yet, in case of grammar compression this example gives some valuable intuition: replacing the substring \(w\) decreases the size counted in symbols but may not always decrease bit encoding size, as we have to store some additional information, regarding replaced string, which can be costly, depending on the encoding method.

**Lemma 11.** There exist a family of strings \(S\) such that Re-Pair with both incremental and entropy encoding uses at least \(\frac{1}{2}|S|H_0(S) - o(|S|\log \sigma)\) bits, assuming that we encode the grammar of size \(g\) (i.e. with \(g\) nonterminals) using at least \(g\log g - O(g)\) bits. Moreover \(|S|H_0(S) = \Omega(|S|\log \sigma)\), which implies that the cost of encoding, denoted by Re-Pair\((S)\), satisfies
\[
\limsup_{|S| \to \infty} \frac{\text{Re-Pair}(S)}{|S|H_0(S)} \geq \frac{1}{2}.
\]

**Proof of Lemma 11.** Fix \(n\) and an alphabet \(\Gamma = \{a_1, a_2, \ldots, a_n, \#\}\). Consider the word \(S\) which contains all letters \(a_i\) with \# in-between, first in order from 1 to \(n\), then in order \(n\) to 1: \(S = a_1\#a_2\#a_3\# \cdots a_{n-1}\#a_n\#a_1\#a_2\# \cdots a_2\#a_1\#\).

Detailed calculations are provided in the Appendix.

The example from Lemma 11 shows that at some iteration bit encoding of Re-Pair can increase. Even though it requires large alphabet and is somehow artificial, we cannot ensure that a similar instance does not occur at some iteration of Re-Pair, as the size of the alphabet of the working string increases. In the above example size of the grammar was significant. It is the only possibility to increase bit size as by Theorem 1 adding new symbols does not increase entropy encoding of working string significantly.

Main observation needed for the proof of Theorem 12 is that if the size that if at some point the grammar size is significant, then the entropy of the working string is also large. In such a case the grammar transformations do not increase overall cost of encodings too much. The crucial element in this reasoning is the usage of Theorem 1 to lower-bound the \(k\)-entropy of the input string, showing that entropy at desired point is indeed large.

**Theorem 12.** Let \(|S| = n\) be a string over a \(\sigma\)-size alphabet, \(k = o(\log \sigma \cdot n)\). Then the size of Re-Pair output is at most \(\frac{3}{2}|S|H_k(S) + o(|S|\log \sigma)\) for the incremental and the entropy encoding.

### 4.3 Irreducible grammars and their properties

Kieffer et al. introduced the concept of irreducible grammars [24], which formalise the idea that there is no immediate way to make the grammar smaller. Many heuristics fall into this category: Sequential, Sequitur, LongestMatch, and Greedy, even though some were invented before the notion was introduced [5]. They also developed an encoding of such grammars which was used as a universal code for finite state sources.

**Definition 13.** A full grammar \((S, G)\) is irreducible if:
(IG1) no two nonterminals have the same expansion;
(IG2) every nonterminal, except the starting symbol, occurs at least twice in rhs(S, G);
(IG3) no pair occurs twice (without overlaps) in S and G right-hand sides.

Unfortunately, most irreducible grammar has the same issue as Re-Pair: they can introduce new symbols without decreasing entropy of the starting string but increasing bit size of the grammar. In particular, the example from Lemma 14 applies to irreducible grammars (i.e. any irreducible grammar generating S have at least \( \Omega(|S|) \) nonterminals).

Ideally, for an irreducible grammar we would like to repeat the argument as for Re-Pair: we can stop at some of the iteration (or decompress some nonterminals as in [3]) such that the grammar is small and has a small, i.e. \( \mathcal{O}(n^c) \), number of nonterminals. It turns out that there are examples of irreducible grammars which do not have this property, Example 14 gives such a grammar. Moreover, grammar compressors that work in a top-down manner, like LongestMatch, tend to produce such grammars. Lastly, the grammar in Example 14 has size \( \mathcal{O}\left(\frac{n}{\log \sigma \cdot n}\right) \), which is the best possible estimation for the size of irreducible grammars.

◮ Example 14. Consider the grammar where each production represents binary string: \( S' \rightarrow X_{000}X_{000}X_{001}X_{001} \ldots X_{111}X_{111} ; X_{000} \rightarrow X_{000} 0 ; X_{001} \rightarrow X_{001} 1 ; X_{00} \rightarrow X_{00} 0 ; X_{010} \rightarrow X_{010} ; \ldots \)

The above example can be generalized for binary string of any length. Decompressing any set of nonterminals such that only \( \mathcal{O}(n^c) \) nonterminals remain yields a grammar of size \( \omega\left(\frac{n}{\log \sigma \cdot n}\right) \).

Still, we are able to prove positive results assuming our encodings, though with worse constant. Similarly, as in the proof of Theorem 12 we use Theorem 1 to lower bound the entropy of rhs \( (S', G) \) in the naive case, thus it seems, that using only previously known tools [25] such bounds could not be obtained.

◮ Theorem 15. Let \( S \) be a string over \( \sigma \)-size alphabet, \( k = o(\log \sigma \cdot n) \) Then the size of the entropy coding of any irreducible full grammar generating \( S \) is at most \( 2|S|h_k(S) + o(|S| \log \sigma) \).

◮ Theorem 16. Let \( S \) be a string over an alphabet of size \( \sigma \), \( k = o(\log \sigma \cdot n) \). The size of fully naive coding of any irreducible full grammar generating \( S \) is at most \( 6|S|h_k(S) + o(|S| \log \sigma) \).

4.4 Greedy

Greedy [2] can be viewed as a non-binary Re-Pair: in each round it replaces a substring in \( (S, G) \), obtaining \( (S', G') \), such that \( |\text{rhs}(S', G')| \) is smallest possible. It is known to produce small grammars in practice, both in terms of nonterminal size and bit size. Its asymptotic construction time so far has been only bounded by \( \mathcal{O}(n^2) \). Moreover, it is notorious for being hard to analyse in terms of approximation ratio [3, 20].

Greedy has similar properties as Re-Pair: the frequency of the most frequent pair does not decrease and the size of the grammar can be estimated in terms of this frequency. In particular, there always is a point of its execution in which the number of nonterminals is \( \mathcal{O}(n^c) \) and the full grammar is of size \( \mathcal{O}(n/\log \sigma \cdot n) \). The entropy encoding at this time yields \( |S|h_k(S) + o(|S| \log \sigma) \), while Greedy run till the end achieves \( \frac{1}{2}|S|h_k(S) \) using entropy coding and \( 2|S|h_k(S) \) using fully naive encoding, so the same as in the case of Re-Pair.

In practice stopping Greedy is beneficial: similarly as in Re-Pair there can exist a point where we add new symbols that do not decrease the bitsize of output. Indeed, it was suggested [2] to stop Greedy as soon as the decrease of grammar size is small enough, yet this was not experimentally evaluated. Moreover, as the time needed for one iteration is
Theorem 17. Let $S$, $|S| = n$ be a string over $\sigma$-size alphabet, $k = o(\log_\sigma n)$. When the size of the full grammar $(S', G)$ (i.e., $|S'| + |\text{rhs}(G)|$) produced by Greedy is first below $\frac{64k}{\log_\sigma n}$ then the number of nonterminals $|N(G)|$ of $G$ is at most $\sqrt{n \log_\sigma n} + 3$ and the bit-size of entropy coding of $(S', G)$ is at most $|S|H_k(S) + o(|S| \log \sigma)$. For any string $S$ such point always exists.

Theorem 18 (cf. [30]). Let $S$, $|S| = n$ be a string over $\sigma$-size alphabet, $k = o(\log_\sigma n)$. Then the size of the fully naive encoding of full grammar produced by Greedy is at most $2|S|H_k(S) + o(|S| \log \sigma)$.

Theorem 19. Let $S$, $|S| = n$ be a string over $\sigma$-size alphabet, $k = o(\log_\sigma n)$. Then the size of entropy encoding of full grammar produced by Greedy is at most $\frac{4}{3} |S|H_k(S) + o(|S| \log \sigma)$.

5 Lower bound on parsing-based methods

All upper bounds presented in previous sections assumed $k = o(\log_\sigma n)$ and have an additive term $o(n \log \sigma)$. In this section, we show that both are unavoidable. To this end, we construct a family of strings for which the bounds of Theorem 2 are tight and explain how this implies that the conditions on $k$ and the additive term cannot be strengthened. This in particular answers (negatively) question from [30], whether we can prove similar bounds for Re-Pair when $k = a \log_\sigma n$, $a < 1$.

Generalized de Bruijn words The constructed family of words generalize de Bruijn strings, which, for a given alphabet $\Gamma$ and order $k$, contain exactly once each word $w \in \Gamma^k$ as a substring.

Theorem 20. For every $k > 0$, $l \geq 0$, $p \geq 1$ there exists a string $S$ over alphabet of size $\sigma = 4^p$ of length $\sigma^{k+l+1}$ such that:
1. $\log \sigma - O(\frac{i \log |S|}{|S|}) \leq H_i(S) \leq \log \sigma$ for $i < k$;
2. $\log \sigma - O(\frac{i \log |S|}{|S|}) \leq H_i(S) \leq \frac{\log \sigma}{2}$ for $k \leq i \leq k + l$;
3. no word of length $k + l + 1$ occurs more than once in $S$.

For $l = 0$ the promised family are constructed from de Bruijn strings by appropriate letter merges. For those strings the frequency of each substring depends (almost) only on its length, thus the bounds on the entropy are easy to show. For larger $l$ we make an inductive (on $l$) construction, which is similar to construction of de Bruijn strings: we construct a graph with edges labelled with letters and the desired strings corresponds to an Eulerian cycle in this graph, to be more precise, the $(l+1)$st graph is exactly the line graph of the $l$th one. We guarantee that the frequency of words depends only on their lengths, the exact condition is more involved than in case of de Bruijn strings.

Example 21. For $\sigma = 4$, $k = 2$, $l = 0$: $S = aababbbadcdbdcaaadaccbdbbceddc$. For $\sigma = 4$, $k = 1$, $l = 1$ the word is $S = abbdadcacabdc$. 

Natural parsers. The sequence of strings from Theorem 20 are now used to show lower bounds on the size of parsings produced by various algorithms. Clearly, the lower bounds cannot apply to all parsings, as one can take a parsing into a single phrase. Thus we consider the “natural” parsings, in which a word can be made a phrase if it occurs twice or is short.

Definition 22. An algorithm is a natural-parser if given a string $S$ over an alphabet of size $\sigma$ it produces its parsing $Y_S$ such that for each phrase $y = wa$, $|a| = 1$ of $Y_S$ either $|S|_w > 1$, or $y \leq \log_\sigma |S|$; moreover, it encodes $Y_S$ using at least $|Y_S|H_0(Y_S)$ bits.

Note that phrases of length 1 that occur once are allowed, as for them $w = \epsilon$ and $|S|_w = |S| > 1$.

Lemma 23. Re-Pair, algorithms producing irreducible grammars, LZ78 and non self-referencing LZ77 (with appropriate encoding) are natural parsers.

Natural parsers on words defined in Theorem 20 cannot do much better than the mean of entropies, which gives general bounds on algorithms inducing natural parsers.

Theorem 24. Let $A$ be a natural parser. Let $k$ be a non-negative and integer function of $|S|$ and $\sigma$ satisfying, for every $\sigma$, $\limsup_{|S|\rightarrow \infty} \mathbb{E}_{|S|,\sigma} k_{|S|,\sigma} < 1$, where $k_{|S|,\sigma}$ denotes value of $k$ for $|S|$ and $\sigma$. Then for any $\rho > 0$ there exist infinite family of strings $S \in \Gamma^*$, where $|\Gamma| = 4^\rho$, of increasing length, such that the bit-size of the output of $A$ on $S$ is at least:

$$|S|H_k(S) + \frac{\rho |S|(\log \sigma - 2\lambda)}{2} \geq (1 + \rho) |S|H_k(S) - \lambda |S|,$$

where $\rho = \frac{k}{2\log_\sigma |S| - k}$ and $\lambda < 0.54$.

There are several consequences of Theorem 24 for natural parsers. First, they cannot go below $|S|H_k(S)$ bits on each string and if they achieve the entropy (on each string) then an additive term of $\Theta(nk\log \sigma / \log_\sigma n)$ bits is needed.

Corollary 25. Let $k$ be a function of $(n, \sigma)$, where $k = o(\log_\sigma n)$, and $A$ be a natural parsing algorithm. Then there exist an infinite family of strings of increasing length such that for each $S$, $|S| = n$, the size of the output generated by $A$ on $S$ is at least: $|S|H_k(S) + \Omega(\frac{n\log \sigma}{\log_\sigma n})$.

Secondly, extending the bounds to $k = \alpha \log_\sigma n$ for a constant $0 < \alpha < 1$ implies that $|S|H_k(S)$ (without a constant coefficient) is not achievable. This gives (negative) answer to the question asked in [30] whether we can prove results for Re-Pair when $k = \alpha \log_\sigma n$.

Corollary 26. Let $k$ be a function of $(n, \sigma)$ such that $k = \alpha \log_\sigma n$, $0 < \alpha < 1$, and $A$ be a natural parsing based algorithm. Then there exist an infinite family of strings of increasing length such that for each $S$, $|S| = n$, if $A$ achieves $\beta |S|H_k(S) + o(|S| \log \sigma)$ bits then $\beta \geq \frac{2}{2 - \alpha}$.

Lastly, Theorem 2 is tight for natural parsers.

Corollary 27. For any $j$ there exist an infinite family of strings $S$ such that if $j < \log_\sigma |S| - 2$ then no parsing $Y_S$ with phrases shorter than $\log_\sigma |S|$ achieves $|Y_S|H_0(Y_S) \leq (1 - \epsilon) \frac{|S|}{j} \sum_{i=0}^{j-1} H_i(S) + o(|S| \log \sigma)$, for $\epsilon > 0$.

6 Conclusions and open problems

The lower bounds provided in Section 5 hold for specific types of algorithms. Yet, there are algorithms achieving $|S|H_k(S) + o(n)$ bits for $k = \alpha \log_\sigma n$ where $\alpha < 1$, e.g. the ones based on
Moreover, k-order PPM-based methods should also encode words defined above efficiently, as many of them use adaptive arithmetic coding for each context separately. Can we generalize the techniques so that they provide some bounds also for those scenarios?

There are parsing based compressed text representations \[18\] achieving \(|S|H_k(S) + o(n)\) for \(k = \alpha \log_\sigma n\), where \(\alpha < \frac{1}{k}\), but they encode parsing using 1-order entropy coders. This comes at a cost, as such representations do not allow for retrieval of substrings of length \(\Theta(\log_\sigma n)\) in constant time, which is possible for indexes based on parsings and using 0-order entropy coding \[14, 11\]. Can we estimate the time-space tradeoffs?

We considered bounds \(\beta|S|H_k(S) + f(|S|, \sigma)\), where \(f(|S|, \sigma) = o(n \log \sigma)\). Kosaraju and Manzini \[25\] considered also the stronger notion of coarse optimality, in which they require that \(f(|S|, \sigma) = o(|S|H_k(S))\). They developed coarse optimal algorithms only for \(k = 0\). It is not known if similar results can be obtained for grammar compression, though our lower bounds provide some insights. On one hand, there are examples of small entropy strings on which most grammar compressors perform badly \[5, 19\], but there exceptions, e.g. Greedy.

It should be possible to extend construction of words from Theorem \[20\] such that for any constant \(\beta = 2^j, j > 0\) we have \(H_i(S) = \frac{\log \sigma}{\alpha}\) for \(i \geq k\) and \(H_i(S) = \log \sigma\) for \(i < k\), for example by starting the construction with de-Bruijn words over larger alphabet than binary. This would prove that we cannot hope for a bound of \(O(|S|H_k(S))\) for \(k = \alpha \log_\sigma n\).
References

1. Janos Aczél. On Shannon’s inequality, optimal coding, and characterizations of Shannon’s and Rényi’s entropies. In *Symposia Mathematica*, volume 15, pages 153–179, 1973.

2. Alberto Apostolico and Stefano Lonardi. Some theory and practice of greedy off-line textual substitution. In *Data Compression Conference, 1998. DCC’98. Proceedings*, pages 119–128. IEEE, 1998.

3. Martin Bokler and Eric Hildebrandt. Rule reduction—a method to improve the compression performance of grammatical compression algorithms. *AEU-International Journal of Electronics and Communications*, 65(3):239–243, 2011.

4. Kutrin Casel, Henning Fernau, Serge Gaspers, Benjamin Gras, and Markus L. Schmidt. On the complexity of grammar-based compression. In *ICALP*, LNCS. Springer, 2016. accepted.

5. Moses Charikar, Eric Lehman, Ding Liu, Rina Panigrahy, Manoj Prabhakaran, Amit Sahai, and Abhi Shelat. The smallest grammar problem. *IEEE Trans. Information Theory*, 51(7):2554–2576, 2005. URL: http://dx.doi.org/10.1109/TIT.2005.850116, doi:10.1109/TIT.2005.850116.

6. Francisco Claude and Gonzalo Navarro. A fast and compact web graph representation. In Nivio Ziviani and Ricardo A. Baeza-Yates, editors, *SPIRE*, volume 4726 of LNCS, pages 118–129. Springer, 2007. URL: http://dx.doi.org/10.1007/978-3-540-75530-2_11, doi:10.1007/978-3-540-75530-2_11.

7. Francisco Claude and Gonzalo Navarro. Improved grammar-based compressed indexes. In *International Symposium on String Processing and Information Retrieval*, pages 180–192. Springer, 2012.

8. Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, 2006.

9. Paolo Ferragina, Fabrizio Luccio, Giovanni Manzini, and S Muthukrishnan. Structuring labeled trees for optimal succinctness, and beyond. In *Foundations of Computer Science, 2005. FOCS 2005. 46th Annual IEEE Symposium on*, pages 184–193. IEEE, 2005.

10. Paolo Ferragina and Giovanni Manzini. Compression boosting in optimal linear time using the Burrows-Wheeler transform. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA ’04, pages 655–663, Philadelphia, PA, USA, 2004. Society for Industrial and Applied Mathematics. URL: http://dl.acm.org/citation.cfm?id=982792.982892.

11. Paolo Ferragina and Rossano Venturini. A simple storage scheme for strings achieving entropy bounds. *Theor. Comput. Sci.*, 372(1):115–121, 2007. URL: https://doi.org/10.1016/j.tcs.2006.12.012, doi:10.1016/j.tcs.2006.12.012.

12. Johannes Fischer, Veli Mäkinen, and Gonzalo Navarro. Faster entropy-bounded compressed suffix trees. *Theor. Comput. Sci.*, 410(51):5354–5364, 2009. URL: http://dx.doi.org/10.1016/j.tcs.2009.09.012, doi:10.1016/j.tcs.2009.09.012.

13. Michal Gańczorz and Artur Jeż. Improvements on re-pair grammar compressor. *2017 Data Compression Conference (DCC)*, pages 181–190, 2017.

14. Rodrigo González and Gonzalo Navarro. Statistical encoding of succinct data structures. In Moshe Lewenstein and Gabriel Valiente, editors, *Combinatorial Pattern Matching, 17th Annual Symposium, CPM 2006, Barcelona, Spain, July 5-7, 2006, Proceedings*, volume 4009 of *Lecture Notes in Computer Science*, pages 294–305. Springer, 2006. URL: https://doi.org/10.1007/11780441_27, doi:10.1007/11780441_27.

15. Rodrigo González and Gonzalo Navarro. Compressed text indexes with fast locate. In *Annual Symposium on Combinatorial Pattern Matching*, pages 216–227. Springer, 2007.

16. Rodrigo González and Gonzalo Navarro. Compressed text indexes with fast locate. In Bin Ma and Kaizhong Zhang, editors, *CPM*, volume 4580 of LNCS, pages
Rodrigo González, Gonzalo Navarro, and Héctor Ferrada. Locally compressed suffix arrays. *J. Exp. Algorithmics*, 19:1:1.1–1.1:1.30, January 2015. URL: http://doi.acm.org/10.1145/2594408, doi:10.1145/2594408.

Roberto Grossi, Rajeev Raman, Srinivasa Rao Satti, and Rossano Ventura. Dynamic compressed strings with random access. In *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part I*, pages 504–515. Springer, 2013. URL: https://doi.org/10.1007/978-3-642-39206-1_43, doi:10.1007/978-3-642-39206-1_43.

Danny Hucke, Artur Jeż, and Markus Lohrey. Approximation ratio of repair. *CoRR*, abs/1703.06061, 2017.

Danny Hucke, Markus Lohrey, and Carl Philipp Reh. The smallest grammar problem revisited. In Shunsuke Inenaga, Kunihiko Sakakane, and Tetsuya Sakai, editors, *SPIRE*, volume 9954 of *LNCS*, pages 35–49, 2016. URL: https://doi.org/10.1007/978-3-319-46049-9_4, doi:10.1007/978-3-319-46049-9_4.

Guy Joseph Jacobson. *Succinct Static Data Structures*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, 1988. AAI8918056.

Artur Jeż. A really simple approximation of smallest grammar. *Theoretical Computer Science*, 616:141–150, 2016. URL: http://dx.doi.org/10.1016/j.tcs.2015.12.032, doi:10.1016/j.tcs.2015.12.032.

Takuya Kida, Tetsuya Matsumoto, Yusuke Shibata, Masayuki Takeda, Ayumi Shinohara, and Setsuo Arikawa. Collage system: a unifying framework for compressed pattern matching. *Theor. Comput. Sci.*, 1(298):253–272, 2003. URL: http://dx.doi.org/10.1016/S0304-3975(02)00426-7, doi:10.1016/S0304-3975(02)00426-7.

John C. Kieffer and En-Hui Yang. Grammar-based codes: A new class of universal lossless source codes. *IEEE Transactions on Information Theory*, 46(3):737–754, 2000. URL: https://doi.org/10.1109/18.841160, doi:10.1109/18.841160.

S. Rao Kosaraju and Giovanni Manzini. Compression of low entropy strings with Lempel-Ziv algorithms. *SIAM J. Comput.*, 29(3):893–911, 1999. URL: https://doi.org/10.1137/S0097539797331105, doi:10.1137/S0097539797331105.

Sebastian Kreft and Gonzalo Navarro. Self-indexing based on lz77. In *Annual Symposium on Combinatorial Pattern Matching*, pages 41–54. Springer, 2011.

N. Jesper Larsson and Alistair Moffat. Offline dictionary-based compression. In *Data Compression Conference*, pages 296–305. IEEE Computer Society, 1999. doi:10.1109/DCC.1999.755679.

Markus Lohrey. Algorithmics on SLP-compressed strings: A survey. *Groups Complexity Cryptology*, 4(2):241–299, 2012.

Giovanni Manzini. An analysis of the burrows– wheeler transform. *J. ACM*, 48(3):407–430, May 2001. URL: http://doi.acm.org/10.1145/382780.382782, doi:10.1145/382780.382782.

Gonzalo Navarro and Luís M. S. Russo. Re-Pair achieves high-order entropy. In *DCC*, page 537. IEEE Computer Society, 2008. URL: http://dx.doi.org/10.1109/DCC.2008.79, doi:10.1109/DCC.2008.79.

Craig G Nevill-Manning and Ian H Witten. Compression and explanation using hierarchical grammars. *The Computer Journal*, 40(2_and_3):103–116, 1997.
Craig G. Nevill-Manning and Ian H. Witten. Identifying hierarchical structure in sequences: A linear-time algorithm. *J. Artif. Intell. Res. (JAIR)*, 7:67–82, 1997. doi:10.1613/jair.374

Wojciech Rytter. Application of Lempel-Ziv factorization to the approximation of grammar-based compression. *Theor. Comput. Sci.*, 302(1-3):211–222, 2003. doi:10.1016/S0304-3975(02)00777-6

Kunihiko Sadakane and Roberto Grossi. Squeezing succinct data structures into entropy bounds. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 1230–1239. Society for Industrial and Applied Mathematics, 2006.

Hiroshi Sakamoto. A fully linear-time approximation algorithm for grammar-based compression. *J. Discrete Algorithms*, 3(2-4):416–430, 2005. doi:10.1016/j.jda.2004.08.016

James A. Storer and Thomas G. Szymanski. The macro model for data compression. In Richard J. Lipton, Walter A. Burkhard, Walter J. Savitch, Emily P. Friedman, and Alfred V. Aho, editors, *STOC*, pages 30–39. ACM, 1978.

Wojciech Szpankowski. Asymptotic properties of data compression and suffix trees. *IEEE transactions on Information Theory*, 39(5):1647–1659, 1993.

Yasuo Tabei, Yoshimasa Takabatake, and Hiroshi Sakamoto. A succinct grammar compression. In *Annual Symposium on Combinatorial Pattern Matching*, pages 235–246. Springer, 2013.

Yoshimasa Takabatake, Yasuo Tabei, and Hiroshi Sakamoto. Variable-length codes for space-efficient grammar-based compression. In *International Symposium on String Processing and Information Retrieval*, pages 398–410. Springer, 2012.

Michal Vasinek and Jan Platos. Prediction and evaluation of zero order entropy changes in grammar-based codes. *Entropy*, 19(5):223, 2017. URL: https://doi.org/10.3390/e19050223 doi:10.3390/e19050223

Raymond Wan. *Browsing and Searching Compressed Documents*. Phd thesis, Department of Computer Science and Software Engineering, University of Melbourne, 2003. URL: http://hdl.handle.net/11343/38920

Aaron D. Wyner and Jacob Ziv. The sliding-window Lempel-Ziv algorithm is asymptotically optimal. *Proceedings of the IEEE*, 82(6):872–877, Jun 1994.

Jacob Ziv and Abraham Lempel. A universal algorithm for sequential data compression. *IEEE Trans. Information Theory*, 23(3):337–343, 1977. URL: https://doi.org/10.1109/TIT.1977.1055714 doi:10.1109/TIT.1977.1055714

Jacob Ziv and Abraham Lempel. Compression of individual sequences via variable-rate coding. *IEEE Trans. Information Theory*, 24(5):530–536, 1978. URL: https://doi.org/10.1109/TIT.1978.1055934 doi:10.1109/TIT.1978.1055934
Appendix

A Additional material for Section 2

For simplicity if $i > j$ then $w[i \ldots j] = \epsilon$.

We extend the notion $|w|_v$ to sets of words, i.e. $|w|_V = \sum_{v \in V} |w|_v$.

The size of the nonterminal $|\text{rhs}(X)|$ is the length of its right-hand side, for a grammar $G$ or full grammar $(S, G)$ the $|\text{rhs}(G)|$, $|\text{rhs}(S, G)|$ denote the lengths of concatenations of strings in $\text{rhs}(G)$, $\text{rhs}(S, G)$, respectively.

B Additional proofs for section 3

Proof Lemma 5. The lemma follows from the definition of $C_k(Y_S)$ and $H_k$. The $|S|H_k(S)$ can be viewed in such way that each letter $a$ occurring in context $v$ substitutes $\log \left( \frac{|w|_a}{|w|_v} \right)$ to the sum. Now observe that $C_k(Y_S)$ is almost the same as $|S|H_k(S)$, but for the first $k$ letters of each phrase instead of summand $\log \left( \frac{|w|_a}{|w|_v} \right)$ we have $\log \sigma$. ◐

Proof lemma 6. Consider $l$ parsings $Y^0_S, \ldots, Y^{l-1}_S$ of $S$, where in $Y^i_S$ the first phrase has $i$ letters and the other phrases are of length $l$, except maybe the last phrase (to streamline the argument, we add an empty phrase to $Y^0_S$). Denote $Y^i_S = y^i_0, y^i_1, \ldots$. We estimate the sum $\sum_{i=0}^{l-1} C(Y^i_S)$. The costs of each of the first phrases is upper-bounded by $\log n$, as for any phrase cost is at most $\log n$:

\[
C(y) = -\log \prod_{j=1}^{|y|} \frac{|S|_{y[1..j-1]}}{|S|_{y[1..j]}}
= -\log \left( \frac{|S|_y}{|S|_v} \right)
\leq \log(n/1).
\]

Then $\sum_{i=0}^{l-1} C(Y^i_S)$ without the costs of these phrases is:

\[
-\sum_{i=0}^{l-1} \sum_{p=1}^{l-1} \sum_{j=1}^{|Y^i_p|} \log \frac{|S|_{y^i_p[1..j]}}{|S|_{y^i_p[1..j-1]}} \tag{3}
\]

and we claim that this is exactly $\sum_{i=0}^{l-1} H_i(S)$. Together with the estimation of the cost of the first phrases this yields the claim, as

\[
\sum_{i=0}^{l-1} C(Y^i_S) \leq l \log n + |S| \sum_{i=0}^{l-1} H_i(S)
\]

and so one of the $l$ parsings has cost that is at most the right-hand side divided by $l$.

To see that (3) is indeed the sum of entropies observe for each position $m$ of the word we count log of probability of this letter occurring in preceding $i = 0, 1, \ldots \min(l, m - 1)$-letter context exactly once: this is clear for $m \geq l$, as the consecutive parsings are offsetted by one position; for $m < l$ observe that in $Y^S_0, Y^S_1, \ldots, Y^S_{m-1}$ the letter at position $m$ we count log of probability of this letter occurring in preceding $m, m - 1, \ldots, 0$ letter context, while in $Y^S_m, Y^S_{m+1}, \ldots$ we include it in the first phrase, so it is not counted in (3). ◐
proof of Theorem 7 We prove only the first inequality, the proof of the second one is similar.

Let $Y$ denote set of different phrases of $Y_S$. For each $y \in Y$ define

$$p(y) = \frac{|L||y|}{|L|} \cdot \mathcal{P}(y) = \frac{|L||y|}{|L|} \cdot 2^{C(y)}.$$ 

We want to show that $\sum_{y \in Y} p(y) \leq 1$, that is, it satisfies conditions of Lemma 3.

First, we prove that for each $l$ it holds that

$$\sum_{y \in \Gamma^l} \mathcal{P}(y) \leq 1 .$$ 

This claim is similar to [25, Lem A.1], we prove it by induction on $l$. For $l = 1$ we have that

$$\sum_{s \in \Gamma} \mathcal{P}(s) = \sum_{s \in \Gamma} \frac{|S|}{|S|} = 1$$

For $l > 1$ we will group elements in sum by their $l - 1$ letter prefixes.

$$\sum_{y \in \Gamma^l} \mathcal{P}(y) = \sum_{y' \in \Gamma^{l-1}} \sum_{s \in \Gamma} \mathcal{P}(y's)$$

$$\leq \sum_{y' \in \Gamma^{l-1}} \sum_{s \in \Gamma} \mathcal{P}(y') \cdot \frac{|S|}{|S|}$$

$$\leq \sum_{y' \in \Gamma^{l-1}} \mathcal{P}(y') \cdot \sum_{s \in \Gamma} \frac{|S|}{|S|}$$

$$\leq \sum_{y' \in \Gamma^{l-1}} \mathcal{P}(y') \cdot 1$$

$$\leq \sum_{y' \in \Gamma^{l-1}} \mathcal{P}(y')$$

$$= 1 ,$$

where the last equation follows from induction hypothesis. Then

$$\sum_{y \in Y} p(y) \leq \sum_{y \in Y} \frac{|L||y|}{|Y_S|} \cdot \mathcal{P}(y)$$

$$\leq \sum_{l=1}^{\frac{|Y_S|}{|Y_S|}} \frac{|L||y|}{|Y_S|} \cdot \sum_{y : y \in Y, |y| = l} \mathcal{P}(y)$$

$$\leq \sum_{l=1}^{\frac{|Y_S|}{|Y_S|}} \frac{|L||y|}{|Y_S|} \cdot 1$$

$$= 1 .$$
Thus $p$ satisfies the assumption of Lemma 3 and we can apply it on $Y_S$ and $p$:

$$|Y_S| H_0(Y_S) \leq - \sum_{i=1}^{\lfloor |Y_S| \rfloor} \log p(y_i)$$

$$= - \sum_{i=1}^{\lfloor |Y_S| \rfloor} \log \left( \frac{|L||y_i|}{|L|} \right) \cdot P(y_i)$$

$$= - \sum_{i=1}^{\lfloor |Y_S| \rfloor} \log \frac{|L||y_i|}{|L|} - \sum_{i=1}^{\lfloor |Y_S| \rfloor} \log P(y_i)$$

$$\leq |L| H_0(L) + \sum_{i=1}^{\lfloor |Y_S| \rfloor} C(y_i)$$

$$= |L| H_0(L) + C(Y_S) \quad \blacklozenge$$

We now estimate the entropy of lengths $|L| H_0(L)$, in particular in case of small parsings, i.e. when $|Y_S| = o(|S|)$. Those estimations include standard results on entropy of lengths [8, Lemma 13.5.4] and some simple calculations. (Note that a weaker estimation with stronger assumption was used implicitly in [25, Lemma A.3]).

Lemma 28 (Entropy of lengths). Let $S$, $|S| = n$ be a string, $Y_S$ its parsing and $L = \text{Lengths}(Y_S)$. Then:

$$|L| H_0(L) \leq |L| \log \frac{|S|}{|L|} + |L|(1 + \log e) .$$

In particular, if $|Y_S| = o(|S|)$ then

$$|L| H_0(L) = o(|S|) ,$$

and for any value of $|Y_S|:

$$|L| H_0(L) = \mathcal{O}(|S|) .$$

Proof of Lemma 28. Introduce random variable $U$ such that $Pr[U = l] = \frac{|L||y_i|}{|L|}$. Then $|L| H_0(L) = |L| H(U)$ and $E[U] = \frac{|S|}{|L|}$, where $H$ is the entropy function for random variables and $E$ is the expected value. It is known [8, Lemma 13.5.4] that:

$$H(U) \leq E[U] + \log E[U] + 1 - E[U] \log E[U] .$$

Translating those results back to the setting of empirical entropy we obtain:

$$H_0(L) \leq \left( 1 + \frac{|S|}{|L|} \right) \log \left( 1 + \frac{|S|}{|L|} \right) - \frac{|S|}{|L|} \log \frac{|S|}{|L|}$$

$$= \log \left( 1 + \frac{|S|}{|L|} \right) + \frac{|S|}{|L|} \log \left( 1 + \frac{|L||y_i|}{|L|} \cdot \frac{|L|}{|S|} \right)$$

$$\leq \log \left( 2 \cdot \frac{|S|}{|L|} \right) + \frac{|S|}{|L|} \log \left( 1 + \frac{|L|}{|S|} \right)$$

$$\leq \log \left( 2 \cdot \frac{|S|}{|L|} \right) + 1 + \log \left( \left( 1 + \frac{1}{|S|/|L|} \right)^{|S|/|L|} \right)$$

$$\leq \log \frac{|S|}{|L|} + 1 + \log e .$$
Let the original encoding be $|S|$. We prove that $|S|$ is $o(n)$.

Moving to the second claim: assume $|S| = o(n)$ and let $n = |S|$, then $|L| = |S| = o(n)$.

To avoid ambiguity, we will denote $x_n = |L_n|/n$, then it is $o(1)$, as a function of $n$. We want to show that $|L_n|/|S|$ is also $o(1)$:

$$
\frac{|L_n|/|S|}{|n|} = \frac{|L_n|/|n|}{\log |L_n|} + (1 + \log e) \cdot \frac{|L_n|}{n} = x_n \log(1/x_n) + (1 + \log e)x_n.
$$

If $\lim_{n \to \infty} x_n = 0$ then also $\lim_{n \to \infty} x_n \log(1/x_n) + (1 + \log e)x_n = 0$, which yields the claim.

Concerning the last claim, when $|L| = |S|$ is arbitrary (but at most $n$), clearly $|L|(1+\varepsilon) \in \mathcal{O}(|S|)$ and the function $f(x) = x \log(n/x) + (1 + \log e)x$ is maximized for $x = n/e$, for which it has value $n \log n / e \in \mathcal{O}(n)$, which is easily shown by computing the derivative; so the lemma holds.

**Proofs of Theorem 1 and Theorem 2.** The inequalities in Theorem 1 follow from Lemma 6. Theorem 7 follows from Lemma 28. To see the bound on the Huffman coding observe that its size for $|S|$ is $|Y_S|/|S| + |Y_S|$ and $|Y_S| = o(n)$. It is worth mentioning that when it comes to Huffman coding we also need to store the dictionary. We can do it in a couple of ways: first, instead of storing the dictionary we can store frequencies of each symbol in unary (or Elias δ-code, see proof of Lemma 3). This consumes $\mathcal{O}(|Y_S|)$ bits of space. Second, more involved one, uses the fact that dictionary of Huffman encoding is a binary tree with vertices labeled with 0 or 1. It is enough to store the series of labels in preorder sequence, and shape of the tree separately. As dictionary is of size $n$ at most $\mathcal{O}(|Y_S|)$, then the tree also have at most $\mathcal{O}(|Y_S|)$ vertices. Applying succinct tree encoding like for example balanced parentheses [21], which takes $2v + o(v)$ bits for trees of size $v$ yields desired bound. It is worth noting that there exist also indexes for labeled trees [9], which are able not only to encode Huffman tree within bounds, but also allow to access vertices of tree in compressed form.

Similarly Theorem 2 follows from Theorem 1 follow from Lemma 6 and Theorem 7. In this case we can bound $|L|/|S|$ by $\mathcal{O}(\log |S|)$ as parsing from Lemma 6 consist of phrases of the same length, except for first and last one.

## C. Expanded commentary for section 4

We prove that Re-Pair and Greedy stopped after certain iteration achieve $|S|/\log_2|S| + o(n \log \sigma)$ bits and their number of different nonterminals is small, i.e $O(n^c)$ for some $c < 1$. Our analysis suggest that we can decrease value $c$ at the expense of the constant hidden in $o(n \log \sigma)$. This is particularly important in practical applications as it translates to grammar which have small output alphabets, and as in most compressors every symbol is encoded using prefix-free code, this reduces bitsize required to store this codes. In comparison certain dictionary methods, like LZ78, do not have this property, as they can produce string over a large, i.e $\Omega(\frac{n}{\log_2 n})$-sized alphabet, which implies a large dictionary size.

### C.1 Proofs for 4.1

We first complete promised argument that at least some of original encodings of Re-Pair match previously presented bound on CNF encoding. As mentioned before original paper on Re-Pair [27] suggested encoding based on division of nonterminals $G$ into $z$ groups $g_1, \ldots, g_z$ where $X \rightarrow AB \in g_i$ if and only if $A \in g_{i-1}$ and $B \in g_j$ or $B \in g_{i-1}$ and $A \in g_j$, for some $j \leq i - 1$, where $g_0$ contains all letters of the input alphabet. It was shown that with above assumptions each group $g_i$, $i > 0$ can be represented as 0/1 array of size...
\( p_i^2 - p_{i-1}\) are \( p_i - p_{i-1}\) ones, where \( p_i = (\sum_{j \leq i} |y_j|)\). The original paper on Re-Pair \cite{gancorz2016} encoded the arrays using arithmetic coding. Now using arithmetic coding with probabilities \( P(1) = \frac{1}{|N(G)| + \sigma}\) and \( P(0) = \frac{|N(G)| + \sigma - 1}{|N(G)| + \sigma}\) yields that each group is encoded using at most \((p_i - p_{i-1}) \log(|N(G)| + \sigma) + (p_i^2 - p_{i-2}) \log \frac{|N(G)| + \sigma}{|N(G)| + \sigma - 1} + O(1)\) bits. Summing over all groups we get that the size is bounded by

\[
\sum_{i=1}^{z} \left( (p_i - p_{i-1}) \log(|N(G)| + \sigma) + (p_i^2 - p_{i-2}) \log \frac{|N(G)| + \sigma}{|N(G)| + \sigma - 1} + O(1) \right)
\]

Which, as the sum telescopes, can be bounded by:

\[
|N(G)| \log(|N(G)| + \sigma) + (|N(G)| + \sigma)^2 \log \frac{|N(G)| + \sigma}{|N(G)| + \sigma - 1} + O(|N(G)|)
\]

As \( x \log \frac{1}{x} = O(1) \) we have that the total cost of encoding is at most:

\[
|N(G)| \log(|N(G)| + \sigma) + O(|N(G)| + \sigma)
\]

It is worth mentioning that instead of using fixed arithmetic coder original solution used adaptive one, which can be beneficial in practice.

**Proof of Lemma** \cite{gancorz2016} In each of the encodings we need to store sizes of right hand sides of nonterminals, storing this values in unary is within bounds for each method. In the case of entropy coding, it is sometimes more practical to add separator characters to string, observe that this solution also costs additional \( O(|\text{rhs}(S', G)|) \) bits.

The bound for fully naive encoding is obvious, as symbols on the right sides are either one of original \( \sigma \) symbols or one of \(|N(G)|\) nonterminals.

In the naive encoding the starting string is encoded using entropy coding, so \(|S'| H_0(S') + O(|S'|)\) bits are used, the rest is the same as in the first case.

In the case of incremental encoding, let us first argue that we can always reorder nonterminals appropriately. Consider the following procedure: start with nonterminals which have only letters on their right side, and sort them lexicographically, treating each nonterminal as a pair of letters. Call the created sequence \( R \), this sequence contains already processed nonterminals. Then take the first unprocessed nonterminal \( X \to YZ \) with \( Y \) being the earliest possible nonterminal occurring in \( R \), and append \( X \) to \( R \). At the end of the above procedure it is necessary to rename nonterminals so that they will correspond to numbers numbers in created sequence.

In incremental encoding we encode the second element of pairs naively. As first elements are sorted, we store only consecutive differences. For this differences we use Elias \( \delta \)-codes. These codes, for a number \( x \) consume at most \( \log x + 2 \log \log (1 + x) + 1 \) bits. As all these differences sum up to \(|N(G)| + \sigma\), it can be shown that this encoding takes at most \( O(|N(G)| + \sigma) \) bits \cite{gancorz2016} Theorem 2.2]). Observe that encoding these differences in unary is also within the required bounds.

**Lemma 29** (Full version of Lemma \cite{gancorz2016}). Let \( S \) be a string over an alphabet of size \( \sigma \) and \((S', G)\) a full grammar generating it, and \( k = \sigma(\log_\sigma n)\). Denote \( S_G = \text{rhs}(S', G) \). Assume that no two nonterminals in \((S', G)\) have the same expansion. If \(|S_G| = O\left(\frac{n}{\log^{k-1} n}\right)\) then the entropy coding of \((S', G)\) is

\[
|S_G| H_0(S_G) \leq |S| H_k(S) + |N(G)| \log |S| + \sigma(|S| \log \sigma).
\]
In particular, if additionally $|\mathcal{N}(G)| = o\left(\frac{|S|}{\log |S|}\right)$ then

$$|S_G|H_0(S_G) \leq |S|H_k(S) + o(|S| \log \sigma) .$$

Proof of Lemma 9. From the full grammar $(S', G)$ we can create a parsing $Y_{S''}$ of string $S$ of size $|Y_{S''}| \leq |\text{rhs}(G)| + |S'|$ by iterating the following procedure: Take the starting string $S'$ as $S''$. While there is a nonterminal $X$ such that it occurs in $S''$ and it was not processed before, replace one of its occurrence in $S''$ with the rhs$(X)$. Clearly $S''$ is over the input alphabet and nonterminals and $|S''| = |S'| + |\text{rhs}(G)| - |\mathcal{N}(G)| = O\left(\frac{n}{\log n}\right)$, as each right-hand side is substituted and each nonterminal is replaced once. String $S''$ induces the parsing $Y_S$ of $S$ and applying Theorem 1 to $Y_S$ yields

$$|Y_S|H_0(Y_S) \leq |S|H_k(S) + o(n \log \sigma) .$$

It is left to compare $|Y_S|H_0(Y_S)$ with the size of entropy encoding $S_G = |\text{rhs}(S', G)|$. Note that up to permuting of letters, $S''$ is a concatenation of $S$ and $|\text{rhs}(G)|$ with one occurrence of each nonterminal removed. As entropy coding has the same size regardless of the order of letters, the size of the entropy coding of $S$ and $G$ is the same as the size of entropy coding of $S''$ with one occurrence of each nonterminal concatenated to it. Adding one symbol to string of length at most $m$ can increase the entropy by at most $\log m + \beta$, for some constant $\beta$, we obtain the bound on the entropy coding of $S_G$:

$$|S_G|H_0(S_G) \leq |S|H_k(S) + o(n \log \sigma) + |\mathcal{N}(G)| \log |S| + O(|\mathcal{N}(G)|) .$$

The additional assumption on $|\mathcal{N}(G)|$ in the second statement clearly yields the second bound.  

C.2 Proofs for 4.2

Let the input text be called $S$ and let $S'$ be the name of the working string. We recall some well-known properties of Re-Pair, some were used in previous analysis [30]:

Lemma 30 ([30, Lem. 3]). The frequency of the most frequent pair in the working string does not increase during Re-Pair’s execution.

Lemma 31. If the most frequent pair in the working string occurs at least $z$ times then $|\mathcal{N}(G)| < \frac{n}{z}$.

Proof. By Lemma 30 all replaced pairs had frequency at least $z$ thus each such a replacement removed at least $z$ letters from the starting string. As it initially had $n$ letters, this cannot be done $n/z$ times.

We also need the following property as we want to apply Theorem 1 to a working string. This can be proved by simple induction, see [30, Lem. 4] for a proof of a more general statement.

Lemma 32. Let $G$ be a grammar generated by Re-Pair at any iteration. Then no two nonterminals $X, Y \in G$ have the same expansion.

Equipped with above properties we are ready to prove the Theorem 10.
proof of Theorem 10. Assume that the working string $S'$ of Re-Pair has size $c > \frac{e}{4 \log_{\sigma} n}$, the constant $e$ is defined later on. Then $S'$ induces a parsing $Y_S = y_1 y_2 \ldots y_c$. A phrase $y_i$ in $Y_S$ is long if $|y_i| \geq \frac{1}{4 \log_{\sigma} n}$, and short otherwise. Let $c_l$ and $c_s$ be the number of long and short phrases, respectively. Then

$$c_l \cdot \frac{4 \log_{\sigma} n}{e} \leq n \quad \Rightarrow \quad c_l \leq \frac{n \cdot e}{4 \log_{\sigma} n} \leq \frac{c}{4} \quad \Rightarrow \quad c_s \geq \frac{3c}{4} = \frac{3n \cdot e}{4 \log_{\sigma} n}.$$ 

On the other hand, the number of different short phrases is at most (it is here where we fix $e$ to be 16):

$$\lceil \frac{4 \log_{\sigma} n - 1}{2} \rceil \sum_{i=1}^{\frac{c}{4}} \sigma^i \leq 2n^\frac{1}{4} = 2n^\frac{1}{4}. \quad (4)$$

There are exactly $c - 1$ digrams in $Y_S$. Any long phrase can partake in 2 pairs, so there are at most $c/2$ such pairs, the remaining $(c - 2)/2$ pairs consist of two short phrases. Compare this with the estimation in (4) we conclude that some pair has at least $(c - 2)/8 \sqrt{n}$ occurrences, as there are at most $2n^\frac{1}{4} \cdot 2n^\frac{1}{4} = 4\sqrt{n}$ pairs of short phrases. Those occurrences can overlap if its two phrases are the same, thus the number of disjoint pairs is at least

$$\frac{c - 2}{8 \sqrt{n}} \geq \left( \frac{16n}{\log_{\sigma} n} - 2 \right) \frac{1}{8 \sqrt{n}}$$

$$\geq \frac{n}{\log_{\sigma} n} \frac{1}{\sqrt{n}}$$

$$= \frac{\sqrt{n}}{\log_{\sigma} n},$$

which is larger than 1 except for very small inputs.

Note that this in particular implies that if the working string is longer than $16n/\log_{\sigma} n$ then there is another iteration of Re-Pair. Since there is a pair with frequency $\sqrt{n}/\log_{\sigma} n$, by Lemma 31 we can estimate $|N(G)| \leq \sqrt{n} \log_{\sigma} n$, thus $|\text{rhs}(G)| \leq 2\sqrt{n} \log_{\sigma} n$ and so any considered encoding of $G$ can be bounded by $o(n)$, see Lemma 8.

Now, consider the first iteration in which $|S'| \leq \frac{16n}{\log_{\sigma} n}$. Then at this point $|N(G)| \leq 1 + 4\sqrt{n} \log_{\sigma} n$ and so again any encoding of $G$ takes $o(n)$ bits. On the other hand, from Theorem 1 the entropy coding of string $S'$ takes at most:

$$|S'| H_0(S') \leq |S| H_k(S) + o(n \log \sigma).$$

Summing the bit size of string and the grammar gives us that the encoding size is bounded by $|S| H_k(S) + o(n \log \sigma)$ for both the naive and the incremental encoding.

The bound for the entropy coding of full grammar $(S', G)$ follows directly from Lemma 9.

Proof of Lemma 11. Fix $n$ and an alphabet $\Gamma = \{a_1, a_2, \ldots, a_n, \#\}$. Consider the word $S$ which contains all letters $a_i$ with # inbetween, first in order from 1 to $n$, then in order $n$ to 1:

$$S = a_1 \# a_2 \# a_3 \# \cdots \# a_{n-1} \# a_n \# a_{n-1} \cdots a_2 \# a_1 \#.$$

Then
\[ |S| H_0(S) = 2n \log \frac{4n}{2} + 2n \log \frac{4n}{2n} \]
\[ = 2n \log (2n \cdot 2) \]
\[ = \frac{|S|}{2} \log |S| . \]

Every pair occurs twice in \( S \). We can assume that Re-Pair takes pairs in left-to-right order in case of a tie, hence Re-Pair will produce a starting string
\[ S' = X_1 X_2 \ldots X_n X_n X_n - 1 \ldots X_1 . \]

Its entropy is
\[ |S'| H_0(S') = \frac{|S|}{2} \log \frac{|S|/2}{2} \]
\[ = \frac{|S|}{2} \log |S| - |S| , \]

i.e. the entropy of working string decreased only by a lower order term during the execution of Re-Pair. The produced grammar contains \( \frac{|S|}{4} \) rules, thus by the assumption that grammar of size \( g \) takes at least \( g \log g - O(g) \) bits the total cost of encoding is at least:
\[ \frac{|S|}{2} \log |S| - |S| + \frac{|S|}{4} \log \frac{|S|}{4} = O \left( \frac{|S|}{4} \right) = \frac{3}{4} \log |S| - O(S) . \]

The same claim holds for entropy encoding of the grammar: Let \( S_G \) be a string obtained by concatenating right hand sides of \( (S', G) \). We have
\[ |S_G| H_0(S_G) = \frac{3}{4} \log |S| - O(|S|) , \]

as each of letters \( a_i \) and nonterminals \( X_i \) occurs constant number of times in \( S_G \).

We define the class of grammar required for proofs for Re-Pair and Greedy. Both of the above compressor fall into this category, we will use this results when entropy of working string (or entropy of rhs \( (S', G) \) ) is large. The Lemma 35 and Lemma 36 state that for the defined grammars example from Lemma 11 is tight. This result are also of its own interest, as it shows that for large entropy strings even not-so-reasonable grammar transformations preserve entropy within nontrivial factor.

\textbf{Definition 33.} (weakly non-redundant grammars) A full grammar \((S', G)\) is weakly non-redundant if every nonterminal \( X \), except the starting symbol, occurs at least twice in the derivation tree of \((S', G)\) and for every nonterminal \( X \) we have \(| \text{rhs}(X) | \geq 2 \).

For example grammar: \( S \to AA, A \to Ba, B \to aa \) is weakly non-redundant, \( S \to AB, A \to cc, B \to aa \) is not.

We state the following property of weakly non-redundant grammar.

\textbf{Lemma 34.} Let \( S \) be a string and \((S', G)\) be a weakly non-redundant grammar. Then \((S', G)\) can be obtained by a series of replacements, where each replacement operates only on current starting \( S'' \), replaces at least 2 occurrences of some substring \( w \), where \(|w| > 1 \) and \( w \) occurs at least 2 times, and adds a rule \( X_w \to w \) to the grammar.
Proof. Take any weakly non-redundant grammar \((S', G)\) generating \(S\). Consider the following inductive reasoning: Take some nonterminal \(X \rightarrow ab\), where \(ab\) are original symbols of \(S\) and \(ab\) occurs at least twice in \(S\). Replace \(ab\) in \(S\) by \(X\) obtaining \(S''\). Now we can remove \(X\) from \((S', G)\) and obtain weakly non-redundant grammar for \(S''\), where lemma holds by the induction hypothesis.

In the proofs of following Lemmas the crucial assumption is that \(|\text{rhs}(X)| \geq 2\), for every nonterminal \(X\), as then each introduced nonterminal shortens the starting string by at least 2.

We need the following Lemma to prove the bound on incremental encoding of Re-Pair.

Intuitively adding new rule to grammar in the case of incremental encoding costs \(\log n\) bits.

\[ \text{Lemma 35.} \]

Let \(S\) be a string and \(n \in \mathbb{N}\) a number. Assume \(|S| \leq n\) and \(|S|H_0(S) \geq \frac{|S| \log n}{2} - \gamma\), where \(\gamma\) can depend on \(|S|\). Let \((S', G)\) be a weakly non-redundant grammar. Then:

\[ |S'|H_0(S') + |\mathcal{N}(G)| \log n \leq \frac{3}{2} |S|H_0(S) + O(|S| + \gamma) . \]

Proof of Lemma 35. By Lemma 34 we can assume that grammar is produced by series of replacements on the starting string \(S\) that is we never modify right hand sides of \(G\) after adding a rule. Assume that:

\[ |S'|H_0(S') + |\mathcal{N}(G)| \log n > \frac{3}{2} |S|H_0(S) , \tag{5} \]

as otherwise the lemma trivially holds. Then \(S'\) induces some a parsing of \(S\), as each different symbol expands to a different substring. To upper bound the entropy first consider (5):

\[ \frac{3}{2} |S|H_0(S) < |S'|H_0(S') + |\mathcal{N}(G)| \log n \]

\[ \leq (|S'| + |\mathcal{N}(G)|) \log n . \]

On the other hand:

\[ \frac{3}{2} |S|H_0(S) \geq \frac{3}{4} |S| \log n - \frac{3\gamma}{2} \]

by Lemma assumption

\[ \geq \frac{3}{4} (|S'| + 2|\mathcal{N}(G)|) \log n - \frac{3\gamma}{2} \]

as \(|S| \geq |S'| + 2|\mathcal{N}(G)|\)

Comparing those two we obtain

\[ (|S'| + |\mathcal{N}(G)|) \log n > \frac{3}{4} (|S'| + 2|\mathcal{N}(G)|) \log n - \frac{3\gamma}{2} \]

and so

\[ \frac{1}{4} |S'| \log n \geq \frac{1}{2} |\mathcal{N}(G)| \log n - \frac{3\gamma}{2} . \tag{6} \]

Let \(L = \text{Lengths}(S')\). Then estimating the entropy of parsing from Theorem 1 and \(|L|H_0(L)\) from Lemma 28 yields

\[ |S'|H_0(S') \leq |S|H_0(S) + |L|H_0(L) \leq |S|H_0(S) + O(S) . \tag{7} \]
Combining (6) and (7) gives:
\[ \frac{3}{2} |S| H_0(S) \geq |S'| H_0(S') + \frac{1}{2} |S| H_0(S) - O(|S|) \]
\[ \geq |S'| H_0(S') + \frac{1}{4} |S| \log n - \frac{\gamma}{2} - O(S) \quad \text{by Lemma assumption} \]
\[ \geq |S'| H_0(S') + \frac{1}{4} (|S'| + 2|\mathcal{N}(G)|) \log n - \frac{\gamma}{2} - O(|S|) \]
\[ \geq |S'| H_0(S') + |\mathcal{N}(G)| \log n - O(|S| + \gamma) \quad \text{by (6),} \]
which ends the proof.

The following proof is more involved than the previous one. It uses similar idea as proof of Theorem 1 that is we assign some \( p \) values to each symbol on the right hand side of \((S', G)\) and apply Lemma 3.

\[ \textbf{Lemma 36.} \text{ Let } n \in \mathbb{N} \text{ be a number and } S, |S| \leq n. \text{ Assume } |S| H_0(S) \geq \frac{|S| \log n}{2} - \gamma, \]
where \( \gamma \) can depend on \( |S| \). Let \((S', G)\) be a full weakly non-redundant grammar. Let \( S_G \) be a string obtained by concatenation of right hand sides of \((S', G)\) Then:
\[ |S_G| H_0(S_G) \leq \frac{3}{2} |S| H_0(S) + O(|S| + \gamma). \]

\[ \textbf{Proof of Lemma 36.} \text{ By Lemma 3} \text{ we can assume that grammar is produced by series of replacements on the starting string } S, \text{ that is we never modify right hand sides of } G \text{ after adding a rule.} \]

Let \( \Gamma \) be the original alphabet. and \( S_G = \text{rhs}(S', G) \). We show that
\[ |S'|_{\Gamma} + 2 |S_G|_{\mathcal{N}} \leq |S| \quad (8) \]
This clearly holds at the beginning. Suppose that we replace \( k \) copies of \( w \) in \( S' \) by a new nonterminal \( X \) and add a rule \( X \rightarrow w \). Let \( |w|_{\Gamma} \) and \( |w|_{\mathcal{N}} \) denote the number of occurrences of letters and nonterminals in \( w \), then \( |w|_{\Gamma} + |w|_{\mathcal{N}} = |w| \geq 2 \). After the replacement the values change as follows:
\[ (|S'|_{\Gamma} - k |w|_{\Gamma}) + 2 (|S_G|_{\mathcal{N}} - (k - 1)|w|_{\mathcal{N}} + k) = |S'|_{\Gamma} + 2 |S_G|_{\mathcal{N}} - (k - 1)(|w|_{\Gamma} + 2|w|_{\mathcal{N}} - 2) + 2 - |w|_{\Gamma} \]
\[ \leq |S'|_{\Gamma} + 2 |S_G|_{\mathcal{N}} - (|w|_{\Gamma} + 2|w|_{\mathcal{N}} - 2) + 2 - |w|_{\Gamma} \]
\[ \leq |S'|_{\Gamma} + 2 |S_G|_{\mathcal{N}} - 2(|w|_{\Gamma} + |w|_{\mathcal{N}} - 2) \]
\[ \leq |S'|_{\Gamma} + 2 |S_G|_{\mathcal{N}} \]
\[ \leq |S| \quad . \]

As introducing new nonterminal can only decrease number of letters we have:
\[ |S| H_0(S) \geq \sum_{a \in \Gamma} -|S'|_{a} \log p(a) + \sum_{a \in \Gamma} -|G|_{a} \log p(a) \quad (9) \]
We use Lemma 3 to bound the entropy of \( S_G \). Define \( p(a) = \frac{|S|}{|S'|} \) as the empirical probability of letter \( a \) in \( S \) and let \( p'(a) = \frac{1}{|S|} p(a) \) for original letters of \( S \) and \( p'(X) = \frac{1}{|\mathcal{N}|} \).
for nonterminal symbols. Observe that they satisfy the condition of Lemma 3

\[ \sum_a |S_G|_{a}p'(a) = \sum_{a \in \Gamma} |S_G|_{a}p'(a) + \sum_{X \in \mathcal{N}} |S_G|_{X}p'(X) \]

\[ = \sum_{a \in \Gamma} |S_G|_{a} \frac{1}{2}p(a) + \sum_{X \in \mathcal{N}} |S_G|_{X} \frac{1}{2n} \]

\[ \leq \frac{1}{2} \sum_{a \in \Gamma} |S|_{a}p(a) + |S_G|_{X} \frac{1}{2n} \]

\[ \leq \frac{1}{2} + \frac{|S|}{2} \cdot \frac{1}{|S|} \]

\[ \leq 1 . \]

So we can bound the entropy of \( S_G \) using Lemma 3

\[ |S_G|H_0(S_G) \leq \sum_a -|S_G|_{a} \log p'(a) \]

\[ = \sum_{a \in \Gamma} -|S_G|_{a} \log p'(a) + \sum_{X \in \mathcal{N}} -|S_G|_{X} \log p'(X) \]

\[ = \sum_{a \in \Gamma} -|S_G|_{a} \log p(a) + |S_G|_{X} \log |S| + |S_G|_{\Gamma} \]

\[ \leq \sum_{a \in \Gamma} -|S'|_{a} \log p(a) + \sum_{a \in \Gamma} -|G|_{a} \log p(a) + |S_G|_{\mathcal{N}} \log |S| + |S_G|_{\Gamma} \]  

(10)

If \( |S_G|_{\mathcal{N}} \leq \frac{|S|}{4} \) then

\[ |S_G|H_0(S_G) \leq \sum_a -|S'|_{a} \log p(a) + \sum_a -|G|_{a} \log p(a) + |S_G|_{\mathcal{N}} \log |S| + |S_G| \]

\[ \leq |S|H_0(S) + \frac{|S|}{4} \log |S| + |S_G| \]

\[ \leq |S|H_0(S) + \frac{|S|}{4} \log n + O(|S|) \]

\[ \leq \frac{3}{2} |S|H_0(S) + O(|S| + \gamma) , \]

which yields the claim.

So consider the case when \( |S_G|_{\mathcal{N}} > \frac{|S|}{4} \), let \( |S_G|_{\mathcal{N}} = \frac{|S|}{4} + k \), for some \( k > 0 \). We consider two cases, depending on whether \( \sum_a -|G|_{a} \log p(a) \geq k \log n \) or not.

Suppose first that \( \sum_{a \in \Gamma} -|G|_{a} \log p(a) \geq k \log n \). Observe that for each letter \( a \in \Gamma \) it holds that:

\[ |S'|_{a} + 2|G|_{a} \leq |S|_{a} . \]  

(11)

This is shown by easy induction: clearly it holds at the beginning. When we replace \( k \geq 2 \) occurrences of a word \( w \) with a nonterminal \( X \) and add the rule \( X \rightarrow w \) then \( |S'|_{a} \) drops by \( k|w|_{a} \) while \( 2|G|_{a} \) increases by \( 2|w|_{a} \leq k|w|_{a} \). Multiplying \( \log p(a) \leq 0 \) and summing over all \( a \in \Gamma \) yields

\[ \sum_a -|S'|_{a} \log p(a) + 2 \sum_a -|G|_{a} \log p(a) \leq |S|H_0(S) , \]

(12)

which implies:

\[ \sum_a -|S'|_{a} \log p(a) + \sum_a -|G|_{a} \log p(a) \leq |S|H_0(S) - k \log n . \]  

(13)
Plugging (13) and the equality $|S_G|_{\mathcal{N}} = \frac{|S|}{4} + k$ into (10) gives

$$|S_G|H_0(S_G) \leq \sum_{a \in \Gamma} -|S'|_a \log p(a) + \sum_{a \in \Gamma} -|G|_a \log p(a) + |S_G|_{\mathcal{N}} \log |S| + |S_G|$$

$$\leq |S|H_0(S) - k \log n + \frac{|S|}{4} \log n + k \log n + |S_G|$$

$$\leq |S|H_0(S) + \frac{1}{2}|S|H_0(S) + \frac{1}{2} \gamma + |S_G|$$

$$\leq \frac{3}{2} |S|H_0(S) + O(|S| + \gamma) ,$$

which yields the claim.

Consider the second case, in which $\sum_{a \in \Gamma} -|G|_a \log p(a) < k \log n$. Then:

$$\sum_{a \in \Gamma} -|S'|_a \log p(a) + \sum_{a \in \Gamma} -|G|_a \log p(a) \leq |S'| \log n + k \log n$$

$$\leq (|S| - 2|S_G|_{\mathcal{N}}) \log n + k \log n$$

$$= \left(|S| - 2 \left(\frac{|S|}{4} + k\right)\right) \log n + k \log n$$

$$\leq \frac{|S|}{2} \log n - k \log n$$

$$\leq |S|H_0(S) + \gamma - k \log n . \quad (14)$$

Plugging (14) into (11) gives

$$|S_G|H_0(S_G) \leq \sum_{a \in \Gamma} -|S'|_a \log p(a) + \sum_{a \in \Gamma} -|G|_a \log p(a) + |S_G|_{\mathcal{N}} \log |S| + |S_G|$$

$$\leq |S|H_0(S) + \gamma - k \log n + \frac{|S|}{4} \log n + k \log n + \log |S| + |S_G|$$

$$\leq |S|H_0(S) + \gamma + \frac{1}{2}|S|H_0(S) + \frac{3}{2} \log |S| + |S_G|$$

$$\leq \frac{3}{2} |S|H_0(S) + O(|S| + \gamma) ,$$

which ends the proof. △

**Proof of Theorem 12.** Let $n = |S|$. We will start by proving the theorem for incremental encoding.

We upper-bound small summands by $o(n \log \sigma)$. Since we multiply them by constants and sum up a constant number of those, this allowed. This makes the estimations easier as we do not have to carry smaller-order terms.

Fix some $\epsilon > 0$ and consider the iteration in which the number of nonterminals in the grammar is $\left\lfloor \frac{n}{\log \sigma} \right\rfloor$, call this grammar $G_0$, i.e. $|\mathcal{N}(G_0)| = \left\lfloor \frac{n}{\log \sigma} \right\rfloor$. If no such an iteration exists, then consider the state after last iteration.

Let $S'$ be the working string at this point, note that by Theorem 10 we have that $|S'| \leq \frac{8n}{\log \sigma} = O\left(\frac{n}{\log \sigma}\right)$, as when the working string reaches this size the number of the nonterminals in the grammar is at most $4\sqrt{n} \log \sigma$ and $|\mathcal{N}(G_0)| = \left\lfloor \frac{n}{\log \sigma} \right\rfloor$. In particular, by Theorem [5]

$$|S'|H_0(S') \leq |S|H_k(S) + o(n \log \sigma) \quad (15)$$
Now observe, that case when there is no such iteration that $|N(G_0)| = \left\lfloor \frac{n}{\log^{1+\varepsilon} n} \right\rfloor$ is trivial, as even the naive encoding of the grammar takes $o(n)$ bits, which together with yields desired bound. Thus from this point we assume that this is not the case.

If additionally $|S'| \leq \frac{n}{\log^{1+\varepsilon} n}$ then we can make at most half of this amount of iterations and so at any point the size of the grammar is at most $2^{\frac{n}{\log^{1+\varepsilon} n}}$ so the estimation as in the case when there is no iteration defining $G_0$ yields the claim.

So consider the case in which $|S'| > \frac{n}{\log^{1+\varepsilon} n}$. From Lemma $\S$ we get that each pair occurs at most $\log^{1+\varepsilon} n$ times in $S'$, as otherwise this contradicts $|N(G_0)| = \left\lfloor \frac{n}{\log^{1+\varepsilon} n} \right\rfloor$. We will show that this implies that the entropy $|S'|H_0(S')$ is high.

Consider the parsing of working string $S'$ into consecutive pairs (with possibly last letter left-out), denote it by $Y_{S'}^P$. Let $L_P = \text{Lengths}(Y_{S'}^P)$, observe that those are all 2s except possibly the last, which can be 1. Applying Theorem $\S$ with $k = 0$:

$$|S'|H_0(S') + |L_P|H_0(L_P) \geq |Y_{S'}^P|H_0(Y_{S'}^P).$$

If $L_P \in 2^*$ then clearly $H_0(L_P) = 0$, otherwise $L_P = 2^{|S'| - 1/2}1$ and so its entropy is

$$|L_P| = \frac{|S'| - 1}{2}\log\left(\frac{|S'|/2}{(|S'| - 1)/2}\right) + \log\left(\frac{|S'|/2}{(|S'| - 1)/2}\right) \leq \frac{1}{2}\log \left(1 + \frac{1}{|S'| - 1}\right)^{|S'| - 1} + \log |S'| \leq \frac{1}{2}\log 2 + \log |S'|.$$

On the other hand, as each pair occurs at most $\log^{1+\varepsilon} n$ times, the entropy is minimised when each pair has exactly this number of occurrences, i.e.

$$|Y_{S'}^P|H_0(Y_{S'}^P) \geq |Y_{S'}^P|\log\left(\frac{|S'| - 1}{2}\log^{1+\varepsilon} n\right) \geq \frac{|S'| - 1}{2}\log\left(\frac{n}{2\log^{2+2\varepsilon} n}\right) \geq \frac{|S'| - 1}{2}\log n - \frac{|S'| - 1}{2}\log(2\log^{2+2\varepsilon} n) \geq \frac{|S'| - 1}{2}\log n - (1 + (1 + \varepsilon))(|S'| - 1)(1 + \log \log n) \geq \frac{|S'| - 1}{2}\log n - \gamma',$$

where $\gamma' = (1 + \varepsilon)\log n = o(n \log \sigma)$. Taking those two estimations together yields

$$|S'|H_0(S') \geq \frac{|S'|}{2}\log n - \gamma,$$

where $\gamma = \gamma' + \frac{1}{2}\log 2 + \log |S'| = o(n \log \sigma)$.

We now move to the estimation of size of incremental encoding of grammar. We use estimation from Lemma $\S$ This encoding takes at most $|N(G)|\log|N(G)| + \alpha|N(G)|$ bits, for some constant $\alpha$. We upper-bound the possible increase of this estimation after one iteration, i.e. when grammar size changes from $g$ to $g + 1$:

$$\alpha(g + 1) + (g + 1)\log(g + 1) - \alpha g - g \log g = \alpha + g \log(1 + 1/g) + \log(g + 1) = \alpha n + \log((1 + 1/g)^g) + \log(g + 1) \leq \alpha + \log e + \log n = \beta + \log n$$
for $\beta = \alpha + \log e$. Let $S''$ be a string returned by Re-Pair at the end of its runtime and $G'$ be a grammar at this point. Observe that $S''$ induces a parsing of $S'$. Let $i$ be such that the returned grammar $G'$ has $|N(G_0)| + i$ rules, i.e. there were $i$ compression steps between $G_0$ and the end. Then the size of incremental encoding is at most
\[
o(n) + i(\beta + \log n) + |S''|H_0(S'')
\]
with $o(n)$ for the encoding of $G_0$, $i(\beta + \log n)$ for the following $i$ steps and $|S''|H_0(S'')$ for the entropy coding of $S''$.

We can look at the last $i$ iterations as we start with string $S'$ and end up with $S''$ and some grammar $G' \setminus G_0$, i.e. grammar $G'$ without nonterminals from $G_0$. As this grammar satisfies conditions of weakly irreducible grammar, and $S'$ satisfies $\text{[16]}$ we apply Lemma 35

\[i \log n + |S''|H_0(S'') \leq \frac{3}{2}|S'|H_0(S') + O(|S'| + \gamma).
\]

We bound the total size of the encoding, observe that $i \leq |S'|/2$ and so $i\beta = O(|S'|)$:
\[o(n) + i(\beta + \log n) + |S''|H_0(S'') \leq \frac{3}{2}|S'|H_0(S') + O(|S'| + \gamma) + o(n).
\]

Combining this with $\text{[15]}$ we obtain that the latter term is bounded by:
\[\frac{3}{2}|S'|H_0(S') + O(|S'| + \gamma) + o(n) \leq \frac{3}{2}|S|H_k(S) + O(|S'| + \gamma) + o(n) + o(n \log \sigma).
\]

As $|S'| = o(n)$ and $\gamma = o(n)$ the claim holds for incremental encoding.

Moving to the case of entropy coding, we consider the same iteration as before. Again we assume that $|N(G_0)| = \lceil \frac{n}{\log \frac{\sigma}{|S'|}} \rceil$ and $|S'| \geq \frac{n}{\log \frac{\sigma}{|S'|}}$. otherwise, as in the case for incremental encoding, we can bound the grammar size at the end of the runtime by $2 \left\lceil \frac{n}{\log \frac{\sigma}{|S'|}} \right\rceil$, thus direct application of Lemma 3 holds the claim.

Define $S_G$ as concatenation of right hand sides of full grammar $(S'', G')$, where $G'$ is a grammar at the end of runtime of Re-Pair, and let by $S_G$ be the concatenation of right hand sides of $(S', G_0)$. Using Lemma 36 we can estimate the entropy of $S_G$, note that the second estimation holds as $|N(G_0)| = \left\lceil \frac{n}{\log \frac{\sigma}{|S'|}} \right\rceil$:
\[|S_G|H_0(S_G) \leq |S|H_k(S) + |N(G_0)| \log n + o(|S| \log \sigma) \leq |S|H_k(S) + o(|S| \log \sigma) \quad (17)
\]

Observe that:
\[|S_G|H_0(S_G) \geq |S'|H_0(S')
\]
\[\geq \frac{|S'|}{2} \log n - \gamma
\]
\[\geq \frac{|S|}{2} \log n - \gamma - 2|N(G_0)| \log n
\]
\[\geq \frac{|S|}{2} \log n - \gamma'',
\]
where $\gamma'' = o(n \log \sigma)$, as $\gamma = o(n \log \sigma)$ and $|N(G_0)| = O(\frac{n}{\log \frac{\sigma}{|S'|}})$. Again, we can look at the last iterations like we would start Re-Pair with input string $S_G$ and end up with $(S'', G' \setminus G_0)$, as we can view the remaining iterations like they would do replacements on $S_G$. We apply our results for weakly irreducible grammars, i.e. Lemma 36

\[|S_G'|H_0(S_G') \leq \frac{3}{2}|S_G|H_0(S_G) + O(S_G + \gamma''). \quad (18)
\]

Combining $\text{[17]}$ with $\text{[18]}$ and observing that $|S_G| = |\text{rhs}(S', G)| = o(n)$ yields the claim.
C.3 Proofs for 4.3

The following lemma states a well-known property of irreducible grammars, which in facts holds when only \( \text{IG2} \) is satisfied.

\[ \text{Lemma 37 (\cite{5}, Lemma 4).} \quad \text{Let } (S', G) \text{ be a full grammar generating } S \text{ in which each nonterminal (except for the starting symbol) occurs at least twice on the right hand side, that is it satisfies condition } \text{IG2}. \text{ Then the sum of expansions of right sides is at most } 2|S|. \]

First we show that irreducible grammars are necessarily small, similar to what was shown in \cite{24}, yet we will use more general lemma, which is useful in discussion of Greedy algorithm.

\[ \text{Lemma 38.} \quad \text{Let } (S, G) \text{ be a full grammar satisfying conditions } \text{IG1}–\text{IG2}. \text{ If } |\text{rhs}(S, G)| \geq \frac{64n}{\log_2 n} \text{ then there is a digram occurring at least } \sqrt{n \log_2 n} \text{ times in } S \text{ and right-hand sides of } G. \]

\[ \text{Corollary 39.} \quad \text{For any grammar satisfying conditions } \text{IG1–IG3} \text{ it holds that } |\text{rhs}(S, G)| \leq \frac{64n}{\log_2 n} = \mathcal{O}(n \log_2 n). \]

\[ \text{Proof of Lemma 38.} \quad \text{As in the proof of Theorem 10, we distinguish between long and short symbols, depending on the length of their expansion: a symbol } X \text{ is long if it is a nonterminal and } |\exp(X)| \geq \frac{\log_2 n}{4}, \text{ otherwise it is short. By } c_l \text{ denote number of occurrences of long symbols, and by } c_s \text{ number of short ones, let also } c = |\text{rhs}(S', G)|, \text{ then } c = c_l + c_s. \text{ Assume for the sake of contradiction that } c \geq \frac{64n}{\log_2 n}. \text{ From Lemma 37 the sum of expansions’ lengths is at most } 2n \text{ and so}
\]
\[
c_l \cdot \frac{\log_2 n}{4} \leq 2n \implies c_l \leq \frac{8n}{\log_2 n} \leq \frac{c}{8} \implies c_s \geq \frac{7c}{8} \geq \frac{56n}{\log_2 n}.
\]

By \( \text{IG1} \) the expansions of different nonterminals are different and so the number of different short symbols is at most:
\[
\sum_{i=1}^{\left\lfloor \frac{\log_2 n}{4} \right\rfloor} \sigma^i \leq 2\sigma^{\frac{\log_2 n}{4}} \leq 2n^{\frac{1}{4}} = 2 \sqrt[4]{n}.
\]

There are at least \( |\text{rhs}(S', G)| - |N(G)| - 1 \geq c - 1 - 1 = \frac{c-3}{2} \) digrams on the right hand sides of \((S, G)\). As one long occurrence may be in at most two digrams, at least \( \frac{c-3}{2} - \frac{c}{4} \geq \frac{1}{2} \) digrams consist of two short symbols. Then there exist a digram occurring at least:
\[
\left( \frac{c}{4} - 2 \right) \cdot \frac{1}{2 \sqrt[4]{n} \cdot 2 \sqrt[4]{n}} = \frac{c \sqrt{n}}{16 \sqrt{n}} - \frac{2}{4 \sqrt{n}} \geq \frac{\log_2 n}{2 \sqrt[4]{n}} \frac{1}{2} \geq \frac{\sqrt{2}}{\log_2 n} \frac{1}{2}
\]

times. As it can consist of two identical symbols, there are at least \( \frac{\sqrt{n}}{\log_2 n} \) pairwise disjoint occurrences.

\[ \text{Proof of Theorem 15.} \quad \text{Let } |S| = n \text{ and } (S', G) \text{ be an irreducible full grammar generating } S.
\]

As in the proof of Lemma 9 we will construct parsing \( Y_S \) of \( S \) by expanding nonterminals: set \( S'' \) to be the starting string \( S' \) and while possible, take \( X \) that occurs in \( S'' \) and was
not chosen before and replace a single occurrence of $X$ in $S''$ with $\text{rhs}(X)$. Since every nonterminal is used in the production of $S$, we end up with $S''$ in which every nonterminal was expanded once, observe that $S''$ induce a parsing of $S$. As $(S', G)$ is irreducible, it is small, i.e. $|\text{rhs}(S', G)| = \mathcal{O}\left(\frac{n}{\log_2 n}\right)$ see Corollary \[39\] so as in proof of Lemma \[39\] using Theorem \[1\] we conclude that $|S''|H_0(S'') \leq |S|H_k(S) + o(n \log \sigma)$.

Let $S_N$ be a string in which every nonterminal of $G$ occurs once. Observe that $S''S_N$ has the same count of every symbol as $\text{rhs}(S', G)$: each replacement performed on $S''$ removes a single occurrence of on nonterminal and each nonterminal is processed once. Thus it is enough to estimate $|S''S_N|H_0(S''S_N)$. Observe that each nonterminal is present in $S''$: by \[42\] it had at least two occurrences in $S'$ and right-hand sides of $G$ and one occurrence was removed in the creation of $S''$. Consider the string $Z = S''S_N$: we can reorder characters of $Z$ and remove some of them and obtain $S''S_N$. The entropy is preserved by permutation of letters and it is well known and easy to verify that removal letters from a string does not increase it, thus

$$
|S''S_N|H_0(S''S_N) \leq (|S''|S'')H_0(S''S'')
= 2|S''|H_0(S'')
\leq 2|S|H_k(S) + o(n \log \sigma)
$$

Proof of Theorem \[16\] Denote $n = |S|$ and $(S', G)$ be an irreducible full grammar generating $S$ and $S_G = \text{rhs}(S', G)$. Since $(S', G)$ is irreducible, it is small, i.e. $|S_G| = \mathcal{O}\left(\frac{n}{\log_2 n}\right)$. Furthermore, as $(S', G)$ is irreducible no pair occurs twice on the right hand sides of $(S', G)$.

Therefore there exist $\frac{1}{3}|S_G|$ non-overlapping pairs in $S_G$ that occur exactly once: we can pair letters in each production, except the last letter if production’s length is odd. Let $Y_S$ be a parsing of $S_G$ into those pair and single letters. Then:

$$|Y_S|H_0(Y_S) \geq \frac{1}{3}|S_G| \log \frac{1}{3}|S_G|
= \frac{1}{3}|S_G| \log |S_G| - \mathcal{O}(|S_G|)
= \frac{1}{3}|S_G| \log |S_G| - o(n \log \sigma)
$$

On the other hand, applying Theorem \[15\] for $k = 0$ as in proof of Theorem \[12\] we get:

$$|S_G|H_0(S_G) \geq |Y_S|H_0(Y_S) - o(n \log \sigma)$$

and thus

$$|S_G|H_0(S_G) \geq \frac{1}{3}|S_G| \log |S_G| - o(n \log \sigma)
$$

On the other hand, from Theorem \[15\]

$$2|S|H_k(S) + o(n \log \sigma) \geq |S_G|H_0(S_G)
\geq \frac{1}{3}|S_G| \log |S_G| - o(n \log \sigma)
$$

and so

$$|S_G| \log |S_G| \leq 6|S|H_k(S) + o(n \log \sigma)
$$

Now, the naive coding of $S_G$ uses at most $|S_G| \log |S_G| + |S_G| = |S_G| \log |S_G| + o(n \log \sigma)$ bits, note that $S_G$ includes all letters from the original alphabet. This finishes the proof.
C.4 Proofs for 4.4

We state similar properties of Greedy to those of Re-Pair.

Lemma 40 (cf. Lemma 30). Frequency of the most frequent pair in the working string and grammar right-hand sides does not increase during the execution of Greedy.

Proof. Assume that after performing the replacement \( w \rightarrow X \) some \( AB \) occurs more times than before. Then only possibility is that either \( A = X \) or \( B = X \), as otherwise the frequency of \( AB \) cannot increase. By symmetry let us assume that \( A = X \), for the moment assume also that \( A \neq B \). Let \( A_k \) be the last symbol of \( w \); then \( A_kB \) occurred at least as many times as \( AB \) in the working string and grammar before replacement. The case with \( A = B \) is shown in the same way; the case with \( X = B \) is symmetric.

Lemma 41 (cf. Lemma 31). Let \( z \) be a frequency of the most frequent pair in the working string and grammar right-hand sides at some point of execution of Greedy. Then at this point the number of nonterminals in the grammar is at most

\[
|N(G)| \leq n \sqrt{n \log \sigma} - 2
\]

Proof. Replacing a substring \( w \) of length \( |w| \) and frequency \( f_w \) decreases the size \( |S'| + |\text{rhs}(G)| \) by

\[
f_w(|w| - 1) - |w| = (f_w - 1)(|w| - 1) - 1
\]

In particular, replacing a pair of symbols with frequency \( f \) shortens the grammar by \( f - 2 \) symbols. Now, by Lemma 40 in each previous iteration there was a pair with frequency at least \( z \). As Greedy replaces the string which shortens the working string and right-hand sides of rules by the maximal possible value, in each previous iteration \( |S'| + |\text{rhs}(G)| \) was decreased by at least \( z - 2 \), so the maximal number of previous iterations is \( \frac{n}{z-2} \) and this is a bound on number of added nonterminals.

Proof of Theorem 17. Let \( |S| = n \). From Lemma 38 if \( |S'| + |\text{rhs}(G)| \geq \frac{64n}{\log_\sigma n} \) then there exist a digram occurring \( \sqrt{n \log \sigma} \) times on the right hand side of \( G \) and \( S' \). Thus there is a point in the execution when \( |\text{rhs}(S', G)| < \frac{64n}{\log_\sigma n} \) for the first time. Lemma 41 applied at this points gives a bound on the number of nonterminals:

\[
|N(G)| \leq n \sqrt{n \log \sigma} - 2
\]

and at the first point when \( |S'| + |\text{rhs}(G)| \leq \frac{64n}{\log_\sigma n} \) it is larger by at most 1, so still \( O(\sqrt{n \log \sigma}) \). By the second claim of Lemma 9 we obtain that entropy coding of \( (S', G) \) at this point is bounded by \( |S|H_k(S) + o(n \log \sigma) \), as claimed.

We prove Theorems 18–19 simultaneously, as the latter one is the extension of the former.

Proof of Theorem 19 and Theorem 18. Let \( (S'', G') \) be a full grammar produced by Greedy. Similarly as in proof of Theorem 12 we will lower bound the entropy of grammar after some
iteration. Consider the iteration in which $|\mathcal{N}(G)| = \left\lceil \frac{n}{\log_{1+\epsilon} n} \right\rceil$; if there is no such an iteration then consider the grammar produced at the end. Let $(S', G)$ be the full grammar at this point. Using this estimation together with Theorem 17 yields that

$$|\text{rhs}(S', G)| = \mathcal{O}\left( \frac{n}{\log_{1+\epsilon} n} \right).$$

(20)

If $|\text{rhs}(S', G)| \leq \frac{n}{\log_{1+\epsilon} n}$ then the final encoding has size $o(n)$: the number of nonterminal will be at most $|\text{rhs}(S', G)| + |\mathcal{N}(G)|$: each replacement reduces the total length of right-hand-sides of rules of length at least 2 and it does not affect rules with right-hand sides of length 2. On the other hand, the replacements do not decrease the $|\text{rhs}(\cdot)|$ of the full grammar. Thus the fully naive encoding of $(S', G')$ will use at most of what the fully naive encoding of $(S', G)$, i.e.

$$\frac{n}{\log_{1+\epsilon} n} \cdot \log \left( \frac{n}{\log_{1+\epsilon} n} \right) = o(n)$$

Thus in the following we assume that

$$|\text{rhs}(S', G)| \geq \frac{n}{\log_{1+\epsilon} n}. \quad (21)$$

Using estimation (21) in Lemma 11 yields that no digram occurs more than $2 + \log_{1+\epsilon} n$ times on the right-hand sides of $(S', G)$. On the other hand, we can find at least $\frac{1}{2}(|\text{rhs}((S', G))| - |\mathcal{N}(G)| - 1)$ disjoint pairs on the right-hand side of $(S', G)$, as we can pair nonterminals naively; we subtract $|\mathcal{N}(G)|$ factor because rules and starting string can have odd length. Let $S_G = \text{rhs}(S', G)$ and $Y_S^P$ its parsing into phrases of length 1 and 2, where phrases of length 2 are pairs mentioned above. Then the entropy of $Y_S^P$ is minimised when each possible pair occurs with maximal frequency, i.e. $2 + \log_{1+\epsilon} n$. Thus

$$|Y_S^P|H_0(Y_S^P) \geq \frac{1}{2}(|\text{rhs}(S', G)| - |\mathcal{N}(G)| - 1) \log \left( \frac{n}{\log_{1+\epsilon} n} \cdot \frac{1}{\log_{1+\epsilon} n + 2} \right) \quad (22)$$

$$\geq \frac{1}{2} |\text{rhs}(S', G)| \log n - \gamma' ,$$

where $\gamma' = o(n \log \sigma)$.

Consider the entropy $|S_G|H_0(S_G)$: on one hand it can be upper-bounded by the entropy coding of the grammar and this can be upper-bounded by Lemma 11, note that its second estimation hold as $|S_G| = |\text{rhs}(S', G)| = \mathcal{O}\left( \frac{n}{\log_{1+\epsilon} n} \right)$ and the assumed $|\mathcal{N}(G)| \leq \frac{n}{\log_{1+\epsilon} n}$.

$$|S_G|H_0(S_G) \leq |S|H_k(S) + o(|S| \log \sigma) , \quad (23)$$

On the other hand Theorem 1 with $k = 0$, yields an estimation using the parsing $Y_S^P$, where $L = \text{Lengths}(Y_S^P)$:

$$|S_G|H_0(S_G) \geq |Y_S^P|H_0(Y_S^P) - |L|H_0(L)$$

$$\geq \frac{1}{2} |\text{rhs}(S', G)| \log n - \gamma ,$$

where $\gamma = \gamma' + o(n \log \sigma) = o(n \log \sigma)$. The second estimation follows from estimating $|Y_S^P|H_0(Y_S^P)$ by (22) and estimation of $|L|H_0(L)$: As $Y_S^P$ consists only of phrases of length
1 and 2 we have $|L|H_0(L) \leq |Y_k^2| \leq |\text{rhs}(S', G)| = o(n \log \sigma)$. Combining (23) and (24) we obtain
$$|\text{rhs}(S', G)| \log n \leq 2|S|H_k(S) + o(n \log \sigma).$$
As each symbol is encoded using at most $\lceil k \rceil$ bits and $|\text{rhs}(\cdot)|$ of the full grammar does not increase, the claim of Theorem 18 holds.

When it comes to the proof of Theorem 19, we make the same observation as in analysis for Re-Pair with entropy coding (see proof of Theorem 12), that is we can look at the algorithm which starts with string $S_G$ and finishes with grammar $(S'', G')$, i.e. the one returned by Greedy. Moreover the grammar $(S'', G')$ is weakly non-redundant (see Definition 33). Let $S_G'$ be string obtained by concatenation of right hand sides of $(S'', G')$. Observe that $S_G$ satisfies the condition of Lemma 36 by (24) and thus:
$$|S_G'|H_0(S_G') \leq \frac{3}{2}|S_G|H_0(S_G) + O(|S_G| + \gamma)$$
Noting that $|S_G| = o(n \log \sigma)$ and substituting (23) yields the claim.

\section{Expanded commentary for section 5}

Recall that Theorem 2 showed that for any $l$ and for any string $S$ there is a parsing $Y_S$ of size $\frac{|S|}{n} + 2$ satisfying $|Y_S|H_0(Y_S) \leq |S|\sum_{i} H_i(S) + O(\log n)$. Yet, the mean of entropies is not standard measure of compression, moreover it is not clear how the above measure corresponds to $H_k$. Therefore we will prove two facts: one that there exists family of string $|S|$ for which, for any $k < \log \sigma |S|$, we have $\beta H_i(S) = H_k(S)$, $i < k$, $\beta > 1$, second that wide range of parsing methods must produce a parsing which zero order entropy is lower bounded by mean of entropies of $S$. Moreover our string $S$ have high entropy, meaning that as mean of entropies cannot be contained in lower order term such as $o(n)$ or $o(n \log \sigma)$.

\subsection{Construction of Generalized de Bruijn words}

The promised family of words is much easier to define, if we think of them as cyclic words, meaning that after reading the last letter we can continue to read from the beginning. To distinguish words from cyclic words we denote by $S^\circ$ the cyclic variant of $S$.

A word $w$ occurs in $S^\circ$ if $w$ occurs in $S$, or $w = w_1w_2$ and $w_1$ is a prefix of $S$ and $w_2$ a suffix; we still require that $|w| \leq |S|$. The starting positions of an occurrence of $w$ in $S^\circ$ is defined naturally, two occurrences are different if they start at different positions. Denote by $|S|_{w}^\circ$ the number of different occurrences of $w$ in $S^\circ$. Using this notation we define cyclic $k$-order entropy as:
$$H_k^\circ(w) = -\frac{1}{|w|} \sum_{w: |w|=k} |w|_{w}^{\circ} \log \left( \frac{|w|_{w}^{\circ}}{|w|^2} \right).$$

The difference between circular and standard $k$-th order entropy is that it takes into the account also the first $k$ letters of $w$. Thus it is not difficult to show that it differs from $|w|H_k(w)$ in a small amount.

\begin{lemma}
For any word $S$ and for any $k$ we have:
$$|S|H_k(S) + k \log |S| + O(k) \geq |S|H_k^\circ(S) \geq |S|H_k(S).$$
\end{lemma}
Lemma 43. Let \( |w|_v \) denote the length of a word \( w \) in an alphabet \( \Gamma \) of size \( k \). Consider the difference

\[
\sum_{a: \text{letter}} |w|^0_{va} \log \left( \frac{|w|^0_{va}}{|w|_v} \right) - \sum_{a: \text{letter}} |w|_{va} \log \left( \frac{|w|_{va}}{|w|_v} \right),
\]

our goal is to estimate it when summed over all \( v \) of length \( k \).

Define \( S_1, S_2 \) as the strings of letters that follow cyclic occurrences (standard occurrences) of \( v \) in \( w \), formally for each letter \( a \) they should satisfy

\[
|S_1|_a = |w|^0_{va} \quad |S_2|_a = |w|_{va},
\]

note that this implies that

\[
|S_1| = |w|^0_v \quad |S_2| = |w|_v.
\]

Then left and right summands from (25) are equal to, respectively:

\[
|S_1|H_0(S_1) = \sum_{a: \text{letter}} |w|^0_{va} \log \left( \frac{|w|^0_{va}}{|w|_v} \right), \quad |S_2|H_0(S_2) = \sum_{a: \text{letter}} |w|_{va} \log \left( \frac{|w|_{va}}{|w|_v} \right)
\]

We first show that (25) is positive, which immediately yields the first inequality of the Lemma.

Clearly \( |w|^0_v \geq |w|_{va} \), and so we can obtain \( S_2 \) from \( S_1 \) by removing letters, which can only decrease the entropy. Hence \( |S_1|H_0(S_1) \geq |S_2|H_0(S_2) \), which yields that (25) is positive and so the second inequality of the lemma follows.

To upper bound the difference in (25) observe that \( |S_1| \) is obtained by adding symbols to \( S_2 \) and addition of one letter to a string of length at most \( |S| - 1 \) may increase the entropy by at most \( |S| + \beta \), for some constant \( \beta \). Moreover, there are at most \( k \) such additions of symbols when summing over all possible \( k \)-length contexts \( v \). Thus the first inequality holds.

The constructed family of strings is a generalization of de Bruijn strings, which, for a given alphabet \( \Gamma \) and order \( k \), contains exactly once each word \( w \in \Gamma^k \) as a substring; de Bruijn strings have length \( |\Gamma|^k + k - 1 \) or \( |\Gamma|^k \), if treated as cyclic strings.

Lemma 43. For every \( k > 0, l \geq 1, p \geq 1 \) there exists a string \( S \) over alphabet \( \Gamma' \) of size \( \sigma = 4^p \) of length \( \sigma^{k+l} + 1 \) such that:

\( dB1 \) For any \( w \in (\Gamma')^l, l < k \) we have \( |S|_w = \sigma^{k+l+i+1} \)

\( dB2 \) For any \( w \in (\Gamma')^l, l \leq i \leq k + l + 1 \) we have either \( |S|_w = \sigma^{(k+l+i+1)/2} \) or \( |S|_w = 0 \)

\( dB3 \) no word of length \( k + l + 1 \) occurs cyclically more than once in \( S \).

Proof. Fix \( k \), by \( S_l \) we will denote the string that satisfies the conditions (dB1–dB3) for \( l \).

Let us first construct \( S_0 \). Consider cyclic de Bruijn sequence \( B = a_1a_2\cdots a_n \) of order \( 2k + 1 \) over an alphabet \( \Gamma \) of size \( \sqrt{\sigma} \) (this is well defined, as \( \sigma = 4^p \)). Observe that \( |B| = (\sqrt{\sigma})^{2k+1} = \sigma^{k+\frac{3}{2}} \). Consider two parsings of \( B^5 \):

\[
Y_B' = [a_1a_2a_3a_4] \cdots [a_{n-3}a_{n-2}a_{n-1}a_n]
\]

\[
Y_B'' = [a_2a_3a_4a_5] \cdots [a_{n-2}a_{n-1}a_{n}a_1]
\]

Now replace each pair \( a_i, a_j \) with a new symbol \( b_{i,j} \), such that \( b_{i,j} \neq b_{i',j'} \) if and only if \( (a_i, a_j) \neq (a_{i'}, a_{j'}) \). The size of the new alphabet \( \Gamma'' \) is \( \sigma = 4^p \). Consider the corresponding strings \( B_1' \) and \( B_2' \), treated in the following as cyclic words:

\[
B_1' = b_{1,2}b_{3,4} \cdots b_{n-3,n-2}b_{n-1,n}
\]

\[
B_2' = b_{2,3}b_{4,5} \cdots b_{n-2,n-1}b_{n,1}
\]
We can choose $B$ such that it begins with $a_i^{2k+1}$, for some $a_i$. Then both words $B'_1$ and $B'_2$ begin with $b_i^k$. Take $S_0 = B'_1B'_2$. Then, as the starting $k$-letters of both of them are the same, for each $v$ of length at most $k + 1$ it holds that $|B'_1|^v + |B'_2|^v = |S_0|^v$.

We now calculate $|S_0|^w$ for each possible $w$. For each $k$-letter word $w$ over $\Gamma'$ the $|B'_1|^w + |B'_2|^w$ is $\sqrt{\sigma}$. $w$ is obtained from a fixed $2k$-letter word $w \in (\Gamma)^{2k}$ and such a word occurs cyclically $\sqrt{\sigma}$-times in $B'^2$, as there are $\sqrt{\sigma}$ ways to extend $w$ to a $(2k + 1)$-letter word and each such a word occurs cyclically exactly once in $B'^2$ and each cyclic occurrence of $w$ in $B'^2$ yields one cyclic occurrence of $w'$ in exactly one of $B'^1_0$ and $B'^2_0$. Moreover, as each word $v$ of length $2k + 1$ has exactly one occurrence in $B$, the letters after different cyclic occurrence of $w \in \Gamma'$ in $B'^1_0$ or $B'^2_0$ are pairwise different. Hence, each word of length at least $k + 1$ over $\Gamma$ has at most one occurrence in $S_0$. For a word $w$ of length $i < k$ observe that each of its $\sigma^{-i}$ extensions to a $k$-letter word occurs cyclically exactly $\sqrt{\sigma}$ times in $S_0^k$, thus $w$ occurs exactly $\sigma^{k-i+1/2}$.

We now move the general case of $l > 0$. We cannot define $S_l$ as a power of $S_0$, as then (113) is violated. Instead, we will make a similar construction to the standard constructions of de Bruijn words: we will build a graph with vertices labelled with different words of length $k + l + 1$, define edges between words that can be obtained by shifting by one letter to the right and show that this graph has a Hamiltonian cycle.

Define a family of directed graphs $G_0, G_1, \ldots$, where $G_i = (V_i, E_i)$. The nodes in $V_i$ are labelled with (some) words of length $k + 1 + i$ over $\Gamma'$ (which is of size $\sigma$) and $E_i = \{ (u, u') : u[2] \ldots [u] = u'[1] \ldots [u'] - 1 \}$. We label the edge from $av$ to $vb$ with $avb$. In case of $G_0$ its vertices $V_0$ are all cyclic subwords of $S_0^0$ of length $k + 1$. Recall that given a directed graph $G$ its line graph $L(G)$ has edges of $G$ as nodes and there is an edge $(e, f)$ in $L(G)$ if and only if the end of $e$ is the beginning of $f$. Define $G_{i+1} = L(G_i)$, observe that edges of $G_i$ have labels that are words of length $k + i + 2$, those labels are reused as labels of nodes in $G_{i+1}$.

Let us state some basic properties of the defined graph: firstly, $G_0$ has in-degree and out-degree equal to $\sqrt{\sigma}$ (so it is $\sqrt{\sigma}$-regular): Given a node with a $k + 1$-letter label $w$ all its outgoing labels correspond to occurrences of the $k$-letter suffix of $w$. And by (112), each $k$ letter word has $\sqrt{\sigma}$ cyclic occurrences in $|S_0|^1$ and each $k + 1$ letter word has at most 1. So there are $\sqrt{\sigma}$ outgoing edges, each leading to a different node. Similar argument applies to the incoming edges. It is easy to show that if $G$ is $d$-regular then so is $L(G)$ and moreover if $G$ is connected then so is $L(G)$; clearly $G_0$ is connected, as $S_0$ corresponds to a Hamiltonian path in it. Thus, all $G_i$ are Eulerian. It is well-known and easy to see that an Eulerian cycle in $G$ corresponds to a Hamiltonian cycle in $L(G)$, thus each $G_i$ has a Hamiltonian cycle.

We define the word $S_1$ as the word read when traversing a Hamiltonian path in $G_1$ (note that there may be many such paths: choose one arbitrarily): we begin with an arbitrary vertex $u$ in $G_1$, write its label and when we traverse the edge $u' b$ then we append $b$ to the word. In this way we obtain a cyclic word. Note that a word $w$ of length $k + i + 1$ occurs at position $p$ if and only if $p$-th vertex on the path is labelled with $w$. Concerning the length $|S_1|$, this is exactly $|V_i| = |E_{i-1}| = \sqrt{\sigma}|V_{i-1}|$, as each $G_i$ is $\sqrt{\sigma}$ regular. Since $|V_0| = \sigma^{k+1/2}$, we conclude that $|V_i| = \sigma^{k+i+1}$. We also show that each occurrence of a word $w$ of length $k + i$ in $S_i$ is followed by a different letter, in particular this implies that a word $w'$ of length $k + i + 1$ has at most one occurrence in $S_i$, i.e. (112). We know that this is true for $G_0$, we proceed by induction. Consider all nodes labelled with $w a$ for some letter $a$ in $G_{i+1}$, where $|w| = k + i + 1$. They all correspond to edges in $G_i$ labelled with the same words. They all originate from nodes labelled with $w$ and as $|w| = k + i + 1$, by induction assumption there is exactly one such node. Now, if there were two edges outgoing edge from $w$ labelled with $w a$ then they would lead to two vertices labelled with the same label $w'$.
All considered grammar compressors are natural parsers: the starting

Example 44. For $\sigma = 4, k = 2, l = 0$:

\[ S = \text{aababbbadcebdadaadacebdcdd} \]

Observe that each $k$-letter substring occurs $\sqrt{\sigma}$ times, the letters after those occurrences are pairwise different. $H_0^\sigma(S) = H_1^\sigma(S) = \log \sigma$ and $H_2^\sigma(S) = \frac{\log \sigma}{2}$.

For $\sigma = 4, k = 1, l = 1$ the word is

\[ S = \text{abbbdadcacaabc} \]

and $H_0^\sigma(S) = \log \sigma$, $H_1^\sigma(S) = H_2^\sigma(S) = \frac{\log \sigma}{2}$.

Proof of Lemma 23. All considered grammar compressors are natural parsers: the starting string induces a parsing into expansions of nonterminals and those occur at least twice; moreover, the starting string is encoded using an entropy coder or naively, which takes $|Y_S|H_0(Y_S)$ for parsing $Y_S$.

In LZ78 each new nonterminal corresponds to a number (of some previous phrase) and a letter (last in the nonterminal), in particular after the removal of the last letter, the shortened phrase has another occurrence. The algorithm also guarantees that phrases of nonterminals are pairwise different. Using the standard encoding as pairs (previous phrase number, letter number) we obtain a prefix-free encoding, which is not better than $H_0$ coder.

The standard encoding of LZ77 does not use prefix codes but references to previous positions, so it may not satisfy the claims of the lemma. Yet some methods create LZ77 parsing and use at least $|\log |Y_S||$ bits per phrase, where $|Y_S|$ is the number of phrases. We consider non-self referencing parsing, which means that phrase $y$ and its previous occurrence in $S$ does not overlap.

Lemma 45. Let $S$ be a word from Theorem 27 for parameters $k$ and $l$, and let $z = k + l + 1$. Then for every parsing $Y_S = y_1y_2 \ldots y_{|Y_S|}$ of $S$ such that $|y_i| \leq z$ we have:

\[ |Y_S|H_0(Y_S) \geq \frac{|S|(z + k)}{2z} \log \sigma - |Y_S| \log \frac{S}{|Y_S|}. \]

Proof of Lemma 45. Let $m = |Y_S|$ and $n = |S|$.

For a word $w$ define $l_w$ as the number of occurrences of a word $w$ in parsing $Y_S$ of $S$. Clearly $l_w \leq |S|_{\sigma_l}^w$ and by construction for any $w$ such that $|S|_{\sigma_l}^w > 0$:

\[ |S|_{\sigma_l}^w = \begin{cases} \frac{n}{\sigma l} & \text{for } |w| \leq k \\ \frac{n}{\sigma l |w| + l} & \text{for } k < |w| \leq z \end{cases}. \]

thus
\[ l_w \leq \frac{w}{\rho \cdot n}, \text{ for } |w| \leq k; \]
\[ l_w \leq \frac{n}{\rho \cdot (w + n)}, \text{ for } k < |w| \leq z. \]

Define:
\[
m_1 = \sum_{w \in Y_\rho, |w| \leq k} l_w \quad m_2 = \sum_{w \in Y_\rho, |w| > k} l_w \\
n_1 = \sum_{w \in Y_\rho, |w| \leq k} |w|l_w \quad n_2 = \sum_{w \in Y_\rho, |w| > k} |w|l_w
\]

Then
\[
|Y_\rho|H_0(Y_\rho) = \sum_{w \in Y_\rho} l_w \log \frac{m}{l_w} \geq \sum_{|w| > k} l_w \log \frac{m \sigma |w|}{n} + \sum_{|w| \leq k} l_w \log \frac{m \sigma (|w| + k)/2}{n} = \sum_{w} l_w \log \frac{m}{n} + \sum_{|w| > k} l_w \log \sigma |w| + \sum_{|w| < k} l_w \log \sigma (|w| + k)/2
\]
\[
= m \log \frac{m}{n} + \sum_{|w| > k} l_w |w| \log \sigma + \sum_{|w| < k} \frac{l_w |w|}{2} \log \sigma + \sum_{|w| > k} \frac{l_w k}{2} \log \sigma
\]
\[
= m \log \frac{m}{n} + n_1 \log \sigma + n_2 \frac{2}{2} \log \sigma + \frac{m_2 k}{2} \log \sigma \geq m \log \frac{m}{n} + n_1 \log \sigma + n_2 \frac{2}{2} \log \sigma + \frac{n k}{2z} \log \sigma
\]
\[
\geq \frac{n(z + k)}{2z} \log \sigma - m \log \frac{m}{n}.
\]

**Theorem 46** (Full version of Theorem 24). Let \( A \) be a natural parser. Let \( k \) be a non-negative and integer function of \( |S| \) and \( \sigma \) satisfying, for every \( \sigma \), \( \lim \sup_{|S| \to \infty} \frac{k_{|S|, \sigma}}{\log_{\rho} |S|} < 1 \), where \( k_{|S|, \sigma} \) denotes value of \( k \) for \( |S| \) and \( \sigma \). Then for any \( p > 0 \) there exist infinite family of strings \( S \in \Gamma^* \), where \( |\Gamma| = 4^p \), of increasing length such that the bit-size of output \( A(S) \) of \( A \) on \( S \) is at least:
\[
A(S) \geq |S|H_k(S) + \frac{\rho |S| (|\log \sigma - 2\lambda|)}{2} \geq (1 + \rho) |S|H_k(S) - \lambda |S|.
\]

Moreover, if the size of parsing induced by \( A \) is \( o(|S|) \) then:
\[
A(S) \geq |S|H_k(S) + \frac{\rho |S| |\log \sigma|}{2} - o(|S|) \geq (1 + \rho) |S|H_k(S) - o(|S|),
\]
where \( \rho = \frac{l}{2 \log_{\rho} |S| - k} \) and \( \lambda < 0.54 \).

**Proof of Theorem 24** The proof is a straightforward application of Lemma 13. Fix alphabet \( \Gamma \) of size \( \sigma = 4^p \). By assumption that \( \lim \sup_{|S| \to \infty} \frac{k_{|S|, \sigma}}{\log_{\rho} |S|} < 1 \) for large enough \( |S| \) the \( l = 2 \log_{\rho} |S| - 2k_{|S|, \sigma} - 1 \) is positive. Then \( k_{|S|, \sigma} + \frac{4}{l} = \log_{\rho} |S| \). So it is possible to construct a word \( S \) from Theorem 20 for parameters \( k_{|S|, \sigma}, l, p \).
Let $Y_S = y_1 y_2 \ldots y_{Y_S}$ be a parsing of $S$ induced by $A$. As $|y_i| \leq k + \frac{l+1}{2}$ we use Lemma 45 to lower bound the output of the algorithm:

$$A(S) \geq |Y_S| H_0(Y_S)$$

$$\geq \frac{|S|(2k + l + 1)}{2(k + l + 1)} \log \sigma - m \log \frac{n}{m}$$

$$\geq |S| H_k(S) + \frac{|S|k}{2(k + l + 1)} \log \sigma - m \log \frac{n}{m}$$

$$\geq |S| H_k(S) + \frac{k|S| \log \sigma}{2} - m \log \frac{n}{m}$$

$$\geq (1 + \rho)|S| H_k(S) - m \log \frac{n}{m}.$$

Similarly as in the proof Lemma 28 we can bound $m \log \frac{n}{m}$ by $n \log e < 0.54 n$, and by $o(n)$ if $m = o(n)$. Plugging this values to the above equation yields the claim.

**Proof of corollaries 25, 26, 27.** For any function $k$ and any $\sigma = 4^p, p > 0$ by Theorem 24 we can build infinite family of words $S$ such that output generated by any natural parsing method will be lower bounded by:

$$A(S) \geq |S| H_k(S) + \frac{|S|k(\log \sigma - O(1))}{2(2 \log \sigma |S| - k)}$$

(26)

Above inequality holds for any $\sigma = 4^p$, and length of $S$ can be arbitrarily large, thus we can ignore the $O(1)$ factor.

For Corollary 25 observe that as $k = o(\log \sigma n)$ the second summand is $\Omega\left(\frac{n k \log \sigma}{\log n}\right)$.

In the case of Corollary 26 substituting $k = \alpha \log \sigma n$ yields that right hand side becomes

$$\frac{2^\alpha}{2^{k-\alpha}} |S| H_k(S),$$

moreover as $|S| H_k(S) = \Theta(n \log \sigma)$ and $\frac{2^\alpha}{2^{k-\alpha}} > 1$ the claim holds.

For the Corollary 27 by Theorem 24 for any $k, p > 0$ we can build a word with parameters $k, l = 0$ and $p$. Then the mean of entropies of such word is at most:

$$\frac{1}{j} \sum_{i=0}^{j-1} H_i(S) \leq \log \sigma$$

And as by construction $k = \log \sigma |S| - \frac{1}{2}$ we have:

$$A(S) \geq |S| H_k(S) + \frac{|S| \log \sigma |S| - \frac{1}{2})(\log \sigma - O(1))}{2(2 \log \sigma |S| - \log \sigma |S| + \frac{1}{2})}$$

$$\geq |S|(\log \sigma - O(1)) - O\left(\frac{|S| \log \sigma}{\log \sigma |S|}\right)$$

\[\blacksquare\]