Lempel-Ziv: a “one-bit catastrophe”
but not a tragedy

Guillaume Lagarde ∗       Sylvain Perifel †

August 1, 2017

Abstract

The so-called “one-bit catastrophe” for the compression algorithm LZ’78 asks whether the compression ratio of an infinite word can change when a single bit is added in front of it. We answer positively this open question raised by Lutz and others: we show that there exists an infinite word $w$ such that $\rho_{\text{sup}}(w) = 0$ but $\rho_{\text{inf}}(0w) > 0$, where $\rho_{\text{sup}}$ and $\rho_{\text{inf}}$ are respectively the lim sup and the lim inf of the compression ratios $\rho$ of the prefixes (Theorem 2.6).

To that purpose we explore the behaviour of LZ’78 on finite words and show the following results:

- There is a constant $C > 0$ such that, for any finite word $w$ and any letter $a$, $\rho(aw) \leq C\sqrt{\rho(w)\log |w|}$. Thus, sufficiently compressible words ($\rho(w) = o(1/\log |w|))$ remain compressible with a letter in front (Theorem 2.7);

- The previous result is tight up to a multiplicative constant for any compression ratio $\rho(w) = O(1/\log |w|)$ (Theorem 2.10). In particular, there are infinitely many words $w$ satisfying $\rho(w) = O(1/\log |w|)$ but $\rho(0w) = \Omega(1)$.

1 Introduction

Suppose you compressed a file using your favorite compression algorithm, but you realize there were a typo that makes you add a single bit to the original file. Compress it again and you get a much larger compressed file, for a one-bit difference only between the original files. Most compression algorithms fortunately do not have this strange behaviour; but if your favorite compression algorithm is called LZ’78, one of the most famous and studied of them, then this surprising scenario might well happen. . . In rough terms, that is what

∗Univ Paris Diderot, Sorbonne Paris Cité, IRIF, UMR 7089 CNRS, F-75205 Paris, France. Email: guillaume.lagarde@irif.fr.
†Univ Paris Diderot, Sorbonne Paris Cité, IRIF, UMR 7089 CNRS, F-75205 Paris, France. Email: sylvain.perifel@irif.fr.
we show in this paper, thus closing a question advertised by Jack Lutz under
the name “one-bit catastrophe” and explicitly stated for instance in papers of
Lathrop and Strauss [4], Pierce II and Shields [11], as well as more recently by
López-Valdés [6].

Ziv-Lempel algorithms

In the paper [14] where they introduce their second compression algorithm
LZ’78, Ziv and Lempel analyse its performance in terms of finite-state lossless
compressors and show it achieves the best possible compression ratio. Together
with its cousin algorithm LZ’77 [13], this generic lossless compressor has paved
the way to many dictionary coders, some of them still widely used in practice
today. For instance, the deflate algorithm at the heart of the open source com-
pression program gzip uses a combination of LZ77 and Huffman coding; or the
image format GIF is based on a version of LZ’78. As another example, methods
for efficient access to large compressed data on internet based on Ziv-Lempel
algorithms have been proposed [2].

Besides its practical interest, the algorithm LZ’78 was the starting point of a
long line of theoretical research, triggered by the optimality result among finite-
state compressors proved by Ziv and Lempel. In recent work, for instance, a
comparison of pushdown finite-state compressors and LZ’78 is made in [10]; the
article [3] studies Lempel-Ziv and Lyndon factorisations of words; or the efficient
construction of absolutely normal numbers of [9] makes use of the Lempel-Ziv
parsing.

Some works of bioinformatics have also focussed on Ziv-Lempel algorithms,
since their compression scheme makes use of repetitions in a sequence in a way
that proves useful to study DNA sequences (see e.g. [12]), or to measure the
complexity of a discrete signal [1] for instance.

Actually, both in theory and in practice, Ziv-Lempel algorithms are un-
doubtedly among the most studied compression algorithms and we have chosen
only a very limited set of references: we do not even claim to be exhaustive in
the list of fields where LZ’77 or LZ’78 play a role.

Robustness

Yet, the robustness of LZ’78 remained unclear: the question of whether the
compression ratio of a sequence could vary by changing a single bit appears
already in [4], where the authors also ask how LZ’78 will perform if a bit is added
in front of an optimally compressible word. Since the Hausdorff dimension of
complexity classes introduced by Lutz [8] can be defined in terms of compression
(see [7]), this question is linked to finite-state and polynomial-time dimensions
as [6] shows. As a practical illustration of the issue the (lack of) robustness can
cause, let us mention that the deflate algorithm tries several starting points
for its parsing in order to improve the compression ratio.

In this paper, we show the existence of an infinite sequence $w$ which is
compressible by LZ’78, but the addition of a single bit in front of it makes it
incompressible (the compression ratio of \(0w\) is non-zero, see Theorem 2.6), thus we settle the “one-bit catastrophe” question. To that end, we study the question over finite words, which enable stating more precise results. For a word \(w\) and a letter \(a\), we first prove in Theorem 2.7 that the compression ratio \(\rho(aw)\) of \(aw\) cannot deviate too much from the compression ratio \(\rho(w)\) of \(w\):

\[
\rho(aw) \leq 3\sqrt{2}\sqrt{\rho(w) \log |w|}.
\]

In particular, \(aw\) can only become incompressible (\(\rho(aw) = \Theta(1)\)) if \(w\) is already poorly compressible, namely \(\rho(w) = \Omega(1/\log n)\). This explains why the one-bit catastrophe cannot be “a tragedy” as we point out in the title.

However, our results are tight up to a constant factor, as we show in Theorem 2.10: there are constants \(\alpha, \beta > 0\) such that, for any \(l(n) \in [90^2 \log^2 n, \sqrt{n}]\), there are infinitely many words \(w\) satisfying

\[
\rho(w) \leq \alpha \frac{\log |w|}{l(|w|)} \quad \text{whereas} \quad \rho(0w) \geq \beta \frac{\log |w|}{\sqrt{l(|w|)}}.
\]

In particular, for \(l(n) = 90^2 \log^2 n\), these words satisfy

\[
\rho(w) \leq \frac{1}{\log |w|} \quad \text{and} \quad \rho(0w) \geq \frac{\beta}{90}
\]

(this is the one-bit catastrophe over finite words). But actually the story resembles much more a tragedy for well-compressible words. Indeed, for \(l(n) = \sqrt{n}\) we obtain:

\[
\rho(w) \leq \alpha \frac{\log |w|}{\sqrt{|w|}} \quad \text{whereas} \quad \rho(0w) \geq \beta \frac{\log |w|}{|w|^{1/4}},
\]

that is to say that the compression ratio of \(0w\) is much worse than that of \(w\) (which in that case is optimal). To give a concrete idea, the bounds given by our Theorem 4.1 for words of size 1 billion (\(|w| = 10^9\)) yield a compression for \(w\) of size at most \(d \log d \leq 960,000\) (where \(d = 1.9\sqrt{|w|}\)), whereas for \(0w\) the compression size is at least \(d' \log d' \geq 3,800,000\) (where \(d' = 0.039|w|^{1/4}\)).

This “catastrophe” shows that LZ’78 is not robust with respect to the addition or deletion of bits. Since a usual good behaviour of functions used in data representation is a kind of “continuity”, our results show that, in this respect, LZ’78 is not a good choice, as two words that differ in a single bit can have images very far apart.

**Organization of the paper**

In Section 2 we introduce all the notions related to LZ’78 and state our main results (Section 2.3). Section 3 is devoted to the proof of the upper bound (the

---

1Actually, throughout the paper we preferred readability over optimality and thus did not try to get the best possible constants; simulations show that there is a lot of room for improvement, since already for small words the difference is significant (using notations introduced in Sections 2 and 4, for \(w = \text{Pref}(x)\) with \(x \in \text{DB}(12)\), \(|w| \approx 8.10^6\) and \(w\) is parsed in about 4100 blocks, whereas \(0w\) is parsed in more than 200,000 blocks).
“not a tragedy” part), whereas the rest of the paper is about lower bounds. In Section 4 we explicitly give a word, based on de Bruijn sequences, whose compression ratio is optimal but the addition of a single bit deteriorates the compression ratio as much as the aforementioned upper bounds allows to. That is a particular case of the result of Section 5 but we include it anyway for three reasons: it illustrates the main ideas without obscuring them with too many technical details; the construction is more explicit; and the bounds are better.

In Section 5 we prove our main theorem on finite words (Theorem 2.10). It requires the existence of a family of “de Bruijn-style” words shown in Section 5.1 thanks to the probabilistic method. Finally, Section 6 uses the previous results to prove the “original” one-bit catastrophe, namely on infinite words (Theorem 2.6).

2 Lempel-Ziv, compression and results

Before turning to the description of LZ’78 algorithm, let us recall standard notations on words.

2.1 Basic notations

The binary alphabet is the set \{0, 1\}. A word \(w\) is an element of \(\{0, 1\}^*\), that is, a finite ordered sequence of letters 0 or 1, whose length is denoted by \(|w|\). The empty word is denoted by \(\lambda\). For a word \(w = x_0 \cdots x_{n-1}\) (note that the indices begin at zero), where \(x_i \in \{0, 1\}\), \(w[i..j]\) will denote the substring \(x_i \cdots x_j\) of \(w\) (or \(\lambda\) if \(j < i\)); \(w[i]\) or \(w_i\) will denote the letter \(x_i\); and \(w_{\leq i}\) (respectively \(w_{<i}\)) will denote \(w[0..i]\) (resp. \(w[0..i-1]\)). We say that a word \(m\) is a factor of \(w\) if \(m\) is any substring \(w[i..j]\). In the particular case of \(i = 0\) (respectively \(j = n - 1\)), \(m\) is also called a prefix (resp. a suffix) of \(w\). The set of factors of \(w\) is denoted by \(\mathcal{F}(w)\), and its set of prefixes \(\mathcal{P}(w)\). By extension, for a set \(M\) of words, \(\mathcal{F}(M)\) will denote \(\bigcup_{w \in M} \mathcal{F}(w)\) and similarly for \(\mathcal{P}(M)\). If \(u\) and \(w\) are two words, we denote by \(\text{Occ}_w(u)\) the number of occurrences of the factor \(u\) in \(w\).

The “length-lexicographic order” on words is the lexicographic order where lengths are compared first.

An infinite word is an element of \(\{0, 1\}^\mathbb{N}\). The same notations as for finite words apply.

All logarithms will be in base 2. The size of a finite set \(A\) is written \(|A|\).

2.2 LZ’78

2.2.1 Notions relative to LZ

A k-partition (or just partition) of a word \(w\) is a sequence of \(k\) non-empty words \(m_1, \ldots, m_k\) such that \(w = m_1.m_2.\cdots.m_k\). The LZ-parsing (or just parsing) of a word \(w\) is the unique partition of \(w = m_1 \cdots m_k\) such that:
• \(m_1, \ldots, m_{k-1}\) are all distinct\(^2\).

• \(\forall i \leq k, \mathcal{P}(m_i) \subseteq \{m_1, \ldots, m_i\}\).

The words \(m_1, \ldots, m_k\) are called \textit{blocks}. The \textit{predecessor} of a block \(m_i\) is the unique \(m_j, j < i\), such that \(m_i = m_ja\) for a letter \(a\). The compression algorithm LZ’78 parses the word \(w\) and encodes each block \(m_i\) as a pointer to its predecessor \(m_j\) together with the letter \(a\) such that \(m_i = m_ja\). For instance, the word \(w = 00010110100001\) is parsed as

| Blocks | 0 | 00 | 1 | 01 | 10 | 100 | 001 |
|--------|---|----|---|----|----|-----|-----|
| Block number | 0 | 1 | 2 | 3 | 4 | 5 | 6 |

and thus encoded as

\[(\lambda, 0); (0, 0); (\lambda, 1); (0, 1); (2, 0); (4, 0); (1, 1).\]

The \textit{dictionary} of \(w\) is the set \(\text{Dic}(w) = \{m_1, \ldots, m_k\}\) (in the example, \(\{0, 1, 00, 01, 10, 001, 100\}\)). Remark that, by definition, \(\{\lambda\} \cup \text{Dic}(w)\) is prefix-closed.

The \textit{parsing tree} of \(w\) is the unique rooted binary tree \(T(w)\) whose \((k + 1)\) vertices are labeled with \(\lambda, m_1, \ldots, m_k\), such that the root is \(\lambda\) and if a vertex \(m_i\) has a left child, then it is \(m_i0\), and if it has a right child, then it is \(m_i1\)\(^3\). See Figure 1. Remark also that the depth of a vertex is equal to the size of the corresponding block.

![Figure 1: Parsing tree of 00010110100001.](image)

By abuse of language, we say that a block \(b\) “increases” or “grows” in the parsing of a word \(w\) when we consider one of its successors, or when we consider a path from the root to the leaves that goes through \(b\). Indeed, going from \(b\) to its successor amounts to add a letter at the end of \(b\) (hence the “increase”).

\(^2\)The last word \(m_k\) might be equal to another \(m_i\).

\(^3\)Note that, in order to recover the parsing from the parsing tree, the vertices must also be labeled by the order of apparition of each block, but we do not need that in the sequel.
2.2.2 Compression ratio

As in the example above, given a word \( w \) and its LZ-parsing \( m_1 \cdots m_k \), the LZ-compression of \( w \) is the ordered list of \( k \) pairs \((p_i, a_i)\), where \( p_i \) is the binary representation of the unique integer \( j < i \) such that \( m_j = m_i[0..(|m_i| - 2)] \), and \( a_i \) the last letter of \( m_i \) (that is, the unique letter such that \( m_i = m_ja_i \)). When the LZ-compression is given, one can easily reconstruct the word \( w \).

Remark 2.1. • If \( x \) is a word, we define \( \text{Pref}(x) \) the concatenation of all its prefixes in ascending order, that is,

\[
\text{Pref}(x) = x_0.x_0x_1.x_0x_1x_2.\cdots.x_0\cdots x_{n-2}x_{n-1}.
\]

Then the parsing of the word \( w = \text{Pref}(x) \) is exactly the prefixes of \( x \), thus the size of the blocks increases each time by one: this is the optimal compression. In that case, the number of blocks is

\[
k = |x| = \sqrt{2} \sqrt{|w|} - O(1).
\]

Actually, it is easy to see that this optimal compression is attained only for the words \( w \) of the form \( \text{Pref}(x) \).

In Section 2 we will need the concatenation of all prefixes of \( x \) starting from a size \( p + 1 \), denoted by \( \text{Pref}_{>p}(x) \), that is,

\[
\text{Pref}_{>p}(x) = x_0x_1\cdots x_p.x_0x_1\cdots x_{p+1}\cdots x_0\cdots x_{n-1}.
\]

• On the other hand, if \( w \) is the concatenation, in length-lexicographic order, of all words of size \( \leq n \) (\( w = 0.1.00.01.10.11.000.001\ldots \)), then it has size

\[
|w| = \sum_{i=1}^{n} i2^i = (n - 1)2^{n+1} + 2,
\]

and its parsing consists of all the words up to size \( n \), therefore that is the worst possible case and the number of blocks is

\[
k = 2^{n+1} - 2 = \frac{|w|}{\log |w|} + O\left(\frac{|w|}{\log^2 |w|}\right).
\]

(And that is clearly not the only word achieving this worst compression.)

The number of bits needed in the LZ-compression is \( \Theta(\sum_{i=1}^{k}(|p_i| + 1)) = \Theta(k \log k) \). As the two previous extremal cases show, \( k \log k = \Omega(\sqrt{|w| \log |w|}) \) and \( k \log k = O(|w|) \).

Definition 2.2. The compression ratio of a word \( w \) is

\[
\rho(w) = \frac{|\text{Dic}(w)| \log |\text{Dic}(w)|}{|w|}.
\]
As Remark 2.1 shows,

$$\rho(w) = \Omega\left(\frac{\log|w|}{\sqrt{|w|}}\right) \text{ and } \rho(w) \leq 1 + O\left(\frac{1}{\log|w|}\right).$$

A sequence of words \((w_n)\) is said LZ-compressible if \(\rho(w_n)\) tends to zero (i.e. \(k_n \log k_n = o(|w_n|)\)), and consistently it will be considered LZ-incompressible if \(\liminf_{n \to \infty} \rho(w_n) > 0\) (in other terms, \(k_n \log k_n = \Omega(|w_n|)\)). Actually, the \((\log k)\) factor is not essential in the analysis of the algorithm, therefore we drop it in our definitions (moreover, most of the time we will focus directly on the size of the dictionary rather than the compression ratio).

**Definition 2.3.** The size of the LZ-compression of \(w\) (or compression size, or also compression speed when speaking of a sequence of words) is defined as the size of \(\text{Dic}(w)\), that is, the number of blocks in the LZ-parsing of \(w\).

Remark that \(|\text{Dic}(w)| = \Omega(\sqrt{|w|})\) and \(|\text{Dic}(w)| = O(|w|/\log(|w|))\). We can now restate the definition of incompressibility of a sequence of words in terms of compression speed instead of the number of bits in the LZ-compression.

**Definition 2.4.** A sequence of words \((w_n)\) is said incompressible iff

$$|\text{Dic}(w_n)| = \Theta\left(\frac{|w_n|}{\log(|w_n|)}\right).$$

In those definitions, we have to speak of sequences of finite words since the asymptotic behaviour is considered. That is not needed anymore for infinite words, of course, but then two notions of compression ratio are defined, depending on whether we take the \(\liminf\) or \(\limsup\) of the compression ratios of the prefixes.

**Definition 2.5.** Let \(w \in \{0,1\}^\mathbb{N}\) be an infinite word.

$$\rho_{\text{inf}}(w) = \liminf_{n \to \infty} \rho(w_{<n}) \text{ and } \rho_{\text{sup}}(w) = \limsup_{n \to \infty} \rho(w_{<n}).$$

### 2.3 One-bit catastrophe and results

The one-bit catastrophe question is originally stated only on infinite words. It asks whether there exists an infinite word \(w\) whose compression ratio changes when a single letter is added in front of it. More specifically, a stronger version asks whether there exists an infinite word \(w\) compressible (compression ratio equal to 0) for which \(0w\) is not compressible (compression ratio > 0). At Section 6 we will answer positively that question:

**Theorem 2.6.** There exists \(w \in \{0,1\}^\mathbb{N}\) such that

$$\rho_{\text{sup}}(w) = 0 \text{ and } \rho_{\text{inf}}(w) \geq \frac{1}{6075}.$$
Remark that the lim inf is considered for the compression ratio of $0w$ and the lim sup for $w$, which is the hardest possible combination as far as asymptotic compression ratios are concerned.

But before proving this result, most of the work will be on finite words (only in Section 6 will we show how to turn to infinite words). Let us therefore state the corresponding results on finite words. Actually, on finite words we can have much more precise statements and therefore the results are interesting on their own (perhaps even more so than the infinite version).

In Section 3, we show that the compression ratio of $aw$ cannot be much more than that of $w$. In particular, all words “sufficiently” compressible (compression speed $o(|w|/\log^2 |w|)$) cannot become incompressible when a letter is added in front (in some sense, thus, the one-bit catastrophe cannot happen for those words, see Remark 2.11).

**Theorem 2.7.** For all word $w \in \{0,1\}^*$ and any letter $a \in \{0,1\}$,

$$|\text{Dic}(aw)| \leq 3\sqrt{|w||\text{Dic}(w)|}.$$  

**Remark 2.8.** When stated in terms of compression ratio, using the fact that $|\text{Dic}(w)| \geq \sqrt{|w|}$, this result reads as follows:

$$\rho(aw) \leq 3\sqrt{2}\rho(w) \log |w|.$$  

We also show in Section 4 that this result is tight up to a multiplicative constant, since Theorem 4.1 implies the following result.

**Theorem 2.9.** For an infinite number of words $w \in \{0,1\}^*$,

$$|\text{Dic}(0w)| \geq \frac{1}{35} \sqrt{|w||\text{Dic}(w)|}.$$  

More generally, we prove in Section 5 our main result:

**Theorem 2.10.** Let $l : \mathbb{N} \rightarrow \mathbb{N}$ be a function satisfying $l(n) \in [(90 \log n)^2, \sqrt{n}]$. Then for an infinite number of words $w$:

$$|\text{Dic}(w)| \leq \frac{3 + \sqrt{3}}{2} \cdot \frac{|w|}{l(|w|)} \quad \text{and} \quad |\text{Dic}(0w)| \geq \frac{1}{54} \cdot \frac{|w|}{\sqrt{l(|w|)}}.$$  

This shows that the upper bound is tight (up to a multiplicative constant) for any possible compression speed. This also provides an example of compressible words that become incompressible when a letter is added in front (see Remark 2.11), thus showing the one-bit catastrophe for finite words.

**Remark 2.11.** In particular:

- **Theorem 2.7** implies that, if an increasing sequence of words $(w_n)$ satisfies $|\text{Dic}(w_n)| = o(|w_n|/\log^2 |w_n|)$, then for any letter $a \in \{0,1\}$, $aw_n$ remains fully compressible ($|\text{Dic}(aw_n)| = o(|w_n|/\log |w_n|)$);
• however, by Theorem 2.10 there is an increasing sequence of words \((w_n)\) such that \(|\text{Dic}(w_n)| = \Theta(|w_n|/\log^2|w_n|)\) (compressible) but \(|\text{Dic}(0w_n)| = \Theta(|w_n|/\log|w_n|)\) (incompressible), which is the one-bit catastrophe on finite words;

• the following interesting case is also true: there is an increasing sequence of words \((w_n)\) such that \(|\text{Dic}(w_n)| = \Theta(\sqrt{|w_n|})\) (optimal compression) but \(|\text{Dic}(0w_n)| = \Theta(|w_n|^{3/4})\). This special case is treated extensively in Theorem 4.7.

2.4 Parsings of \(w\) and \(aw\)

We will often compare the parsing of a word \(w\) and the parsing of \(aw\) for some letter \(a\): let us introduce some notations (see Figure 2).

• The blocks of \(w\) will be called the green blocks.

• The blocks of \(aw\) will be called the red blocks and are split into two categories:

  – The junction blocks, which are red blocks that overlap two or more green blocks when we align \(w\) and \(aw\) on the right (that is, the factor \(w\) of \(aw\) is aligned with the word \(w\), see Figure 2).

  – The offset-i blocks, starting at position \(i\) in a green block and completely included in it. If not needed, the parameter \(i\) will be omitted.

\[
\begin{array}{cccccccccccc}
0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\
\end{array}
\]

offset-1

\begin{array}{cccccccccccc}
0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\
\end{array}

offset-0

junction

Figure 2: The green blocks of \(w\) and red blocks of \(0w\) for \(w = 001010100011\).

3 Upper bound

This section is devoted to the proof of Theorem 2.7 giving an upper bound on the compression ratio of \(aw\), for any letter \(a\), as a function of the compression ratio of the word \(w\). In their 1998 paper [4], Lathrop and Strauss ask the following question: “Consider optimally compressed sequences: Will such sequences compress reasonably well if a single bit is removed or added to the front of the sequence?” We give a positive and quantified answer: indeed, a word \(w\) compressed optimally has a compression speed \(O(\sqrt{n})\), thus by Theorem 2.7, the

\[^{4}\text{Except the first block of }aw, \text{ which is the word }a \text{ and which is just called a red block.}\]
word $aw$ has a compression speed $O(n^{3/4})$. (And we shall complete this answer with the matching lower bound in the next section.)

The first lemma bounds the size of the partition of a word $w$ if the partitioning words come from a family with a limited number of words of same size. In its application, the partition will be a subset of the LZ-parsing, and Lemma 3.3 below will give the required bound on the number of factors of a given size.

**Lemma 3.1.** Let $F$ be a family of distinct words such that for each $i$, the number of words of size $i$ in $F$ is bounded by a constant $N$. Suppose that a word $w$ is partitioned into different words of $F$. Then the number of words used in the partition is at most $2\sqrt{N|w|}$.

**Proof.** Let $m(i)$ be the number of words of size $i$ occurring in the partition of $w$, and $k$ the size of the largest words used. We want to prove that

$$\sum_{i=1}^{k} m(i) \leq 2\sqrt{N|w|}.$$ 

We have:

$$|w| = \sum_{i=1}^{k} im(i) \geq \sum_{i \geq \sqrt{\frac{|w|}{N}}} im(i) \geq \sqrt{\frac{|w|}{N}} \sum_{i \geq \sqrt{\frac{|w|}{N}}} m(i)$$

hence

$$\sum_{i \geq \sqrt{\frac{|w|}{N}}} m(i) \leq \sqrt{N|w|}.$$ 

On the other hand, since $m(i) \leq N$:

$$\sum_{i < \sqrt{\frac{|w|}{N}}} m(i) < N\sqrt{\frac{|w|}{N}} = \sqrt{N|w|}. \quad \square$$

**Remark 3.2.** Note that if, for all $i \geq 1$, $F$ contains exactly $\min(2^i, N)$ words of size $i$, the concatenation of all the words of $F$ up to size $s$ gives a word $w$ of size

$$|w| = \sum_{i=1}^{\log N} i2^i + \sum_{i>\log N} iN \leq 2N \log N + (s - \log N)(s + \log N + 1)N/2$$

partitioned into $m$ blocks, where

$$m = \sum_{i=1}^{\log N} 2^i + \sum_{i>\log N} N \geq (s - \log N)N.$$ 

Thus $m \geq \sqrt{2\sqrt{N|w|}}$ if $s >> \log N$. This shows the optimality of Lemma 3.1 up to a factor $\sqrt{2}$. 

10
We now come to the lemma bounding the number of factors of a given size in a word \( w \) as a function of its LZ-parsing.

**Lemma 3.3.** Let \( T \) be the parsing tree of a word \( w \). Then the number of different factors of size \( i \) in the blocks of \( w \) is at most \( |T| - i \) (that is, \(|\mathcal{F}(\text{Dic}(w)) \cap \{0, 1\}^i| \leq |T| - i\)).

**Proof.** A factor of size \( i \) in a block \( b \) corresponds to a subpath of size \( i \) in the path from the root to \( b \) in the parsing tree. The number of such subpaths is bounded by the number of vertices at depth at least \( i \).

Actually, below we will use Lemma 3.3 sub-optimally since we will ignore the parameter \( i \) and use the looser bound \(|T| - 1\).

Let us turn to the proof of Theorem 2.7, the main result of the present section.

**Proof of Theorem 2.7.** Let \( D = \text{Dic}(aw) \) be the set of red blocks. We partition \( D \) into \( D_1 \) and \( D_2 \), where \( D_1 \) is the set of junction blocks together with the first red block (consisting only of the letter \( a \)), and \( D_2 \) is the set of offset blocks.

- **Bound for \( D_1 \):** The number of junction blocks is less than the number of green blocks, therefore \(|D_1| \leq |\text{Dic}(w)| \leq \sqrt{|\text{Dic}(w)| \cdot |w|} \) (recall that \(|\text{Dic}(w)| \leq |w|\)).

- **Bound for \( D_2 \):** Consider \( \tilde{w} \) the word \( w \) where all the junction blocks have been replaced by the empty word \( \lambda \). We know that \( \tilde{w} \) is partitioned into different words by \( D_2 \). But \( D_2 \subset \mathcal{F} \), where \( \mathcal{F} = \mathcal{F}(\text{Dic}(w)) \) (the set of factors contained in the green blocks). By Lemma 3.3 the number of words of size \( i \) in \( \mathcal{F} \) is bounded by \(|\mathcal{F}(w)| - i \), which is at most \(|\text{Dic}(w)|\). Finally, Lemma 3.1 tells us that the number of words in any partition of \( \tilde{w} \) by words of \( \mathcal{F} \) is bounded by \( 2\sqrt{|\text{Dic}(w)| \cdot |\tilde{w}|} \leq 2\sqrt{|\text{Dic}(w)| \cdot |w|} \).

In the end, \(|D| = |D_1| + |D_2| \leq 3\sqrt{|w| \cdot |\text{Dic}(w)|} \).

**Remark 3.4.** Instead of a single letter, we can add a whole word \( z \) in front of \( w \). With the same proof, it is easy to see that

\[ |\text{Dic}(zw)| \leq |\text{Dic}(z)| + 3\sqrt{|w| \cdot |\text{Dic}(w)|} \]

Alternately, if we remove the first letter of \( w = aw' \) (or any prefix) we get the same upper bound:

\[ |\text{Dic}(w')| \leq 3\sqrt{|aw'| \cdot |\text{Dic}(aw')|} \].

11
4 “Weak catastrophe” for the optimal compression ratio

Before the proof of Theorem 2.10, we first present a “weak catastrophe”, namely
the third item of Remark 2.11 in which the compression speed of a sequence
changes from $O(\sqrt{n})$ (optimal compression) to $\Omega(n^{3/4})$ when a letter is added
in front, thus matching the upper bound of Theorem 2.7.

Theorem 4.1. For an infinite number of words $w$:

$$|\text{Dic}(w)| \leq 1.9 \sqrt{|w|} \text{ and } |\text{Dic}(0w)| \geq 0.039|w|^{3/4}.$$  

Remark 4.2. The “true” values of the constants that we will get below are as
follows:

$$|\text{Dic}(w)| \leq 3 \sqrt{\frac{2}{5}} \sqrt{|w|} \text{ and } |\text{Dic}(0w)| \geq \frac{1}{36} \left(\frac{8}{5}\right)^{3/4}|w|^{3/4} - o(|w|^{3/4}).$$

Observe that this weak catastrophe is a special case of Theorem 2.10 (with
better constants, though). The aim of this section is twofold: first, it will be a
constructive proof, whereas the main theorem will use the probabilistic method;
second, this section will set up the main ideas and should help understand the
general proof.

A main ingredient in the construction is de Bruijn sequences, that we intro-
duce shortly before giving the overview of the proof.

4.1 De Bruijn sequences

A de Bruijn sequence of order $k$ (or DB($k$) in short, notation that will also
designate the set of all de Bruijn sequences of order $k$) is a word $x$ of size
$2^k + k - 1$ in which every word of size $k$ occurs exactly once as a substring. For
instance, 00101100 is an example of a DB(3). Such words exist for any order
$k$ as they are, for instance, Eulerian paths in the regular directed graph whose
vertices are words of size $(k - 1)$ and where there is an arc labeled with letter $a$
from $u$ to $v$ iff $v = u[1..k - 2]a$.

Given any $x \in \text{DB}(k)$, the following well-known (and straightforward) property holds:

$(\star)$ Any word $u$ of size at most $k$ occurs exactly $2^{k-|u|}$ times in $x$.

(In symbols, $\text{Occ}_x(u) = 2^{k-|u|}$.) Thus, a factor of size $l \leq k$ in $x$ will identify exactly
$2^{k-l}$ positions in $x$ (the $i$-th position is the beginning of the $i$-th occurrence
of the word).

The use of de Bruijn sequences is something common in the study of this
kind of algorithms: Lempel and Ziv themselves use it in [5], as well as later [11]
and [13] for example.
4.2 Overview of the proof

Recall that a word \( w \) is optimally compressed iff it is of the form \( w = \text{Pref}(x) \) for some word \( x \) (Remark 2.1). Thus we are looking for an \( x \) such that \( 0 \text{Pref}(x) \) has the worst possible compression ratio. In Section 3 the upper bound on the dictionary size came from the limitation on the number of possible factors of a given size: it is therefore natural to consider words \( x \) where the number of factors is maximal, that is, de Bruijn sequences.

Although we conjecture that the result should hold for \( w = \text{Pref}(x) \) whenever \( x \) is a de Bruijn sequence beginning with 0, we were not able to show it directly. Instead, we need to (possibly) add small words, that we will call “gadgets”, between the prefixes of \( x \).

For some arbitrary \( k \), we fix \( x \in \text{DB}(k) \) and start with the word \( w = \text{Pref}(x) \) of size \( n \). The goal is to show that there are \( \Omega(n^{3/4}) \) red blocks (i.e that the size of the dictionary for \( 0w \) is \( \Omega(n^{3/4}) \)): this will be achieved by showing that a significant (constant) portion of the word \( 0w \) is covered by “small” red blocks (of size \( O(n^{1/4}) \)). Let \( s = |x| \), so that \( n = \Theta(s^2) \). More precisely, we show that, in all the prefixes \( y \) of \( x \) of size \( \geq 2s/3 \), at least the last third of \( y \) is covered by red blocks of size \( O(\sqrt{s}) = O(n^{1/4}) \).

This is done by distinguishing between red blocks starting near the beginning of a green block (offset-\( i \) for \( i \leq \gamma k \)) and red blocks starting at position \( i > \gamma k \):

- For the first, what could happen is that by coincidence the parsing creates most of the time an offset-\( i \) red block (called \( i \)-violation in the sequel), which therefore would increase until it covers almost all the word \( w \). To avoid this, we introduce gadgets: we make sure that this happens at most half of the time (and thus cannot cover more than half of \( w \)). More precisely, Lemma 4.5 shows that at most half of the prefixes of \( x \) can contain offset-\( i \) blocks for any fixed \( i \leq \gamma k \). This is due to the insertion of gadgets that “kill” some starting positions \( i \) if necessary, by “resynchronizing” the parsing at a different position.

- On the other hand, red blocks starting at position \( i > \gamma k \) are shown to be of small size by Proposition 4.7. This is implied by Lemma 4.6 claiming that, due to the structure of the DB(\( k \)) (few repetitions of factors), few junction red blocks can go up to position \( (i-1) \) and precede an offset-\( i \) block.

Since all large enough prefixes of \( x \) have a constant portion containing only red blocks of size \( O(n^{1/4}) \), the compression speed is \( \Omega(n^{3/4}) \) (Theorem 4.1).

Gadgets must satisfy two conditions:

- they must not disturb the parsing of \( w \);
- the gadget \( g_i \) must “absorb” the end of the red block ending at position \( (i-1) \), and ensures that the parsing restarts at a controlled position different from \( i \).
The insertion of gadgets in $w$ is not trivial because we need to “kill” positions without creating too many other bad positions, that is why gadgets are only inserted in the second half of $w$. Moreover, gadget insertion depends on the parsing of $0w$ and must therefore be adaptive, which is the reason why we give an algorithm to describe the word $w$.

Let us summarize the organisation of the lemmas of this section:

- Lemma 4.3 is necessary for the algorithm: it shows that, in $0\text{Pref}(x)$, there can be at most one position $i$ such that the number of $i$-violations is too high.
- Lemma 4.4 shows that the parsing of $w$ is not disturbed by gadgets and therefore the compression speed of $w$ is $O(\sqrt{n})$.
- Lemma 4.5 shows that gadgets indeed remove $i$-violations as required, for $i \leq \gamma k$.
- Lemma 4.6 uses the property of the DB($k$) to prove that junction blocks cannot create too many $i$-violations if $i > \gamma k$.
- Finally, Proposition 4.7 uses Lemma 4.6 to show that the offset-$i$ red blocks are small if $i$ is large.

### 4.3 Construction and first properties

Let $\gamma$ be any constant greater than or equal to 3. Let $x$ be a DB($k$) beginning by 01. We denote its size by $s = 2^k + k - 1$. Suppose for convenience that $k$ is odd, so that $s$ is even. For $i \in [0, s - 1]$, let $w_i = x_{<i}$, so that $\text{Pref}(x) = w_0.w_1 \ldots w_{s-1}$.

The word $w$ that we will construct is best described by an algorithm. It will merely be $\text{Pref}(x)$ in which we possibly add “gadgets” (words) between some of the $w_j$ in order to control the parsings of $w$ and $0w$. The letter in front that will provoke the “catastrophe” is the first letter of $w$, that is, 0.

The gadgets $g_j^i$ (for $i \in [0, \gamma k]$ and $j \geq 0$) are defined as follows (where $\bar{x}_i$ denotes the complement of $x_i$):

- $g_0^i = 10^j$;
- and for $i > 0$, $g_i^j = x_{<i}.\bar{x}_i.1^j$.

Recall that the green blocks are those of the parsing of $w$, whereas the red ones are those of the parsing of $0w$. We call “regular” the green blocks that are not gadgets (they are of the form $w_j$ for some $j$). For $i \in [0, s - 1]$, we say that a regular green block in $w$ is $i$-violated if there is an offset-$i$ (red) block in it. Note that gadgets do not count in the definition of a violation.

---

5This is to avoid dealing with the fractional part of $s/2$, but the construction also works in the case where $k$ is even.
Lemma 4.3. For \( i \in [0, s - 1] \), let \( l_i \) be the number of \( i \)-violated blocks in \( \text{Pref}(x) = w_0w_1 \ldots w_{s-1} \). Then for all \( i \neq i' \), \( l_i + l_{i'} \leq s \).

In particular, there can be at most one \( i \) such that the number of \( i \)-violated blocks is \( > s/2 \).

Proof. Let \( i \) and \( i' \) be such that \( 0 \leq i < i' < s \).

Consider the red blocks starting at position \( i \) and \( i' \) in any green block.

No green block in \( w_0 \ldots w_{i'-1} \) is \( i' \)-violated since they are too small to contain position \( i' \). Let \( a \) be the number of \( i \)-violated blocks in \( w_0 \ldots w_{i'-1} \). In \( w_{i'} \ldots w_{s-1} \), let \( b \) be the number of green blocks that are both \( i \)-violated and \( i' \)-violated, and let \( c \) (respectively \( d \)) be the number of \( i \)-violated \( \text{(resp. } i' \text{-violated)} \) blocks that are not \( i' \)-violated \( \text{(resp. } i \text{-violated)} \) blocks.

The number of \( i \)-violations is \( l_i = a + b + c \) and the number of \( i' \)-violations is \( l_{i'} = b + d \). But \( b + c + d \leq s - i' \) and \( b \leq i' - i - a \) (since a red block starting at position \( i \) can only be increased \((i'-i)\) times before it overlaps position \( i' \), and it has already increased \( a \) times in the first \( i' \) green blocks), so that \( l_i + l_{i'} = (b + c + d) + (a + b) \leq (s - i') + (i' - i) \leq s \). Therefore, \( l_i \) or \( l_{i'} \) has to be \( \leq s/2 \).

\( \square \)

The algorithm constructing \( w \), illustrated in Figure 3, is as follows.

1. If the number of \( i \)-violations in \( w_0w_1 \ldots w_{s-1} \) is \( \leq s/2 \) for all \( i \in [0, \gamma k] \), then output \( w = w_0w_1 \ldots w_{s-1} \).

2. Otherwise, let \( i \) be the (unique by Lemma 4.3) integer in \([0, \gamma k]\) for which the number of \( i \)-violations is \( > s/2 \). Let \( c = 0 \) (counter for the number of inserted gadgets) and \( d = s/2 + 1 \) (counter for the place of the gadget to be inserted).

3. For all \( j \in [0, s - 1] \), let \( z_j = w_j \).

4. While the number of \( i \)-violations in \( z_0z_1 \ldots z_{s-1} \) is \( \geq d \), do:

   (a) let \( j \) be such that \( w_j \) is the \( d \)-th \( i \)-violated green block;

   (b) \( z_j \leftarrow g_i^c w_j \) (we add the gadget \( g_i^c \) before the block \( w_j \));

   (c) \( c \leftarrow c + 1 \);

   (d) if \( w_j \) is still \( i \)-violated, then \( d \leftarrow d + 1 \).

5. Return \( w = z_0z_1 \ldots z_{s-1} \).

Some parts of the algorithm might seem obscure, in particular the role of the counter \( d \). The proof of the following properties should help understand this construction, but let us first explain the intuition behind the algorithm. Below (Proposition 4.7) we will have a generic argument (i.e. true without gadgets) to deal with the \( i \)-violations for \( i > \gamma k \), therefore for now we only care of \( i \)-violations for \( i \leq \gamma k \). They are not problematic if there are at most \((\text{roughly}) \) \( s/2 \) of them. Thanks to Lemma 4.3 there is therefore at most one \( i_0 \) which can be problematic. To guarantee the upper bound of \((\text{roughly}) \) \( s/2 \) for the number
Figure 3: Illustration of Step 4(b) of the algorithm.

Figure 4: Left: Form of the word $w$. The blocks in green are the regular blocks, the blocks in blue are the gadgets. The arcs represent the relation of paternity. Right: The shape of the parsing tree of $w$.

of $i_0$-violations, every time it is necessary we insert between two regular green blocks one gadget to kill the $(s/2 + 1)$-th, $(s/2 + 2)$-th, etc., $i_0$-violations. But gadgets are guaranteed to work as expected only if at least $1 + (\gamma + 1)k$ of them have already been inserted (see Lemma 4.5), hence the counter $d$ is useful to avoid inserting two gadgets in front of the same regular block.

From now on, we call $w$ the word output by the algorithm. We first evaluate the size of $w$. Its minimal size is obtained when no gadgets are added during the algorithm:

$$|w| \geq \frac{s(s + 1)}{2}.$$

On the other hand, if $s/2$ gadgets $g_{\gamma k}$ of size $\gamma k + 1 + c$ are added, we obtain an upper bound on $|w|$:

$$|w| \leq \frac{s(s + 1)}{2} + \sum_{c=0}^{s/2-1} (\gamma k + 1 + c) = \frac{5s^2}{8} + o(s^2).$$

Let us show that the word $w$ is nearly optimally compressible (upper bound).
Lemma 4.4. The compression speed of $w$ is at most

$$3\sqrt{\frac{2}{5}}\sqrt{|w|}.$$ 

Proof. If the algorithm stops at step 1, then $w = \text{Pref}(x)$ and it is compressed optimally (see Remark 2.1): the compression speed is

$$\sqrt{2}\sqrt{|w|} + O(1).$$

Otherwise, we add at most one gadget for each $w_j$, and only for $j > s/2$. Therefore, there are at most $s/2$ gadgets. Remark that, for the $i$ fixed in the algorithm, the gadgets $(g^i_j)_j$ are prefixes one of each other, and none of them are prefixes of $x$. Thus the parsing tree of $w$ consists of one main path of size $s$ (corresponding to $w_0, w_1, \ldots, w_{s-1}$), together with another path of size $\leq s/2$ (corresponding to the gadgets $(g^i_j)_j$) starting from a vertex of the main path. See Figure 4.

The worst case for the compression speed is when the second path is of size $s/2$ and starts at the root. Then the size of $w$ is

$$|w| \geq \frac{s(s + 1)}{2} + \frac{1}{2}(1 + \frac{s}{2}) \geq \frac{5s^2}{8}$$

and the size of the dictionary is $3s/2$, yielding the compression speed stated in the lemma.

Let us now turn to the lower bound on the compression speed of $0w$. The next lemma shows that, for $i \leq \gamma k$, there are not too many $i$-violations thanks to the gadgets.

Lemma 4.5. For all $i \in [0, \gamma k]$, the number of $i$-violations in $w$ is at most

$$s/2 + (1 + \gamma)k + 1.$$ 

Proof. If no gadgets have been added during the algorithm, then for all $i \in [0, \gamma k]$, the number of $i$-violations in $w$ is $\leq s/2$.

Otherwise, first remark that Lemma 4.3 remains valid even when the gadgets are added. We need to distinguish on the type ($i = 0$ or $i > 0$) of the most frequent violations in $w$.

• Case 1: the most frequent violations are 0-violations. In that case, we claim that whenever a gadget is inserted before a block $w_i$, the 0-violation in $w_i$ disappears. It is enough to prove that whenever a gadget $g^i_0$ is added, it was already in the dictionary of $0w$, so that the next word in the dictionary will begin with $g^i_0$ and the parsing will overlap position 0 of the next green block.

We proceed by induction: for $j = 0$, $g^0_0 = 1$, and this word is the third block in the parsing of $0w$, because $x$ starts with 01. For $j > 0$: when
\(g_0^{-1}\) was parsed, by induction it was already in the dictionary, so that the block added in the dictionary of \(0w\) starts with \(g_0^{-1}0 = g_0\).

After at most \(s/2\) iterations of the while loop, there is no more \((s/2 + 1)\)-th \(0\)-violation: the number of \(0\)-violations is exactly \(s/2\). Observe that violations for \(i > 0\) have been created, but by Lemma 4.3 for each \(i > 0\), the number of \(i\)-violations remains \(\leq s/2\).

\[\bullet\] Case 2: the most frequent violations are \(i\)-violations for some \(i > 0\). In that case, the first few times when a gadget is inserted, it may fail to kill the corresponding \(i\)-violation. But we claim that the number of such fails cannot be larger than \((\gamma + 1)k + 1\) (equivalently, in the algorithm the counter \(d\) remains \(\leq s/2 + (\gamma + 1)k + 2\)).

Indeed, since we add a gadget only before an \(i\)-violation, the parsing splits the gadget \(g_0^i = x_{<i}; \bar{x}_i; 1^j\) between \(x_{<i}\) and \(\bar{x}_i; 1^j\). Furthermore, by induction, \(\bar{x}_i; 1^j\) is not split by the parsing. But for the gadget \(g_i^{k+1}\), \(\bar{x}_i; 1^{k+1}\) is parsed in exactly one block because this factor does not appear anywhere in \(w\) before \(g_i^{k+1}\). From that moment on, each \(i\)-violation creates through the gadget a \(0\)-violation. The number of blocks that are both \(0\)-violated and \(i\)-violated is at most \(i\) (due to the growth of the block at position 0). Thus, at most \(i\) more gadgets may fail to kill position \(i\). The total number of “failing” gadgets is \(\leq k + 1 + i \leq (\gamma + 1)k + 1\).

\[\square\]

4.4 The weak catastrophe

This section is devoted to the proof of the lower bound: the compression speed of \(0w\) is \(\Omega(|w|^{3/4})\). Thanks to Property (⋆), Lemma 4.6 below bounds the number of junction blocks ending at a fixed position \((i - 1)\) by a decreasing function of \(i\). The proof is quite technical and requires to distinguish three categories among (red) junction blocks:

\[\bullet\] Type 1: junctions over consecutive factors \(w_a\) and \(w_{a+1}\) (no gadget between two regular green blocks);

\[\bullet\] Type 2: junctions starting in a gadget \(g_j^{k'}\) and ending in the following regular green block;

\[\bullet\] Type 3: junctions starting in a regular green block and ending in the following gadget \(g_j^{k'}\).

**Lemma 4.6.** Let \(i \geq 2k + 3\). Let \(uu'\) be a junction block of type 1 over \(w_{a}w_{a+1}\) ending at position \(i - 1\) in \(w_{a+1}\), with \(u\) being the suffix of \(w_a\) and \(u'\) the prefix of \(w_{a+1}\). Then \(|u| \leq k - \log(i - 2k - 1)\).

In particular, the number of such blocks is upper bounded by the number of words of size \(\leq k - \log(i - 2k - 1)\), that is, \(\frac{2^{|u|+1}}{2k} \cdot \).
Proof. Let \( v \) be the prefix of size \( 2k \) of \( u' \) (which is also the prefix of \( x \)). All the prefixes of \( uu' \) of size \( \geq |uv| \) have to be in the dictionary of \( 0w \): we call \( M \) the set of these prefixes (\( |M| = i - 2k \)). We claim that these blocks are junction blocks of type 1 or 3 only (except possibly for one of type 2), with only \( u \) on the left side of the junction. Indeed, let us review all the possibilities:

1. \( uv \) cannot be completely included in a regular block, otherwise \( v[0..k-1] \) would appear both at positions 0 and \( p > 0 \) in \( x \), which contradicts Property (⋆);

2. \( uv \) cannot be completely included in a gadget:
   - if the gadget is \( g_b^0 = 10^j \), impossible because \( v \) cannot have more than \( k \) zeroes since it is a factor of \( x \),
   - if the gadget is \( g_b^j = x_{<b}.\bar{x}_b.1^j \), by the red parsing of gadgets, either \( uv \) is in \( x_{<b} \) (impossible because \( v \) would appear at a position \( \geq |u| \) in \( x \)), or \( uv \) is in \( \bar{x}_b.1^j \) (impossible because \( v \) cannot contain more than \( k \) ones);

3. if \( uv \) is a type 1 junction but not split between \( u \) and \( v \), it is impossible because the three possible cases lead to a contradiction:
   - if \( u \) goes on the right, then \( v \) would appear at another position \( p > 0 \) in \( x \),
   - if \( v \) goes on the left by at least \( k \), then \( v[0..k-1] \) would again appear at two different positions in \( x \),
   - if \( v \) goes on the left by less than \( k \), then it goes on the right by more than \( k \) and \( v[k..2k-1] \) would again appear at two different positions in \( x \);

4. if \( uv \) is a type 2 junction but not split between \( u \) and \( v \), it is again impossible:
   - if \( u \) goes on the right, then \( v \) would appear at another position \( p > 0 \) in \( x \),
   - if \( v \) goes on the left by at least 2, then \( v[0..1] \) would be either 00 or 11 (depending on the gadget), but we know it is \( x_0x_1 = 01 \),
   - otherwise, \( v \) goes on the right by \( 2k - 1 \), and \( v[1..k] \) would appear at positions 0 and 1 in \( x \);

5. if \( uv \) is a type 3 junction, first remark that the gadget is of the form \( g_b^j \) for \( b > 0 \) (because, for gadgets of the form \( g_b^0 \), the red parsing starts at position 0 of the gadget). If \( uv \) is not split between \( u \) and \( v \), it is once again impossible:
   - if \( u \) goes on the right, the red parsing of the gadget stops after \( x_{<b} \) and \( v \) would appear in \( x \) at a non-zero position,
• similarly, if $v$ goes on the left by less than $k$, then $v[k..2k-1]$ would again appear at two different positions in $x$,

• if $v$ goes on the left by at least $k$, then $v[0..k-1]$ would again appear at two different positions in $x$.

Remark finally that all parsings of type 2 junctions have different sizes on the left. Therefore, at most one can contain $u$ on the left. The claim is proved.

Thus, at least $|M| - 1$ regular green blocks have $u$ as suffix. Remark that, since $|M| \geq 3$, there are at least two such green blocks, therefore $|u| \leq k$. Hence by Property $(\ast)$ we have:

$$|M| - 1 \leq 2^{k-|u|}$$

$$i - 2k - 1 \leq 2^{k-|u|}$$

$$|u| \leq k - \log(i - 2k - 1).$$

As a consequence, in the next proposition we can bound the size of offset-$i$ blocks. Along with the role of gadgets, this will be a key argument for the proof of Theorem 4.1. The idea is the following: for a red block $u$ starting at a sufficiently large position $i$, roughly $|u|$ other red blocks have to end at position $(i - 1)$, and in the red parsing $\Omega(|u|^2)$ prefixes of these blocks must appear in different green blocks (and in the dictionary), giving the bound $s = \Omega(|u|^2)$.

**Proposition 4.7.** For any $i > \gamma k$, the size of an offset-$i$ block included in a regular green block is at most

$$2\sqrt{s} + 5k + \frac{2^{k+1}}{i - 2k - 1}.$$

**Proof.** Let $u$ be an offset-$i$ block of size $\geq 2k$.

We claim that the red blocks predecessors of $u$ of size at least $2k + 1$ have to start at position $i$ in regular green blocks. Indeed, let $v$ be a prefix of size $\geq 2k + 1$ of $u$; let us analyse as before the different cases:

• If $v$ is included in a regular green block, then it has to start at position $i$ by Property $(\ast)$;

• $v$ cannot be included in a gadget since it would lead to a contradiction:
  
  – in gadgets of type $g_6^0$, $v$ would contain $0^{2k}$,
  
  – in gadgets of type $g_6^1 = x < a \bar{x}_a 1^j$ (for $a \in [0, \gamma k]$), either $v$ goes into $x < a$ by at least $k$ and $v[0..k-1]$ would appear at two positions in $x$, or $v$ goes into $\bar{x}_a 1^j$ by at least $k + 2$ and $v$ would contain $1^{k+1}$;

• If $v$ is included in a junction block of type 1, then $v$ starts at position $i$ in the left regular block, otherwise either $v[0..k-1]$ would be in the left regular block at a position different from $i$, or $v[k+1..2k]$ would be in the right regular block at a position $\leq \gamma k < i$;
• $v$ cannot be included in a junction block of type 2: indeed, by the red parsing, the left part of the junction (included in a gadget) is either $10^j$ or $a1^j$ for some letter $a \in \{0, 1\}$, thus $v$ cannot go on the left by $\geq k + 2$ and hence has to go on the right by at least $k$ leading to a contradiction with Property $(\ast)$;

• If $v$ is included in a junction block of type 3, then $v$ starts at position $i$ in the left regular block, otherwise either $v[0..k-1]$ would be in the left (regular) block at a position different from $i$, or, by the red parsing, the gadget is of type $g_{j_a}$ (for $a > 0$) and $v[k+1..2k]$ would be included in $x < a$ at a position $\leq \gamma k < i$.

Thus, at least $|u| - 2k$ red blocks end at position $i - 1$.

By Lemma 1.6 at most $\frac{2^{k+1}}{i - 2k - 1}$ of them are junctions of type 1. Note furthermore that, as shown during the proof of Lemma 1.6, if $a \geq k + 1$, the $a$-th junction of type 2 stops at position $0$ or $1$, hence at most $k$ of the blocks ending at position $i - 1$ are junctions of type 2. Finally, there is, by definition, no junction of type 3. Therefore, there are at least $|u| - 3k - \frac{2^{k+1}}{i - 2k - 1}$ offset blocks ending at position $i - 1$. We call $M$ the set of such blocks. See Figure 5. Remark that $|u| - 3k - \frac{2^{k+1}}{i - 2k - 1}$ is a lower bound on the number of offset blocks ending at position $i - 1$. The number of such blocks is at most $i$. Therefore

$$|u| - 3k - \frac{2^{k+1}}{i - 2k - 1} \leq i$$

We distinguish two cases in the proof:

First case: $i \in [\gamma k + 1, 2\sqrt{s}]$. Then

$$|u| - 3k - \frac{2^{k+1}}{i - 2k - 1} \leq 2\sqrt{s}$$

so that

$$|u| \leq 2\sqrt{s} + 3k + \frac{2^{k+1}}{i - 2k - 1} \leq 2\sqrt{s} + 5k + \frac{2^{k+1}}{i - 2k - 1}.$$

Second case: $i > 2\sqrt{s}$.

All the words in $M$ are in the dictionary and are of different size, since two offset blocks ending at the same position and of same size would be identical, which is not possible in the LZ-parsing. The words of $\mathcal{P}(M)$ (the set of prefixes of the words in $M$) are also in the dictionary. Let

$$A = \frac{|u| - 5k - \frac{2^{k+1}}{i - 2k - 1}}{2}.$$  

Observe that $i - A - 2k \geq \frac{i}{2} - k$ as $|u| - 3k - \frac{2^{k+1}}{i - 2k - 1} \leq i$. Therefore $i - A - 2k \geq \sqrt{s} - k$, which is large against $\gamma k$. Consider the words of $\mathcal{P}(M)$ containing $x[i - A - 2k..i - A]$: they must start at a position $\leq i - A - 2k$ and end at a position $\in [i - A, i - 1]$. The number of such words is at least the
product of the number of blocks in \(M\) starting at position \(\leq i - A - 2k\) and of the number of possible ending points, that is, at least

\[
\left(\left(\left|u\right| - 3k - \frac{2^{k+1}}{i - 2k - 1}\right) - (A + 2k)\right)A.
\]

Remark that these words contain a part of a regular green block of size at least \(2k + 1\) starting at position \(i - A - 2k > \gamma k\). Hence, by the same case analysis as before, for these words, the part corresponding to the factor \(x[i - A - 2k..i - A - k - 1]\) must appear included in a regular green block, so that two such words cannot appear in the same regular green block by Property \((\ast)\). But there are at most \(s\) distinct regular green blocks, thus:

\[
\left(\left|u\right| - 5k - \frac{2^{k+1}}{i - 2k - 1} - A\right)A \leq s.
\]

The value of \(A\) gives:

\[
\left(\frac{\left|u\right| - 5k - \frac{2^{k+1}}{i - 2k - 1}}{2}\right)^2 \leq s
\]

\[
\left|u\right| \leq 2\sqrt{s} + 5k + \frac{2^{k+1}}{i - 2k - 1}.
\]

\[\square\]

Figure 5: Blocks ending at position \(i - 1\) for the proof of Proposition \ref{prop:main}.

We are ready for the proof of the main theorem of this section.

**Proof of Theorem \ref{thm:main}**. The intuition is the following: by Proposition \ref{prop:main}, the red blocks starting at position \(j\), for \(j = \Omega(\sqrt{s})\), are of size \(\Theta(\sqrt{s}) = \Theta(|w|^{1/4})\), so if we prove that a portion of size \(\Theta(|w|)\) of the word \(0w\) is covered by offset-\(j\)
blocks for \( j \) large enough, then the compression speed will be \( \Omega(|w|^{3/4}) \). To that purpose, we prove that for large enough regular green blocks, there is an interval of positions \([2s/3 - l, 2s/3]\) (with \( l = 2\sqrt{s} + 5k + 3 \)), such that there is at least one offset-\( i \) block for \( i \in [2s/3 - l, 2s/3] \).

In every regular green block of size larger than \( 2s/3 \), let us show that there is an offset-\( i \) red block, for \( i \in [2s/3 - l, 2s/3] \). Indeed, for every \( i < 2s/3 - l \), the maximal size \( f(i) \) of a red block starting at position \( i \) satisfies \( i + f(i) \leq 2s/3 \): in the case where \( i > \gamma k \) we use the bound given by Proposition 4.7, and in the case where \( i \leq \gamma k \), the \((t - \gamma k)\) predecessors of size \( \geq \gamma k + 1 \) of a red block of size \( t \) starting at position \( i \) start at position \( i \) as well (since \( x \leq \gamma k \) is not a factor of a gadget, it cannot be seen anywhere by a red block except at position \( i \) in a regular green block), hence the size of an offset-\( i \) block in that case is at most \( \gamma k \) plus the number of \( i \)-violations. Therefore by Lemma 4.6 red blocks starting at position \( i \) have their size upper bounded by \( \gamma k + s/2 + 1 + (1 + \gamma)k \).

Therefore, since a red block starting at position \( i \geq 2s/3 - l \) is of size at most \( B = 2\sqrt{s} + 5k + \frac{2^{k+1} + 1}{2s/3 - l - 2k - 1} \) by Proposition 4.7, each green block of size \( h \geq 2s/3 \) is covered by at least

\[
\frac{h - 2s/3}{B} \geq \frac{h - 2s/3}{2\sqrt{s} + O(k)}
\]

red blocks. Thus, the total number of red blocks is at least

\[
\frac{1}{2\sqrt{s} + O(k)} \sum_{h=2s/3}^{s} (h - 2s/3) = \frac{1}{36} s^{3/2} + o(s^{3/2}).
\]

With the gadgets, the size of \( w \) is at most \((5/8)s^2 + o(s^2)\), therefore the total number of red blocks is at least:

\[
\frac{1}{36} \left( \frac{8}{5} \right)^{3/4} |w|^{3/4} - o(|w|^{3/4}) \geq 0.039 |w|^{3/4}.
\]

\[\square\]

**Remark 4.8.** Despite the fact that 1Pref\((x)\) compresses optimally, this is not at all the case with the gadgets, since Theorem 4.1 remains valid with the new word \( w \) output by the algorithm even when we put 1 instead of 0 in front of \( w \).

## 5 General case

In this section we prove Theorem 2.10. The proof first goes through the existence of a family \( F \) of “independent” de Bruijn-style words which will play a role similar to the de Bruijn word \( x \) in the proof of Theorem 4.1. The existence of this family is shown using the probabilistic method in Section 5.1 with high probability, a family of random words satisfies a relaxed version (P1) of the “local” Property (+), together with a global property (P2) that forbids repetitions of large factors throughout the whole family.
The word $w$ that we will consider is the concatenation of “chains” roughly equal to $\text{Pref}(x)$ for all words $x \in F$, with gadgets inserted if necessary as in Section 3. (The construction is actually slightly more complicated because in each chain we must avoid the first few prefixes of $x$ in order to synchronise the parsing of $w$; and the gadgets are also more complex.) Properties (P1) and (P2) guarantee that each of the chains of $w$ are “independent”, so that the same kind of argument as in Section 4 will apply individually. By choosing appropriately the number of chains and their length, we can obtain any compression speed for $w$ up to $\Theta(n/\log^2 n)$ and the matching bound for $0w$ (see Theorem 2.10).

The organisation of the section is as follows: Section 5.1 is devoted to the proof of existence of the required family of words. Section 5.2 defines the gadgets, describes the construction of $w$ thanks to an algorithm, and gives the upper bound on the compression speed of $w$. Finally, Section 5.3 shows the lower bound on the compression speed of $0w$ thanks to a series of results in the spirit of Section 4.4.

Throughout the present section, we use parameters with some relations between them that are worth being stated once and for all in Figure 6 for reference.

| Parameter | Description |
|-----------|-------------|
| $n$       | sufficiently large (the size of $w$) |
| $\gamma$  | $\geq 10$ (an absolute constant) |
| $l$       | $\in [(9\gamma)^2 \log^2 n, \sqrt{n}]$ (size of the $x^j$) |
| $p$       | $\log \frac{l}{n}$ (2$^p$ is the number of chains) |
| $k$       | $= \log \frac{l}{p}$ (parameter in (P1)) |
| $m$       | $= \max(\gamma p, \gamma \log l)$ (parameter in (P2)). |

Figure 6: Parameters used throughout Section 5

In particular, note that we have the following relations:

$$0 \leq p \leq \sqrt{l} \quad \text{and} \quad \frac{\gamma \log n}{3} \leq m \leq \frac{\sqrt{l}}{9}.$$

5.1 Family of de Bruijn-type words

We need two properties for a family $F$ of $2^p$ words $x^1, \ldots, x^{2^p}$ of size $l$ (the parameters $n, l, p, k, \gamma$ and $m$ are those given in Figure 6): the first is a relaxed version of Property (⋆) on “true” de Bruijn words; the second guarantees that the words of $F$ are “independent”.

- (P1) For all $x \in F$, for all words $u$ of size $\leq k$,

$$\text{Occ}_x(u) \leq \frac{kl}{2|u|}.$$
• (P2) Any factor $u$ of size $m$ appears in at most one word of the family $F$, and within that word at only one position.

Note that in Section 4, we did not need (P2) since only one word was concerned, but still (P2) was true for the same value $k$ as in (⋆), instead of $m$ here.

The following lemmas show that (P1) and (P2) hold with high probability for a random family $F$. We first recall the well-known Chernoff bound.

**Theorem 5.1 (Chernoff bound).** Let $X_1, \ldots, X_n$ be independent random variables over $\{0, 1\}$, and $X = \sum X_i$. Denote by $\mu$ the expectation of $X$. Let $\delta > 1$.

Then:

$$\Pr(X > \delta \mu) < 2^{-\frac{(\delta - 1)\mu \log \delta}{2}}.$$ 

For (P1), we need to consider positions separated by a distance $k$ in order to obtain the independence required for the Chernoff bound; then a union bound will complete the argument for the other positions.

**Lemma 5.2 ((P1) holds whp).** Let $p$ and $l$ be positive integers such that $p \leq \sqrt{l}$. Let $F$ be a family of $2^p$ words $x^1, \ldots, x^{2^p}$ of size $l$ chosen uniformly and independently at random. Then $F$ satisfies Property (P1) with probability $2^{-\Omega(\sqrt{l} \log \log l)}$.

**Proof.** Fix $x \in F$ and a word $u$ of size $\leq k$. For $i \in [0, k-1]$ and $j \in [0, l/k-1]$, let

$$X^i_j = \begin{cases} 
1 & \text{if } u \text{ occurs at position } i + jk \text{ in } x \\
0 & \text{otherwise}.
\end{cases}$$

For a fixed $i$, the $X^i_j$ are independent. Let $\mu_i = E(\sum_j X^i_j)$. We have:

$$\mu_i = \frac{l}{k2^{|u|}}.$$ 

By the Chernoff bound (Theorem 5.1):

$$\Pr\left(\sum_j X^i_j > k\mu_i\right) < 2^{-\frac{(k-1)(\log k)\mu_i}{2}},$$

that is,

$$\log \Pr\left(\sum_j X^i_j > \frac{l}{2^{|u|}}\right) < -(k - 1)(\log k) \frac{l}{K2^{k+1}}.$$ 

By union bound over all the words $u$ of size at most $k$, all the words of $F$ and all the moduli $i \in [0, k-1]$, we have:

$$\log \Pr\left(\sum_{i,j} X^i_j > \frac{kl}{2^{|u|}}\right) < (k + 1) + p + \log k - (k - 1)(\log k) \frac{l}{K2^{k+1}} = -\Omega(\sqrt{l} \log l).$$ 

$\Box$
The analysis for (P2) does not use Chernoff bounds, but instead it uses a slight "independence" on the occurrences of a factor $u$ obtained by showing that $u$ can be supposed "self-avoiding" (the precise meaning of these ideas will be clear in the proof).

**Lemma 5.3 ((P2) holds whp).** Let $p$ and $l$ be positive. Recall that $m = \max(\gamma p, \gamma \log l)$ in (P2). Let $F$ be a family of $2^p$ words $x^1, \ldots, x^{2^p}$ of size $l$ chosen uniformly and independently at random. Then $F$ satisfies Property (P2) with probability at least $1 - \frac{1}{2l}$.

**Proof.** Let us first show that we can assume with high probability that factors of $w_i$ are not overlapping too much. We say that a word $u$ of size $m$ is "bad" if it overlaps itself at least by half, that is:

$$\exists i \in [\lfloor |u|/2 \rfloor, |u| - 1] : u[0..i - 1] = u[|u| - i..|u| - 1]$$

(we say that $u$ is $i$-bad).

(Remark that a word $u$ can be both $i$-bad and $j$-bad for $i \neq j$.) Let us first bound the number of bad words. If $u$ is $i$-bad, then for each $j < i$, $u_{j+|u|-i} = u_j$. Therefore, specifying the $|u| - i$ first bits specifies the whole word $u$, meaning that there are at most $2^{m-i}$ i-bad words. In total, there are at most

$$\sum_{i = \lfloor |u|/2 \rfloor}^{\lfloor |u|-1 \rfloor} 2^{m-i} = 2^{1+|u|/2} - 2$$

bad words, that is, a fraction $< 2^{-m/2+1}$ of all words of size $m$.

Now, we say that a word $x^j$ of size $l$ is "good" if it contains no bad factor. Let us show that, with high probability, all the words $x^j \in F$ are good (Property (G)). Fix $j \in [1, 2^p]$. If $x^j$ is not good, then there is at least one position where a bad factor $u$ occurs:

$$\Pr(x^j \text{ is not good}) \leq |x^j| \Pr_{|u|=m}(u \text{ is bad}) \leq l2^{-m/2+1}.$$

We use the union bound over all $2^p$ words $x^j \in F$ to obtain:

$$\Pr(G) \geq 1 - l2^{-m/2+1}.$$

Since property $G$ has very high probability, we will only show that (P2) holds with high probability when $G$ is satisfied. Let $x = x^1 \ldots x^{2^p}$ (the size of $x$ is therefore $l2^p$). Let $u$ be a word of size $m$, which is not bad. Let $X_u$ be the number of occurrences of $u$ in $x$. In order to get at least two occurrences of $u$, we have to choose two positions, the $|u|$ bits of the first occurrence, and the bits of the second occurrence that are not contained in the first; but $u$ can’t overlap itself by more than $m/2$ bits, thus:

$$\Pr(X_u \geq 2) \leq \frac{|u|}{2^m} \frac{|u|}{2^{m/2}} \leq l^2 2^p - 2^m.$$

Using the union bound over all good words $u$ of size $m$, of which there are at most $2^m$, we get:

$$\Pr(\forall \text{ good } u, X_u \leq 1) \geq 1 - 2^m l^2 2^p - 2^m = 1 - l^2 2^p - m/2.$$
Now, the probability that $F$ respects Property (P2) can be lower bounded by the probability that $F$ contains no bad words, and that the number of occurrences of good words is at most 1, which gives:

$$\Pr(F \text{ satisfies (P2)}) \geq \Pr(G \land \forall \text{ good } u, X_u \leq 1)$$

$$\geq 1 - l^{2^{2p-m/2} - l^{2p-m/2+1}}$$

$$> 1 - \frac{2}{l} \quad \text{since } \gamma \geq 10$$

(for the last line, consider the two following cases: $p \geq \log l$ where $m = \gamma p$ and $l \leq n^{1/3}$; and $p \leq \log l$ where $m = \gamma \log l$ and $l \geq n^{1/3}$).

**Corollary 5.4.** For all sufficiently large $l$ and $p \leq \sqrt{l}$, there exists a family $F$ of $2^p$ words $x_1, \ldots, x_{2^p}$ of size $l$ satisfying Properties (P1) and (P2), and where the first bit of $x^1$ is 1.

### 5.2 Construction

(Recall the choice of parameters $n, l, p, k, \gamma$ and $m$ defined in Figure 6)

For $n$ sufficiently large and $l \in [(9\gamma)^2 \log^2 n, \sqrt{n}]$, we will construct a word $w$ of size $n$ whose compression speed is $\Theta(n/l)$ whereas the compression speed of $0w$ is $\Theta(n/\sqrt{l})$ (thus matching the upper bound of Theorem 2.7). Let $F$ be a family as in Corollary 5.4. For some integers $q_j$ (defined below), the word $w$ will merely be the concatenation of $\text{Pref}_{>q_j}(x^j)$ (see Remark 2.1 for the definition of $\text{Pref}_{>q}(x)$) for all the $2^p$ words $x^j$ of the family $F$, with possibly some gadgets added between the prefixes of $x^j$ (each $\text{Pref}_{>q_j}(x^j)$ together with the possible gadgets will be denoted $z^j$ and called a “chain”), and a trailing set of zeroes so as to “pad” the length to exactly $n$. The integer $q_j$ will be chosen so that the first occurrence of $x^j[0..q_j]$ is parsed in exactly one green block.

Each chain $z^j$ (with gadgets) is of size $\Theta(l^2)$ and is fully compressible in $w$ (compressed size $\Theta(l)$) since it is made of prefixes (plus gadgets that won’t impede much the compression ratio). Thus the total compression size of $w$ is $\Theta(l^{2p})$, compared to $|w| = n = \Theta(l^{2p})$ for a compression speed of $\Theta(n/l)$.

On the other hand, due to the properties of $F$ and similarly to Theorem 4.1 in $0w$ each chain will compress only to a size $\Theta(l^{3/2})$, thus the total compression size of $0w$ is $\Theta(l^{3/2}2^p)$, for a compression speed of $\Theta(n/\sqrt{l})$.

**Remark 5.5.**

- If we take the smallest possible $l$, that is, $l = (9\gamma)^2 \log^2 n$, then we obtain compression speeds of $\Theta(n/\log^2 n)$ and $\Theta(n/\log n)$, thus showing the one-bit catastrophe.

- On the other hand, if we take the largest possible $l$, that is, $l = \sqrt{n}$, then we obtain $\Theta(\sqrt{n})$ and $\Theta(n^{3/4})$ as in Theorem 4.1

Let us now start the formal description of the word $w$. As previously, we will call green the blocks in the parsing of $w$ and red those in the parsing of $0w$. The green blocks in each chain $z^j$ that are not gadgets will be called “regular
blocks in (they are of the form $x^j[0..q]$ for some $q$). Recall that the chain $z^j$ will be of the form $\text{Pref}_{\triangleright q}(x^j)$ with possibly some gadgets between the prefixes.

We can already define the integers $q_j$:

$$q_j = \min\{i \geq 0 : x^j[0..i] \text{ is not a prefix of } x^1, \ldots, x^{j-1}\}.$$ 

In that way, we guarantee that the first green block in each $z^j$ is exactly $x^j[0..q_j]$. Remark that, by Property (P2), $q_j \in [0, m]$. For all $j$ we will denote by $s_j = |x^j| - q_j = l - q_j$ the number of regular green blocks in $z^j$.

Fix $n$ and $l = l(n) \in [(9\gamma)^2 \log^2 n, \sqrt{n}]$, and let $k = (\log l)/2$ and $p = \log(n/l^2)$. As in Property (P2), call $m = \max(\gamma p, \gamma \log l)$. Here are the new gadgets that will (possibly) be inserted in the chain $z^j$ ($j \in [1, 2^p]$), for $i \in [0, 2k\sqrt{l}]$:

- for $c \geq 0$: $g_0^c(j) = ua^c$, where $a = x^j[0]$ is the first letter of $x^j$, and $u$ is the smallest word in

$$\text{Dic}(0 z^1 \ldots z^{j-1} \triangleright \text{Pref}_{\triangleright q_j}(x^j[0..|x^j|/2]))$$

but not in

$$\text{Dic}(z^1 \ldots z^{j-1} \triangleright \text{Pref}_{\triangleright q_j}(x^j)) :$$

this is a word which is in the parsing of $0w$ up to the insertion of $g_0^c(j)$ but not in the corresponding parsing of $w$ (Lemma 5.7 below guarantees the existence of such a word and proves it is of size $\leq m$);

- for $i > 0$ and $c \geq 0$, let $m' = \max(i, m)$ and $v = x^j[0..m - 1]1^i$. Then:

$$g_i^c(j) = x^j[0..m' - 1]\overline{x_m^j}v[0..c - 1]$$

where $\overline{x_m^j}$ denotes the complement of $x^j[m']$.

We define $i$-violations in each chain $z^j$ as previously, that is, a regular green block is $i$-violated if it contains an offset-$i$ red block. The following lemma is proved in the exact same way as Lemma 4.3

**Lemma 5.6.** For $j \in [1, 2^p]$ and $i \in [0, s_j - 1]$, let $l_i^j$ be the number of $i$-violated blocks in $z^j$. Then for all $j$ and all $i \neq i'$, $l_i^j + l_{i'}^j \leq s_j$.

In particular, for each $z^j$ there can be at most one $i$ such that the number of $i$-violated blocks is $> s_j/2$.

The formal construction of the word $w$ is once again best described by an algorithm taking as parameters $n$ and $l$:

1. For all $j \in [1, 2^p]$ and $i \in [q_j + 1, l]$, $z_i^j \leftarrow x^j[q_j..i - 1]$. Throughout the algorithm, $z^j$ will denote $z_{q_j+1}^j \ldots z_{l}^j$ (and thus will vary if one of the $z_i^j$ varies).

2. For $j = 1$ to $2^p$ do:
(a) if there is \( i \in [0, 2k\sqrt{l}] \) (unique by Lemma 5.6) such that the number of \( i \)-violations in the chain \( z^j \) is \( > s_j/2 \), then:

i. let \( c = 0 \) (counter for the number of inserted gadgets in \( z^j \)) and

\( d = s_j/2 + 1 \) (counter for the place of the gadget to be inserted),

ii. while the number of \( i \)-violations in the chain \( z^j \) is \( \geq d \), do:

A. let \( r \) be such that \( z^j_r \) is the \( d \)-th \( i \)-violated green block in \( z^j \),

B. \( z^j_r \leftarrow gc_i(j)z^j_r \) (we add the gadget \( gc_i(j) \) before the block \( x^j[0..r-1] \)),

C. \( c \leftarrow c + 1 \),

D. if \( z^j_r \) is still \( i \)-violated, then \( d \leftarrow d + 1 \).

3. Let \( w' = z^1z^2\ldots z^{2p} \). Return \( w = w'0^{n-|w'|} \) (padding to obtain \( |w| = n \)).

Remark that we have the following bounds on the size of \( w' \). Its size is minimal if no gadgets are added:

\[
|w'| \geq 2^p \left( \sum_{i=m}^{l} i \right) = \frac{n}{l^2} \cdot \frac{(l-m+1)(l+m)}{2} \geq \frac{n}{2} - o(n)
\]

and its size is maximal if each chain contains \( l/2 \) gadgets \( gc_i \) (whose size is at most \( 2k\sqrt{l} + 1 + c \)):

\[
|w'| \leq 2^p \left( \sum_{i=1}^{l} i + \sum_{c=0}^{l/2-1} (2k\sqrt{l} + 1 + c) \right)
\leq \frac{n}{l^2} \left( \frac{5l^2}{8} + 2kl^{3/2} \right) \leq n.
\]

Therefore at the end of the algorithm it is legitimate to pad \( w' \) with at most \( n/2 + o(n) \) zeroes to obtain the word \( w \) of size precisely \( n \).

The following lemma justifies the existence of the gadgets \( gc_0(j) \).

**Lemma 5.7.** There is a constant \( C > 0 \) such that, for all \( n \), for all \( l \in [(9\gamma)^2\log^2 n, \sqrt{n}] \) with \( l > C \), for all \( j \in [1, 2^p] \) (where \( p = \log(n/l^2) \)) there exists a word \( u \) of size \( \leq m \) in

\[
\text{Dic}(0z^1\ldots z^{j-1}\text{Pref}_{>q_j}(x^j[0..|x^j|/2]))
\]

but not in \( \text{Dic}(z^1\ldots z^{j-1}\text{Pref}_{>q_j}(x^j)) \).

This implies that we can insert the gadgets \( gc_0(j) \) in a chain \( z^j \) whenever we need to.

**Proof.** For \( j = 1 \): the first red block in \( z^1 \) is 0, but all the regular green blocks in \( z^1 \) begin with 1 (cf. Corollary 5.4). Therefore there exists a word \( u \) of size 1 in

\[
\text{Dic}(0\text{Pref}_{>q_1}(x^1[0..|x^1|/2]))
\]

not in \( \text{Dic}(\text{Pref}_{>q_1}(x^1)) \).
For $j > 1$: as we shall see in the proof of Theorem 2.10 below, in $z^{j-1}$ there is an interval of $A = 3\sqrt{\ell}$ positions that contains the starting position of a red block in at least $\ell/3$ regular green blocks. One of the positions of the interval is the starting point of a branch of at least $l/3A = \sqrt{\ell}/9 \geq m$ red blocks. Thus there is a red block of size $m$ which appears nowhere in $z^1, \ldots, z^{j-1}$ by (P2) and is not a prefix of $z^j$, thus it is a word of $\text{Dic}(0z^1 \ldots z^{j-1})$ not in $\text{Dic}(z^1 \ldots z^{j-1} \text{Pref}_{>q_j}(x^j))$. \qed

We can now show that $w$ has compression speed $O(|w|/\ell)$ by giving an upper bound on the size of the dictionary of $w$.

**Lemma 5.8.** The compression speed $|\text{Dic}(w)|$ of $w$ is at most

$$\frac{3 + \sqrt{3}}{2} \cdot \frac{|w|}{\ell}.$$  

**Proof.** The definition of the integers $q_j$ guarantees that the parsing resynchronizes at each beginning of a new chain $z^j$.  

In a chain $z^j$, the definition of $q^0_j(j)$ guarantees that this gadget, if present, will be parsed in exactly one green block, and after that the subsequent gadgets $g^i_c(j)$ also.

Similarly, (P2) together with the fact that gadgets are only inserted in the second half of a chain (thus, after more than $m$ green blocks) imply that the possible gadgets $g^i_c(j)$ for $i > 0$ are also parsed in exactly one green block.

For each chain $z^j$, the parsing tree consists in a main path of size $l$ (regular green blocks) together with another path of size $\leq \ell/2$ corresponding to the gadgets $g^i_c(j)$. The compression speed cannot be worse than in the (hypothetical) case where these two paths begins at depth 0, for all $j$. In that case, there are $\leq (3/2)l$ green blocks for each chain, and a size

$$|z^j| \geq \frac{l(l+1)}{2} + \frac{l(l + \frac{1}{2})}{2} \geq \frac{5}{8}l^2.$$  

Since the number of chains is $2^p = n/\ell^2$, in that (hypothetical) worst case the number of green blocks in $w'$ is at most $3n/2\ell$ and $|w'| \geq 5n/8$. The $\leq 3n/8$ trailing zeroes of $w$ are parsed in at most $\sqrt{3n}/2 \leq (\sqrt{3}/2)n/l$ green blocks. Hence the compression speed of $w$ is at most $(3/2 + \sqrt{3}/2)(n/l)$. \qed

### 5.3 Proof of the main theorem

(Recall the choice of parameters $n$, $\ell$, $p$, $k$, $\gamma$ and $m$ defined in Figure 6)
We now prove the lower bound of Theorem 2.10. Recall that \( z^j \) denote the \( j \)-th chain of \( w \). We will write \( w^j_i \) the \( i \)-th regular block of the chain \( z^j \). As in the previous section, we will distinguish junctions over two consecutive regular blocks (type 1); junctions starting in a gadget and ending in a regular block (type 2); and junctions starting in a regular block and ending in a gadget (type 3).

The next proposition is the core of the argument, and Theorem 2.10 will follow easily. The proposition is a corollary of lemmas that we will show afterwards.

**Proposition 5.9.** Let \( f(i) \) be the maximal size of an offset-\( i \) (red) block included in a regular green block.

- If \( i \leq 2k\sqrt{l} \) then \( f(i) \leq \frac{i}{2} + 4k\sqrt{l} + 2m + 1 \).
- Otherwise, \( f(i) \leq 2\sqrt{l} + 3k + 7m + \frac{2kl}{i-4m-2} \).

**Proof.** The first point is a consequence of Lemmas 5.10 and 5.11. The second point is exactly Lemma 5.13. \( \square \)

With Proposition 5.9 in hand, let us prove the main theorem.

**Proof of Theorem 2.10.** We will show that each chain \( z^j \) in \( 0w \) is parsed in at least \( \frac{1}{54} \frac{l^{3/2}}{2} \) blocks, thus

\[
|\text{Dic}(0w)| \geq 2^h \frac{1}{54} \frac{l^{3/2}}{2} = \frac{1}{54} \frac{|w|}{\sqrt{l}}.
\]

Fix an index \( j \). In order to prove that the chain \( z^j \) is parsed in at least \( \frac{1}{54} \frac{l^{3/2}}{2} \) red blocks, we first prove that in every regular green block of size larger than \( 2l/3 \) in the chain \( z^j \), there is an interval of positions \([2l/3 - A, 2l/3]\) (with \( A = 3\sqrt{l} \)), such that there is at least one offset-\( i \) (red) block for \( i \in [2l/3 - A, 2l/3] \). Indeed, by Proposition 5.9, for any \( i < 2l/3 - A \), the maximal size \( f(i) \) of a red block starting at position \( i \) satisfies \( i + f(i) \leq 2l/3 \).

Therefore, since the red blocks starting at position \( i \geq 2l/3 - A \) are of size at most \( f(2l/3 - A) \leq 3\sqrt{l} \), a regular green block of \( z^j \) of size \( h \) is covered by at least \( (h - 2l/3)/(3\sqrt{l}) \) red blocks. Thus the number of red blocks in the parsing of \( z^j \) is at least

\[
\sum_{h=2l/3}^{i} \frac{h - 2l/3}{3\sqrt{l}} \geq \frac{1}{54} \frac{l^{3/2}}{2}.
\]

Now we prove Proposition 5.9 thanks to the next four lemmas. The first two show that the gadgets do their job: indeed, for small \( i \) (the indices \( i \) covered by the gadgets), the offset-\( i \) blocks are not too large. For that, we first bound the number of violations.
Lemma 5.10. For any $i \in [0, 2k\sqrt{l}]$ and $j \in [1, 2^p]$, the number of $i$-violations in the chain $z^j$ is at most $\frac{1}{2} + 2m + 1 + 2k\sqrt{l}$.

Proof. We fix $j$ and focus on the number of $i$-violations in the chain $z^j$. Recall that $s_j$ denotes the number of regular blocks in the chain $z^j$ ($s_j \leq l$).

If no gadgets have been added during the execution of the algorithm, then for all $i \in [0, 2k\sqrt{l}]$, the number of $i$-violations is $\leq s_j/2 \leq l/2$.

Otherwise, we distinguish on the type of the most frequent violation ($i = 0$ or $i > 0$).

- Case 1: the most frequent violations are 0-violations. In that case, a proof similar to the case 1 of Lemma 4.5 (with 0 replaced by $a = x^j[0]$ the first letter of $x^j$) shows that the number of $i$-violations for any $i \in [0, 2k\sqrt{l}]$ is $\leq s_j/2 \leq l/2$.

- Case 2: the most frequent violations are $i$-violations for some $i > 0$. Let us see how the parsing of $0w$ splits the gadgets. As in the definition of the gadgets, let $m' = \max(i,m)$ and $v = x^j[0..m - 1]1^j$. When the first gadget $g^j_0(j) = x^j[0..m' - 1]\bar{x}_{m'}^j$, is added, the red parsing splits the gadget $g^j_0(j)$ between $x^j_{<i}$ and $x^j[i..m' - 1]\bar{x}_{m'}^j$, because the gadget is added before a regular block with an $i$-violation. Furthermore, $x^j[i..m' - 1]\bar{x}_{m'}^j$ is not split by the parsing, because at that moment in the algorithm, the number of $i$-violations in the previous regular green blocks is $s_j/2 \geq m' - i$, so that, as the position $i$ has been seen $\geq m' - i$ times, the word $x^j[i..m' - 1]$ is already in the dictionary of $0w$. Similarly, the gadget $g^j_{1}(j) = x^j[0..m' - 1]\bar{x}_{m'}^j.v[0..c - 1]$ is split by the red parsing between $x^j_{<i}$ and $x^j[i..m' - 1]\bar{x}_{m'}^j.v[0..c - 1]$, with the additional property that this second part is not split by the parsing.

But for the gadget $g^j_{2m+1}(j)$, the second part $x^j[i..m' - 1]\bar{x}_{m'}^j.x^j[0..m - 1]1^{m+1}$ is parsed in exactly one block because this factor does not appear anywhere in a regular block because of $1^{m+1}$ (cf. (P2)) nor in a gadget of a preceding chain because of $x^j[0..m - 1]$ (cf. (P2) again). From that moment on, each $i$-violation creates a 0-violation. The number of green blocks that are both 0-violated and $i$-violated is at most $i \leq 2k\sqrt{l}$. Thus, at most $2k\sqrt{l}$ more gadgets fail to kill the corresponding $i$-violation. The total number of “failing” gadgets in the chain $z^j$ is at most $2m + 1 + 2k\sqrt{l}$.

\[\]
Now, the next two results show that, for large \( i \), the size of offset-i blocks is small. First, we need to bound the number of junction blocks ending at position \( i - 1 \).

**Lemma 5.12.** Let \( j \) be fixed and \( i > 2k\sqrt{l} \). Let \( uu' \) be a junction block of type 1 between two regular green blocks \( w_j^i \) and \( w_{j+1}^i \), ending at position \( i - 1 \) in \( w_{j+1}^i \) (thus \( |u'| = i \)). Then \( |u| \leq \log(kl) - \log(i - 4m - 2) \).

In particular, the number of such blocks is upper bounded by \( \frac{2kl}{i - 4m - 2} \).

**Proof.** Let \( v \) be the prefix of size \( 4m + 1 \) of \( u' \) (which is also the prefix of \( x^j \)). We claim that all the prefixes of \( uu' \) of size \( \geq |uv| \) are junction blocks of type 1 or 3 only (except possibly for one of type 2), with only \( u \) on the left side of the junction. Indeed, recalling the red parsing of gadgets explained in the proof of Lemma 5.10 and Property (P2), we distinguish the following cases:

1. \( uv \) cannot be completely included in a regular block, otherwise \( v[0..m - 1] \) would appear both at positions 0 and \( p > 0 \) in \( x^j \), which contradicts Property (P2);

2. \( uv \) cannot be completely included in a gadget:
   - if the gadget is \( g_0^j(j) \), then \( v \) would contain \( a^{m+1} \) for some \( a \in \{0, 1\} \),
   - if the gadget is \( g_b^j(j) \) for \( b > 0 \), let \( m' = \max(b, m) \): the red parsing splits this gadget between \( x^j[0..b - 1] \) and \( x^j[b..m' - 1]x_{m'}^jx^j[0..m - 1]1^d \). Then \( uv \) is not contained in the first part by (P2), nor in the second part since it cannot contain \( 1^{m+1} \);

3. if \( uv \) is a type 1 junction but not split between \( u \) and \( v \), it is impossible because the three possible cases lead to a contradiction:
   - if \( u \) goes on the right, then \( v \) would appear at another position \( p > 0 \) in \( x^j \),
   - if \( v \) goes on the left by at least \( m \), then \( v[0..m - 1] \) would again appear at two different positions in \( x^j \),
   - if \( v \) goes on the left by less than \( m \), then it goes on the right by more than \( m \) and \( v[3m + 1..4m] \) would again appear at two different positions in \( x^j \);

4. if \( uv \) is a type 2 junction but not split between \( u \) and \( v \), it is again impossible:
   - if \( u \) goes on the right, then \( v \) would appear at another position \( p > 0 \) in \( x^j \),
   - if \( v \) goes on the left by at least \( 3m + 1 \), in case of \( g_0^j(j) \) then \( v \) would contain \( a^{2m+1} \) and in case of \( g_b^j(j) \) then \( v \) would contain \( 1^{m+1} \) (recall where the red parsing splits this gadget),
   - otherwise, \( v \) goes on the right by at least \( m + 1 \), and \( v[3m + 1..4m] \) would appear at two different positions \( x^j \);
5. if $uv$ is a type 3 junction, first remark that the gadget is of the form $g^b(j)$ for $b > 0$ (because, for gadgets of the form $g^0(j)$, the red parsing starts at position 0 of the gadget). If $uv$ is not split between $u$ and $v$, it is once again impossible:

- if $u$ goes on the right, the red parsing of the gadget stops after $x^j_{<b}$ and $v$ would appear in $x^j$ at a non-zero position,
- similarly, if $v$ goes on the left by less than $m$, then $v[m..2m-1]$ would again appear at two different positions in $x^j$,
- if $v$ goes on the left by at least $m$, then $v[0..m-1]$ would again appear at two different positions in $x^j$.

Remark finally that all parsings of type 2 junctions have different sizes on the left. Therefore, at most one can contain $u$ on the left. The claim is proved.

Thus $u$ appears at least $i - 4m - 2$ times as a suffix of a regular green block.

Remark that Property (P1) implies that factors of size more than $k$ appear at most $k\sqrt{l}$ times in $x^j$. Thus, since $i - 4m - 2 > k\sqrt{l}$, we have $|u| \leq k$. Hence by Property (P1), the number of occurrences of $u$ is upper bounded by $kl/2^{|u|}$. Therefore

$$i - 4m - 2 \leq \frac{kl}{2^{|u|}}$$

$$|u| \leq \log(kl) - \log(i - 4m - 2),$$

which proves the first part of the lemma.

The number of such blocks is then upper bounded by the number of words of size $\leq \log(kl) - \log(i - 4m - 2)$, that is, $\frac{2kl}{i - 4m - 2}$.

The last lemma completes the preceding one: if an offset-$i$ block is large, then a lot of blocks have to end at position $i - 1$ and too many of their prefixes would have to be in different green blocks.

**Lemma 5.13.** For any $j \in [1, 2^p]$ and any $i > 2k\sqrt{l}$, the size of an offset-$i$ block included in a regular green block of the chain $z^j$ is at most

$$2\sqrt{l} + 3k + 7m + \frac{2kl}{i - 4m - 2}.$$

**Proof.** We argue as in Proposition 4.7. Let $u$ be an offset-$i$ block included in a regular green block of the chain $z^j$. We show as before that the $|u| - 3m$ predecessors of $u$ of size $\geq 3m$ have to start at position $i$ in regular green blocks. Indeed, let $v$ be a prefix of size $\geq 3m$ of $u$; let us analyse the different cases:

- If $v$ is included in a regular green block, then it has to start at position $i$ by Property (P2);
- $v$ cannot be included in a gadget since it would lead to a contradiction:
  - in gadgets of type $g^0(j)$, $v$ would contain $a^{m+1}$ for some letter $a \in \{0, 1\}$,
in gadgets of type $g_c^b(j)$ (for $b \in [0,2k\sqrt{l}]$), either $v$ would contain $1^{m+1}$ or a factor of $x^j$ of size $m$ and at a position different from $i$;

- If $v$ is included in a junction block of type 1, then $v$ starts at position $i$ in the left regular block, otherwise either $v[0..m-1]$ would be in the left regular block at a position different from $i$, or $v[m-1..2m-2]$ would be in the right regular block at a position $\leq 3m < i$;

- $v$ cannot be included in a junction block of type 2. Indeed, it cannot go by $\geq m$ on the right (by (P2)), thus it goes on the left by at least $2m + 1$: for $g_c^b(j)$ it would contain $a^{m+1}$ (for some $a \in \{0, 1\}$), and for $g_c^b(j)$ ($b > 0$), it would either contain $1^{m+1}$, or a factor of $x^j$ of size $m$ at a position $< m' \leq i$;

- If $v$ is included in a junction block of type 3, then $v$ starts at position $i$ in the left regular block, otherwise either $v[0..m-1]$ would be in the left (regular) block at a position different from $i$, or, by the red parsing, the gadget is of type $g_c^b(j)$ (for $b > 0$) and $v[m-1..2m-2]$ would be included in $x^j$ at a position $\leq 3m < i$.

Thus, at least $|u| - 3m$ red blocks end at position $i - 1$ in the regular blocks of $z^j$. Among them:

- By Lemma 5.12 at most $\frac{2kl}{i-4m-2}$ of them are junctions of type 1.

- At most $2m$ of them are junctions of type 2, since from the $(2m + 1)$-th gadget on, the type 2 junctions end at position 0 (in case of gadgets $g_c^b(j)$ for $b > 0$, see the proof of Lemma 5.10) or $\leq m + 1 \leq i - 1$ (in case of gadgets $g_c^b(j)$).

- There is no junction of type 3 by definition.

Overall, at least $|u| - 5m - \frac{2kl}{i-4m-2}$ of them are offset blocks, ending at position $i - 1$. We call the set of such blocks $M$. Let

$$A = \frac{|u| - 7m - \frac{2kl}{i-4m-2}}{2}$$

and

$$S = \{u \in \mathcal{P}(M) \text{ containing } x^j[i - A - 2m..i - A - 1]\}.$$

We say that a red block $w$ is **problematic** if $w \in S$ but the part of $w$ corresponding to the factor $x^j[i - A - 2m..i - A - m - 1]$ is not completely included in a regular green block. We show that the number of problematic blocks is at most $2k\sqrt{l} + 2m + 1$.

1. The number of problematic blocks that overlap a gadget $g_c^b(j) = ua^c$ in the red parsing is at most $m + 1$. Indeed, $ua^c$ is never split by the red parsing, therefore for $c \geq m + 1$, a red block that overlaps $g_c^b(j)$ would contain $a^{m+1}$, which is not a factor of $x^j$. 

35
2. For $b > 0$ (recall that $b \leq 2k\sqrt{l}$), note that the red parsing splits the gadget $g_b^c(j)$ after $x^j[0..b-1]$.

- Observe first that the number of problematic blocks that overlap the second part of the gadget ($g_b^c(j)_{\geq b}$) is at most $m$. Indeed, this part is not split by the red parsing, therefore for $c \geq m$ a red block that overlaps this part would contain $x^j\cdot x^j[0..m-1]$, which is not possible since the position of the word $x^j[0..m-1]$ should be 0 by Property (P2).

- The number of problematic blocks that appear completely included in the first part of a gadget ($g_b^c(j)_{< b}$) is at most $b$. Otherwise, the red parsing creates at least one red block completely included in the first part $x^j[0..b-1]$ of the gadget, and we claim that this can happen at most $b$ times. Indeed, each time the parsing falls in this case, the last red block included in the first part of the gadget has to end at position $b-1$, but the size of this block has to be different each time, so that this second case can occur at most $b$ times. Finally, each time a gadget is parsed, at most one of the red blocks included in the first part of the gadget can be a word of $S$ by Property (P2).

3. There is no problematic blocks that are junction blocks of type 1 or 3. Indeed, if it were the case, the right part of the junction would be of size $\leq m-1$ since otherwise the problematic block would contain $ax^j[0..m-1]$ for some letter $a$, which is not possible. Therefore, within the problematic block, the factor $x^j[i - A - 2m..i - A - m - 1]$ appears on the left side of the junction and is thus included in a regular block.

4. The number of problematic blocks that are junction blocks of type 2 has already been considered when considering the gadgets.

All the red blocks corresponding to words of $S$ and that are not problematic have to appear in distinct regular green blocks by Property (P2). As before, a word of $S$ is obtained by choosing its beginning before the interval and its end after, so that

$$|S| \geq (|u| - 5m - \frac{2kl}{i - 4m - 2} - (A + 2m)) \cdot A.$$ 

Therefore:

$$|S| - (2k\sqrt{l} + 2m + 1) \leq l$$

$$\left(\frac{|u| - 7m - \frac{2kl}{i - 4m - 2}}{2}\right)^2 - (2k\sqrt{l} + 2m + 1) \leq l,$$

so that

$$|u| \leq 2\sqrt{l + 2k\sqrt{l} + 2m + 1} + 7m + \frac{2kl}{i - 4m - 2} \leq 2\sqrt{l + 3k + 7m + \frac{2kl}{i - 4m - 2}}.$$
6 Infinite words

The techniques on finite words developed in the preceding sections can almost be used as a black box to prove the one-bit catastrophe for infinite words (Theorem 2.6). Our aim is to design an infinite word \( w \in \{0,1\}^\mathbb{N} \) for which the compression ratios of the prefixes tend to zero, whereas the compression ratios of the prefixes of \( 0w \) tend to \( \epsilon > 0 \). In Section 4 we concatenated the bricks obtained in Section 3; now, we concatenate an infinite number of bricks of Section 5 of increasing size (with the parameters that gave the one-bit catastrophe on finite words). As before, each chain of size \( l \) will be parsed in \( \Theta(l) \) green blocks and \( \Theta(l^{3/2}) \) red blocks. To guarantee that the compression ratio always remains close to zero in \( w \) and never goes close to zero in \( 0w \), the size of the bricks mentioned above will be adjusted to grow neither too fast nor too slow, so that the compression speed will be locally the same everywhere.

We will need an infinite sequence of families \( (F_i)_{i \geq 0} \) of words similar to that of Section 5; thus we will need infinite sequences of parameters to specify them.

- For \( i \geq 0 \), the size of words in \( F_i \) will be \( l_i = l_0.2^i \), for \( l_0 \) sufficiently large.
- Let \( p_i = \sqrt{l_i}/(9\gamma) - 2 \log l_i \), where \( \gamma \geq 10 \) is a constant. For \( i > 0 \), the number of words in \( F_i \) will be \( |F_i| = 2^{p_i} - 2^{p_{i-1}} \) (and \( |F_0| = 2^{p_0} \)). Remark that \( \sum_{j=0}^{i} |F_j| = 2^{p_i} \) and \( |F_{i+1}| \sim |F_i|^{\sqrt{2}} \).
- The parameter \( k_i = (\log l_i)/2 \) will be the maximal size of words in Property P1(i) below.
- The parameter \( m_i = \gamma p_i \) will be the size of words in Property P2(i) below.

We shall later show that there exists an infinite sequence \( F = (F_i)_{i \geq 0} \) matching these parameters and satisfying some desired properties (generalized versions of Properties (P1) and (P2), see below). But from an arbitrary sequence \( (F_i)_{i \geq 0} \), let us first define the “base” word from which \( w \) will be constructed.

**Definition 6.1.** Given a sequence \( F = (F_i)_{i \geq 0} \) where each \( F_i \) is a family of words, we denote by \( w_F \) the word

\[
w_F = \prod_{i=0}^{\infty} \prod_{x \in F_i} \text{Pref}_{> q^i_x}(x)
\]

where \( q^i_x = \max \{ a : x < a \text{ is a prefix of a word in } \cup_{j<i} F_j \} \).

For a particular sequence \( F = (F_i) \), the word \( w \) will be equal to \( w_F \) with some gadgets inserted between the prefixes as in the previous sections. The sequence \( F \) that we shall consider will be a sequence of families of random words which will satisfy the following properties (Lemma 6.2 below shows that these properties are true with high probability).
P1(i): For all \( x \in F_i \), for all words \( u \) of size at most \( k_i \), \( \text{Occ}_x(u) \leq k_i l_i/2^{|u|} \).

(P1'): For all \( i \geq 0 \), P1(i).

P2(i): Any factor \( u \) of size \( m_i \) appears in at most one word of \( \bigcup_{j \leq i} F_j \), and within that word at only one position.

(P2'): For all \( i \geq 0 \), P2(i).

Again, (P2') guarantees a kind of “independence” of the families \( F_0, F_1, \ldots \), whereas (P1') is a de Bruijn-style “local” property on each word of each family \( F_i \).

Our first lemma shows that there exists a sequence \( F = (F_i)_{i \geq 0} \) satisfying (P1') and (P2').

**Lemma 6.2.** For every \( i \geq 0 \), let \( F_i \) be a set of \( 2^{p_i} - 2^{p_i-1} \) words of size \( l_i \) (and \( 2^{p_0} \) words of size \( l_0 \) for \( F_0 \)) taken uniformly and independently at random.

Then the probability that \( F \) satisfies Properties (P1') and (P2') is non-zero.

**Proof.** Let us show that the probability that \( F \) satisfies (P1') is \( > 1/2 \), and similarly for (P2'). We only show it for (P2'), as an analogous (and easier) proof gives the result for (P1') as well.

By Lemma 5.3, the probability that \( F \) does not satisfy P2(i) is less than \( 2/l_i = 2^{1-i}/l_0 \). Thus, by union bound, the probability that all P2(i) are satisfied is larger than

\[
1 - \sum_{i=0}^{\infty} \frac{2^{1-i}}{l_0} = 1 - \frac{4}{l_0}.
\]

From now on, we consider a sequence of families \( F = (F_i)_{i \geq 0} \), with parameters \((l_i, p_i)_i\), that has both Properties (P1') and (P2') for the parameters \((m_i, k_i)_i\) defined above. Remark that the integers \( q^i_x \) defined in Definition 6.1 satisfy \( q^i_x \leq m_i \) thanks to Property P2(i).

The word \( w \) that we consider is the word \( w_F \) (Definition 6.1) where gadgets have possibly been added between the regular green blocks exactly as in the algorithm of Section 5. Since \( F \) satisfies (P1') and (P2'), and the parameters \( l_i, p_i \) fall within the range of Theorem 2.10, it can be shown as in Section 6 that a chain of \( w \) coming from \( F_i \) will be parsed in \( \geq l_i^{3/2}/54 \) red blocks in \( w \) but in only \( l_i/2 \) green blocks. The following two lemmas show Theorem 2.6 i.e. that \( w \) satisfies the one-bit catastrophe. We begin with the upper bound on the compression ratio of \( w \), before proving the lower bound for \( 0w \) in Lemma 6.4.

**Lemma 6.3.** \( \rho_{\text{sup}}(w) = 0 \).

**Proof.** By definition (Definition 2.5),

\[
\rho_{\text{sup}}(w) = \limsup_{n \to \infty} \rho(w_{<n}),
\]
Therefore we need to show that \( \rho(w_{<n}) = G(\log G)/n \) tends to zero, where \( G = |\text{Dic}(w_{<n})| \) is the number of green blocks in the parsing of \( w_{<n} \). Let us evaluate this quantity for a fixed \( n \).

Let \( j \) and \( q \) be the integers such that the \( n \)-th bit of \( w \) belongs to the \( q \)-th chain of the \( j \)-th family, or in other terms, that \( w_{<n} \) is the concatenation of the chains coming from \( \cup_{i<j} F_i \) and of the first \( q - 1 \) chains of \( F_j \), together with a piece of the \( q \)-th chain of \( F_j \).

We first give a lower bound on \( G \). We can now bound the compression ratio of \( w_{<n} \).

\[
\sum_{j=m_i}^{l_i} j \geq \frac{l_i^2 - m_i^2}{2} = \frac{l_i^2}{2} - o(l_i^2).
\]

Thus, using \( l_j = 2l_{j-1} \), we get:

\[
n + o(n) \geq \sum_{i=0}^{j-1} |F_i| \frac{l_i^2}{2} + (q - 1) \frac{l_j^2}{2} \geq \frac{l_j^2}{2}(|F_{j-1}| + q)
\]

as soon as \( 2(q - 1) \geq q/2 \), that is, \( q \geq 2 \). (We shall take care of the case \( q = 1 \) below.)

On the other hand, when all possible gadgets are added, each chain has a size at most \( 5l_j^2/8 + o(l_j^2) \) (see Section 5.2). Using the fact that \( |F_{i+1}| \sim |F_i|^{\sqrt{2}} \) (i.e. the growth of the sequence \( (|F_i|)_{i \geq 0} \) is more than exponential), we obtain the following upper bound:

\[
n - o(n) \leq \frac{5}{8} \left( \sum_{i=0}^{j-1} |F_i|l_i^2 + ql_j^2 \right) \leq \frac{5}{8} (2|F_{j-1}|l_{j-1}^2 + ql_j^2) = \frac{5}{4} (|F_{j-1}| + 2q)l_{j-1}^2.
\]

In particular,

\[
\log G \leq \log n \leq 2 \log |F_{j-1}|.
\]

Let us now bound the number of green blocks. A chain coming from \( F_i \), with gadgets, is parsed in at most \( 3l_i/2 \) blocks. Hence

\[
G \leq \frac{3}{2} \left( \sum_{i=0}^{j-1} |F_i|l_i + ql_j \right) \leq \frac{3}{2} (2|F_{j-1}|l_{j-1} + 2ql_{j-1}) = 3l_{j-1}(|F_{j-1}| + q).
\]

We can now bound the compression ratio of \( w_{<n} \):

\[
\rho(w_{<n}) = \frac{G \log G}{n} \leq \frac{6}{l_{j-1}} \cdot 2 \log |F_{j-1}| \xrightarrow{n \to \infty} 0.
\]

Finally, for the case \( q = 1 \), looking back at the inequalities above we have:

\( n + o(n) \geq |F_{j-1}|l_{j-1}^2/2 \) and \( G \leq 3l_{j-1}(|F_{j-1}| + 1) \), thus \( \rho(w_{<n}) \) again tends to zero.
Finally we turn to the lower bound on the compression ratio of $0^w$.

**Lemma 6.4.** $\rho_{\inf}(0^w) \geq 2/(1215\gamma)$.

**Proof.** Define $j$ and $q$ as in the proof of Lemma 6.3, we want to give a lower bound on $(R \log R)/n$, where $R = |\text{Dic}(0^w_{<n})|$ is the number of red blocks in the parsing of $0^w_{<n}$. The upper bound for $n$ given there still hold:

$$n - o(n) \leq \frac{5}{4}(|F_{j-1}| + 2q)l_{j-1}^3.$$

Let us now give a lower bound on $R$. Suppose for now that $q \geq 4$, so that $2\sqrt{2}(q - 1) \geq 2q$. The proof of Theorem 2.10 in Section 5 shows that each chain coming from a family $F_i$ is parsed in at least $\epsilon l_i^{3/2}$ red blocks, where $\epsilon = 1/54$. Hence:

$$R \geq \sum_{i=0}^{j-1} \epsilon|F_i|l_i^{3/2} + \epsilon(q - 1)l_{j-1}^{3/2} \geq \epsilon(|F_{j-1}|l_{j-1}^{3/2} + 2\sqrt{2}(q - 1)l_{j-1}^{3/2}) \geq \epsilon(|F_{j-1}| + 2q)l_{j-1}^{3/2}.$$ 

Therefore,

$$\frac{R \log R}{n} \geq \frac{4\epsilon}{5} \log |F_{j-1}| \sqrt{l_{j-1}} \sim \frac{4\epsilon}{5 \times 9\gamma}.$$ 

In the case $q \leq 3$, we have $q << |F_{j-1}|$ and the same bound holds. \qed

7 Future work

A word on what comes next. As mentioned in the introduction, we have privileged “clarity” over optimality, hence constants can undoubtedly be improved rather easily. In that direction, a (seemingly harder) question is to obtain $\rho_{\sup}(w) = 0$ and $\rho_{\inf}(0^w) = 1$ in Theorem 2.6. The main challenge though, to our mind, is to remove the gadgets in our constructions. Remark that the construction of Section 4 can also be performed with high probability with a random word instead of a de Bruijn sequence (that is what we do in Section 5 in a more general way). Thus, if we manage to get rid of the gadgets using the same techniques presented here, this would mean that the “weak catastrophe” is the typical case for optimally compressible words. Simulations seem to confirm that conclusion. But, as a hint that removing the gadgets may prove difficult, Remark 4.8 emphasizes the vastly different behaviour of the LZ-parsings on $1^w$ with and without gadgets.

8 Acknowledgments

We want to thank Elvira Mayordomo for sharing, long time ago, the “one-bit catastrophe” question with us. The long discussions with Sophie Laplante have
helped us present our results in a more readable (less unreadable) way. We thank Lucas Boczkowski for his close scrutiny of parts of this paper, and Olivier Carton for useful discussions.

References

[1] Mateo Aboy, Roberto Hornero, Daniel E. Abásolo, and Daniel Álvarez. Interpretation of the Lempel-Ziv complexity measure in the context of biomedical signal analysis. *IEEE Trans. Biomed. Engineering*, 53(11):2282–2288, 2006.

[2] Christopher Hoobin, Simon J. Puglisi, and Justin Zobel. Relative Lempel-Ziv factorization for efficient storage and retrieval of web collections. *Proc. VLDB Endow.*, 5(3):265–273, November 2011.

[3] Juha Kärkkäinen, Dominik Kempa, Yuto Nakashima, Simon J. Puglisi, and Arseny M. Shur. On the size of Lempel-Ziv and Lyndon factorizations. In Heribert Vollmher and Brigitte Vallée, editors, *34th Symposium on Theoretical Aspects of Computer Science, STACS 2017, March 8-11, 2017, Hannover, Germany*, volume 66 of *LIPIcs*, pages 45:1–45:13. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.

[4] J. Lathrop and M. Strauss. A universal upper bound on the performance of the Lempel-Ziv algorithm on maliciously-constructed data. In *Proceedings of the Compression and Complexity of Sequences 1997, SEQUENCES ’97*, pages 123–135, Washington, DC, USA, 1997. IEEE Computer Society.

[5] Abraham Lempel and Jacob Ziv. On the complexity of finite sequences. *IEEE Trans. Information Theory*, 22(1):75–81, 1976.

[6] María López-Valdés. Lempel-Ziv dimension for Lempel-Ziv compression. In *Proceedings of the 31st International Conference on Mathematical Foundations of Computer Science*, MFCS’06, pages 693–703, Berlin, Heidelberg, 2006. Springer-Verlag.

[7] María López-Valdés and Elvira Mayordomo. Dimension is compression. *Theory Comput. Syst.*, 52(1):95–112, 2013.

[8] Jack H. Lutz. Dimension in complexity classes. *SIAM J. Comput.*, 32(5):1236–1259, 2003.

[9] Jack H. Lutz and Elvira Mayordomo. Computing absolutely normal numbers in nearly linear time. *CoRR*, abs/1611.05911, 2016.

[10] Elvira Mayordomo, Philippe Moser, and Sylvain Perifel. Polylog space compression, pushdown compression, and Lempel-Ziv are incomparable. *Theory Comput. Syst.*, 48(4):731–766, 2011.
[11] Larry A. Pierce II and Paul C. Shields. *Sequences Incompressible by SLZ (LZW), Yet Fully Compressible by ULZ*, pages 385–390. Springer US, Boston, MA, 2000.

[12] Yi Zhang, Junkang Hao, Changjie Zhou, and Kai Chang. Normalized Lempel-Ziv complexity and its application in bio-sequence analysis. *Journal of Mathematical Chemistry*, 46(4):1203–1212, November 2009.

[13] J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Trans. Inf. Theor.*, 23(3):337–343, September 1977.

[14] J. Ziv and A. Lempel. Compression of individual sequences via variable-rate coding. *IEEE Trans. Inf. Theor.*, 24(5):530–536, September 1978.