




L-systems for Measuring Repetitiveness

Gonzalo Navarro   

Department of Computer Science, University of Chile, Chile
Centre for Biotechnology and Bioengineering (CeBiB), Chile

Cristian Urbina   

Department of Computer Science, University of Chile, Chile
Centre for Biotechnology and Bioengineering (CeBiB), Chile

Abstract

In order to use them for compression, we extend L-systems (without ε -rules) with two parameters d and n , and also a coding τ , which determines unambiguously a string $w = \tau(\varphi^d(s))[1 : n]$, where φ is the morphism of the system, and s is its axiom. The length of the shortest description of an L-system generating w is known as ℓ , and it is arguably a relevant measure of repetitiveness that builds on the self-similarities that arise in the sequence.

In this paper, we deepen the study of the measure ℓ and its relation with a better-established measure called δ , which builds on substring complexity. Our results show that ℓ and δ are largely orthogonal, in the sense that one can be much larger than the other, depending on the case. This suggests that both mechanisms capture different kinds of regularities related to repetitiveness.

We then show that the recently introduced NU-systems, which combine the capabilities of L-systems with bidirectional macro schemes, can be asymptotically strictly smaller than both mechanisms for the same fixed string family, which makes the size ν of the smallest NU-system the unique smallest reachable repetitiveness measure to date. We conclude that in order to achieve better compression, we should combine morphism substitution with copy-paste mechanisms.

2012 ACM Subject Classification Mathematics of computing \rightarrow Combinatorics on words; Theory of computation \rightarrow Data compression

Keywords and phrases L-systems, String morphisms, Repetitiveness measures, Text compression

Digital Object Identifier 10.4230/LIPIcs.CPM.2023.14

Funding *Gonzalo Navarro*: Funded by Basal Funds FB0001 and Fondecyt Grant 1-200038, Chile.
Cristian Urbina: Funded by Basal Funds FB0001 and ANID National Doctoral Scholarship – 21210580, Chile.

1 Introduction

In areas like Bioinformatics, it is often necessary to handle big collections of highly repetitive data. For example, two human genomes share 99.9% of their content [23]. In another scenario, for sequencing a genome, one extracts so-called *reads* (short substrings) from it, with a “coverage” of up to 100X, which means that each position appears on average in 100 reads.¹ There is a need in science and industry to maintain those huge string collections in compressed form. Traditional compressors based exclusively on *Shannon’s entropy* are not good for handling repetitive data, as they only exploit symbol frequencies when compressing. Finding good measures of repetitiveness and also compressors exploiting this repetitiveness has then become a relevant research problem.

A strong theoretical measure of string repetitiveness introduced by Kociumaka et al. [12] is δ , based on the substring complexity function. This measure has several nice properties: it

¹ <https://www.illumina.com/science/technology/next-generation-sequencing/plan-experiments/coverage.html>



is computable in linear time, monotone, resistant to string edits, insensitive to simple string transformations, and it lower-bounds almost every other theoretical or *ad-hoc* repetitiveness measure considered in the literature. Further, although $O(\delta)$ space is unreachable, there exist $O(\delta \log(n/\delta))$ -space representations supporting efficient pattern matching queries [12, 11], and this space is tight: no $o(\delta \log(n/\delta))$ -space representation can exist [12].

The idea that δ is a sound lower bound for repetitiveness is reinforced by the fact that it is always $O(b)$, where b is the size of the smallest *bidirectional macro scheme* generating a string [26]. Those macro schemes arguably capture every possible way of exploiting copy-paste regularities in the sequences. Some very recent works [19], however, explore other sources of repetitiveness, in particular *self-similarity*, and are shown to break the lower bound of δ .

The simplest of those schemes, which reuse the name *L-system* for simplicity [19], builds upon Lindenmayer systems [15, 16], in particular on the variant called CPD0L-systems. A CPD0L-system describes the language of the images, under a coding τ , of the powers of a non-erasing morphism φ starting from an string s (called the *axiom*), that is, the set $\{\tau(\varphi^i(s)) \mid i \geq 0\}$. The L-systems extend CPD0L-systems with two parameters d and n , so as to unambiguously determine the string $w = \tau(\varphi^d(s))[1 : n]$. The size of the shortest description of an L-system generating w in this fashion is called ℓ . Intuitively, ℓ works as a repetitiveness measure because any occurrence of the symbol a at level i expands to the same string at level $i + j$ for every j .

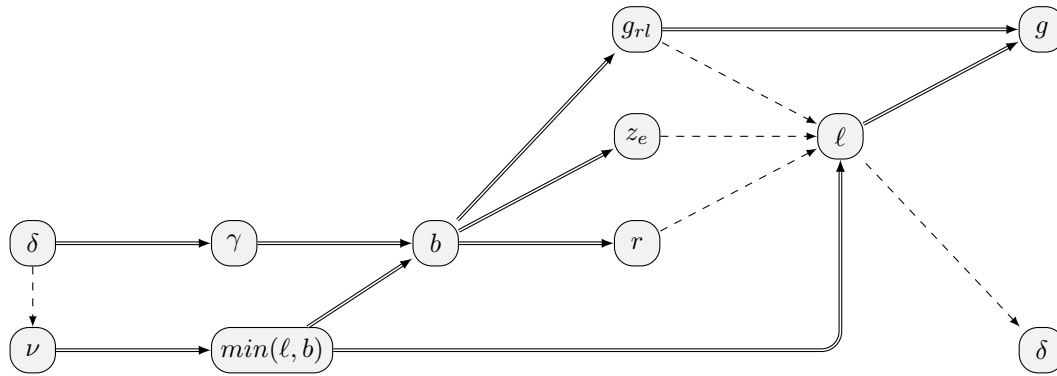
Since ℓ is a reachable measure of repetitiveness (because the L-system is a representation of w of size $O(\ell)$), there are string families where $\delta = o(\ell)$. Intriguingly, it has been shown [19] that there are other string families where $\ell = o(\delta)$, so (1) both measures are not comparable and (2) the lower bound δ does not capture this kind of repetitiveness. On the other hand, it is shown that $\ell = O(g)$, where g is the size of the smallest deterministic context-free grammar generating only w . This comparison is relevant because L-systems are similar to grammars, differing in that they have no terminal symbols, so their expansion must be explicitly stopped at level d and then possibly converted to terminals with τ .

Grammars provide an upper bound to repetitiveness that is associated with well-known compressors, so this upper bound makes ℓ a good measure of repetitiveness.

A more complex scheme that was also introduced [19] are NU-systems, which combine the power of L-systems with bidirectional macro schemes. The measure ν , defined as the size of the smallest NU-system generating w , naturally lower bounds both ℓ and b . The authors could not, however, find string families where ν is asymptotically better than both ℓ and b , so it was unclear if NU-systems are actually better than just the union of both underlying schemes.

In this paper we deepen the study of the relations between these new intriguing measures and more established ones like δ and g . Our results are as follows:

1. We show that ℓ can be much smaller than δ , by up to a \sqrt{n} factor, improving a previous result [19] and refuting their conjecture that $\ell = \Omega(\delta/\log n)$.
2. On the other hand, we expose string families where ℓ is larger than the output of several repetitiveness-aware compressors like the size g_{rl} of the smallest run-length context-free grammar, the size z_e of the greedy LZ-End parse [13], and the number of runs r in the Burrows-Wheeler Transform of the string [2]. We then conclude that ℓ is uncomparable to almost all measures other than g , which suggests that the source of repetitiveness it captures is largely orthogonal to the typical cut-and-paste of macro schemes.
3. We introduce a string family where ν is asymptotically strictly smaller than both ℓ and b , which shows that NU-systems are indeed relevant and positions ν as the unique smallest



■ **Figure 1** Asymptotic relations between ℓ , ν , and other repetitiveness measures. A double black arrow from v_1 to v_2 means that it always holds that $v_1 = O(v_2)$, and there exists a string family where $v_1 = o(v_2)$. A dashed arrow from v_1 to v_2 means that there exists a family where $v_1 = o(v_2)$.

reachable repetitiveness measure to date that captures both kinds of repetitiveness in non-trivial ways.

4. We study various ways of simplifying L-systems and show that, in most cases, we end up with a weaker repetitiveness measure. We also study some of those weaker variants of ℓ , which can be of independent interest.

Overall, our results contribute to understanding how to measure repetitiveness and how to exploit it in order to build better compressors. We summarize the state of ℓ and ν after this work in Figure 1.

2 Basic concepts

In this section we explain the basic concepts needed to understand the rest of the paper, from strings and morphisms to relevant repetitiveness measures.

2.1 Strings

An *alphabet* is a finite set of *symbols* and is usually denoted by Σ . A (finite) *string* w is a finite sequence $w[1]w[2]\dots w[n]$ of symbols where $w[i] \in \Sigma$ for $i \in [1, n]$, and its *length* is denoted by $|w| = n$. The *empty string*, whose length is 0, is denoted by ε . The set of all possible finite strings over Σ is denoted by Σ^* . If $x = x[1]\dots x[n]$ and $y = y[1]\dots y[m]$, the concatenation operation $x \cdot y$ (or just xy) yields the string $x[1]\dots x[n]y[1]\dots y[m]$. Let $w = xyz$. Then y (resp. x, z) is a *substring* (resp. *prefix, suffix*) of w . It is *proper* if it is not equal to w , and *non-trivial* if it is distinct from ε and w . The notation $w[i : j]$ stands for the substring $w[i]\dots w[j]$ if $i \leq j$, and ε otherwise. We also use the conventions $w[i : j] = w[1 : j]$ if $i < 1$, $w[i : j] = w[i : n]$ if $j > n$, and $w[i : j] = \varepsilon$ if $i > n$ or $j < 1$. Other convenient notations are $w[: i] = w[1 : i]$ and $w[i : :] = w[i : |w|]$ for prefixes and suffixes, respectively.

A (right) *infinite string* \mathbf{w} (we use boldface to emphasize them) over an alphabet Σ is a mapping from \mathbb{Z}^+ to Σ , and its length is called ω , which is greater than n for any $n \in \mathbb{Z}^+$. It is possible to define the concatenation $x \cdot \mathbf{y}$ if x is finite and \mathbf{y} infinite. The concepts of substring, prefix, and suffix carry over to infinite strings, with proper prefixes always being finite and suffixes always being infinite. The notations $\mathbf{w}[i]$, $\mathbf{w}[i : j]$, $\mathbf{w}[: i]$, and $\mathbf{w}[i : :] = \mathbf{w}[i]\mathbf{w}[i + 1]\dots$ also carry over to infinite strings.

2.2 Morphisms

The set Σ^* together with the (associative) concatenation operator and the (identity) string ε form a *monoid* structure $(\Sigma^*, \cdot, \varepsilon)$. A *morphism* on strings is a function $\varphi : \Sigma_1^* \rightarrow \Sigma_2^*$ satisfying $\varphi(x \cdot y) = \varphi(x) \cdot \varphi(y)$ for all x, y (i.e., a function preserving the monoid structure), where Σ_1 and Σ_2 are alphabets. To define a morphism on strings, it is sufficient to define how it acts over the symbols in its domain. The pairs $(a, \varphi(a))$ for $a \in \Sigma_1$, usually denoted $a \rightarrow \varphi(a)$, are called the *rules* of the morphism, and there are $|\Sigma_1|$ of them. If $\Sigma_1 = \Sigma_2$, then the morphism is called an *endomorphism*.

Let $\varphi : \Sigma_1^* \rightarrow \Sigma_2^*$ be a morphism on strings. Some useful definitions are *width*(φ) = $\max_{a \in \Sigma_1} |\varphi(a)|$ and *size*(φ) = $\sum_{a \in \Sigma_1} |\varphi(a)|$. A morphism is *non-erasing* if $\forall a \in \Sigma_1, |\varphi(a)| > 0$, *expanding* if $\forall a \in \Sigma_1, |\varphi(a)| > 1$, *k-uniform* if $\forall a \in \Sigma_1, |\varphi(a)| = k > 1$, and it is a *coding* if $\forall a \in \Sigma_1, |\varphi(a)| = 1$ (sometimes called a 1-uniform morphism).

Let $\varphi : \Sigma^* \rightarrow \Sigma^*$ be an endomorphism. Then φ is *prolongable* on a symbol a if $\varphi(a) = ax$ for some string $x \neq \varepsilon$. If this is the case, then for each i, j with $0 \leq i \leq j$, it holds that $\varphi^i(a)$ is a prefix of $\varphi^j(a)$, and $\mathbf{x} = \varphi^\omega(a) = ax\varphi(x)\varphi^2(x)\dots$ is the unique infinite fixed-point of φ starting with the symbol a . An infinite string $\mathbf{w} = \varphi^\omega(a)$ that is the fixed-point of a morphism is called a *purely morphic word*, its image under a coding $\mathbf{x} = \tau(\mathbf{w})$ is called a *morphic word*, and if the morphism φ is k -uniform, then \mathbf{x} is said to be *k-automatic*.

2.3 Repetitiveness measures

A *repetitiveness measure* μ is a function that arguably captures the degree of *repetitiveness* of strings. Repetitiveness is an intuitive and elemental concept, yet is still subject to debate. In general, a repetitive string is understood as one containing many copies of the same substrings. The more repetitive is a string w , the smaller the value $\mu(w)$ should be.

If we can represent every string $w[1 : n]$ within space $O(\mu(w))$ (where the asymptotics refer to n), then we say the measure μ is *reachable*. Space is usually measured in $\Theta(\log n)$ -bit words following the conventions of the *transdichotomous RAM model of computation*. Hence, $O(\mu(w))$ space means $O(\mu(w) \log n)$ bits. We can represent any symbol in the alphabet of $w[1 : n]$ using a constant number of words as long as $|\Sigma| = O(n^d)$ for some $d \geq 0$.

A repetitiveness measure u_1 is *smaller* or *lower-bounds* another measure u_2 if $u_1(w) = O(u_2(w))$ for every $w[1 : n] \in \Sigma^*$. If, in addition, there is an infinite string family $\mathcal{F} \subseteq \Sigma^*$ where $u_1(w) = o(u_2(w))$ for every $w[1 : n] \in \mathcal{F}$, we say that u_1 is *strictly smaller* or *strictly lower-bounds* u_2 . Two repetitiveness measures u_1 and u_2 are *equivalent* if each one is smaller than the other, and *uncomparable* if none is (i.e., $u_1 = o(u_2)$ on a string family \mathcal{F}_1 and $u_2 = o(u_1)$ on another string family \mathcal{F}_2).

In the following, we explain the most relevant repetitiveness measures to be considered in the rest of the paper. For a more in-depth review, see a recent survey [17].

Grammar-based measures

There exist several compressors, and measures of repetitiveness associated with their size, that build on context-free grammars.

A *straight-line program* (SLP) is a deterministic context-free grammar G in Chomsky Normal Form whose language is a singleton $\{w\}$. We denote the string generated by the SLP as $\text{exp}(G) = w$, and extend this notation to the unique strings generated by the non-terminals of the grammar. The measure g is defined as the size of the smallest SLP G generating w . Finding the smallest SLP is an NP-complete problem [3], although there exist algorithms providing log-approximations [7, 24].

A measure based on context-free grammars that strictly lower-bounds g is g_{rl} , the size of the smallest *run-length* SLP (RLSLP) generating w [20]. RLSLPs allow constant-size rules of the form $A \rightarrow B^n$ for $n > 1$, which can make a noticeable difference in some string families like $\{\mathbf{a}^n \mid n \geq 0\}$, where $g = \Theta(\log n)$, but $g_{rl} = O(1)$.

A *collage system* is an RLSLP that, in addition, supports rules of the form $A \rightarrow B[i : j]$ for some $i, j \in [1, |\exp(B)|]$. These mean that $\exp(A) = \exp(B)[i : j]$. The size c of the smallest *collage system* [10] strictly lower-bounds g_{rl} .

Parsing-based measures

A *parsing* of size k produces a factorization of a string w into non-empty *phrases*, i.e., $w = w_1 w_2 \dots w_k$ where $w_i \in \Sigma^+$ for $i \in [1, k]$. Several compressors work by parsing w in a way that storing summary information about the phrases enables recovering w .

The *Lempel-Ziv* (LZ) parsing processes a string greedily from left to right, always forming the longest phrase that has a copy (called a source) starting inside some previous phrase or forming an *explicit* phrase of length 1 otherwise [14]. The source can overlap the new phrase. The LZ-no parsing, instead, does not allow the source overlap the new phrase. The LZ-end parsing [13] requires, in addition, that the source ends at a previous phrase boundary. All of these parsings can be constructed in linear time, and their number of phrases are denoted by z , z_{no} , and z_e , respectively. While z and z_{no} are optimal among the parsings satisfying their respective conditions, this is not always the case for z_e . The optimal factorization where each phrase w_{i+1} appears as a suffix of $w_1 \dots w_j$ for some $j \leq i$ is denoted by z_{end} . Because of the optimality of z , z_{no} , and z_{end} , it holds that $z \leq z_{no} \leq z_{end} \leq z_e$ for every string.

A *bidirectional macro scheme* (BMS) [26] is any parsing where each phrase of length greater than 1 has a copy starting at a different position in such a way that the original string can be recovered following these pointers (assuming that the phrases of length 1 store their symbol explicitly). The measure $b(w)$ is defined as the size of the smallest BMS for w . It strictly lower-bounds all the other reachable repetitiveness measures [18], except for the ones we focus on in this paper, ℓ and ν [19]. On the other hand, b is NP-hard to compute [5].

Another interesting parsing-based measure is the size of the greedy *lexicographic parsing* of w , denoted as $v(w)$ [18]. This parsing processes w from left to right, taking as the next phrase the longest common prefix between the unprocessed part of the string and a lexicographically smaller suffix of the processed part (a unique symbol $\$$, smaller than the others, is assumed to exist at the end of w). It forms an explicit phrase of length one if the longest common prefix is empty or no predecessor exists.

Burrows-Wheeler transform

The Burrows-Wheeler transform (BWT) [2] is a reversible transformation that usually makes a string more compressible. It is obtained by concatenating the last symbols of the sorted suffixes of $w\$$, where $\$$ is a symbol smaller than any symbol appearing in w . The BWT tends to produce long runs of the same symbol when a string is repetitive, and these (maximal) runs can be compressed into one symbol in the alphabet $\{(a, k) \mid a \in \Sigma, k \in [1, n]\}$ using *run-length encoding* (*rle*). A repetitiveness measure based on this idea is defined as $r(w) = |\text{rle}(\text{BWT}(w))|$. Although r is not ideal as a repetitiveness measure [6], its size can be bound in terms of z [8]. It has many practical applications representing repetitive sequences in Bioinformatics because of its indexing power [4].

String attractors

Kempa and Prezza [9] introduced the notion of *string attractor* as a unifying framework and lower bound for dictionary-based compressors.

Let w be a string of length n . A string attractor for w is a set of positions $\Gamma \subseteq [1, n]$ such that for each substring $w[i : j]$ of w , there exist integers $i', j' \in [1, n]$ and $k \in \Gamma$, such that $w[i : j] = w[i' : j']$ and $i' \leq k \leq j'$. That is, every substring of w has a copy covering a position in Γ . The measure $\gamma(w)$ is defined as the size of the smallest string attractor for w .

Computing γ is an NP-complete problem. The measure γ is a lower bound to b and is also strictly smaller than b when considering the infinite family of Thue-Morse words [1]. On the other hand, it is unknown if γ space or even $o(\gamma \log(n/\gamma))$ space is reachable.

Substring complexity

Let $F_w(k)$ be the set of distinct substrings of w of length k . The *complexity function* of w is defined as $P_w(k) = |F_w(k)|$. Kociumaka et al. [12] introduced a repetitiveness measure based on the complexity function, defined as $\delta(w) = \max\{P_w(k)/k \mid k \in [1..|w|]\}$.

The measure δ has several nice properties: it is computable in linear time, monotone, insensitive to reversals, resistant to small edits on w , can be used to construct $O(\delta \log(n/\delta))$ -space representations supporting efficient access and pattern matching queries [12, 11], and is a lower bound to almost every other theoretical or *ad-hoc* repetitiveness measure considered in the literature, including γ . On the other hand, $o(\delta \log(n/\delta))$ space is unreachable [12].

3 The measure ℓ

The class of *CPD0L-systems* is a variant of the original *L-systems*, the parallel grammars without terminals defined by Aristid Lindenmayer to model cell divisions in the growth of plants and algae [15, 16].

Formally, a CPD0L-system is a 4-tuple $L = (\Sigma, \varphi, \tau, s)$, where Σ is the *alphabet*, φ is the set of *rules* (a non-erasing endomorphism on Σ^*), τ is a coding on Σ^* , and $s \in \Sigma^+$ is the *axiom*. The system generates the sequence $(\tau(\varphi^d(s)))_{d \in \mathbb{N}}$. The “D0L” stands for *deterministic L-system with 0 interactions*. The “P” stands for *propagating*, which means that it has no ε -rules. The “C” stands for *coding*, which means that the system is extended with a coding. To define a compressor based on CPD0L-system, we extend them to 6-tuples by fixing d and using another parameter n , so we can uniquely determine a string of the sequence generated by a system and then extract a prefix from it. For simplicity, in the rest of this paper, we refer to these extended CPD0L-systems just as L-systems.

► **Definition 1 (L-systems).** *An L-system is a 6-tuple $L = (\Sigma, \varphi, \tau, s, d, n)$ where Σ is the alphabet, φ is the set of rules (an endomorphism on Σ^*), τ is a coding on Σ^* , $s \in \Sigma^+$ is the axiom, and d and n are two non-negative integers. The string generated by L is $w = \tau(\varphi^d(s))[1 : n]$.*

We now define the size of an L-system and the measure ℓ .

► **Definition 2 (Measure ℓ).** *The size of an L-system $L = (\Sigma, \varphi, \tau, s, d, n)$ is $\text{size}(L) = \text{size}(\varphi) + |s| + |\Sigma| + 2$. The measure $\ell(w)$ is defined as the size of the smallest L-system generating w .*

The size of an L-system accounts for the lengths of the right-hand sides of its rules, the length of the axiom, the coding τ , and the values d and n , so we can effectively represent the system using $O(\text{size}(L))$ space. Hence, the measure ℓ is reachable.

As a convention, we always assume that $d = n^{O(1)}$ and that $\Sigma = n^{O(1)}$. Otherwise, we could need too many words to represent the integer d or the symbols of the alphabet.

A finer-grained analysis about the number of bits needed to represent an L-system of size ℓ yields $O(\ell \log |\Sigma| + \log n)$ bits, the second term corresponding to d and n ; note that Σ contains the alphabet of w .

An important result about ℓ is that it always holds that $\ell = O(g)$ [19]. More importantly, sometimes $\ell = o(\delta)$, which implies that δ is not a lower bound for ℓ , and questions δ as a definitive measure of repetitiveness.

To understand the particularities of ℓ , we study several classes of L-systems with different restrictions and define measures based on them. We define the measure ℓ_e that restricts the morphism to be expanding. The measure ℓ_u restricts the morphism to be k -uniform for some $k > 1$. The variant ℓ_m forces the morphism of the system to be a -prolongable for some symbol a and the axiom to be $s = a$. The variant ℓ_a essentially removes the coding by forcing it to be the identity function. Finally, ℓ_p refers to the intersection of ℓ_m and ℓ_a , and ℓ_a refers to the intersection of ℓ_m and ℓ_u , which intuitively perform well in prefixes of purely morphic words and prefixes of automatic sequences, respectively.

► **Definition 3.** *An L-system $(\Sigma, \varphi, \tau, s)$ is a -prolongable if there exists a symbol a such that $s = a$ and $a \rightarrow ax$ with $x \neq \varepsilon$. An L-system is prolongable if it is a -prolongable for some symbol a .*

► **Definition 4 (ℓ -variants).** *We define the following ℓ -variants*

1. *The variant ℓ_e denotes the size of the smallest L-system generating w , satisfying that all its rules have a size at least 2.*
2. *The variant ℓ_m denotes the size of the smallest prolongable L-system generating w .*
3. *The variant ℓ_a denotes the size of the smallest L-system generating w , satisfying that τ is the identity function.*
4. *The variant ℓ_u denotes the size of the smallest L-system generating w , satisfying that all its rules have the same size, at least 2.*
5. *The variant ℓ_p denotes the size of the smallest prolongable L-system generating w , satisfying that τ is the identity function.*
6. *The variant ℓ_a denotes the size of the smallest prolongable L-system generating w , satisfying that all its rules have the same size, at least 2.*

It is known that different classes of L-systems produce different classes of languages [21]. Some of these classes also differ in the factor complexity of the sequences they can generate [22]. It is interesting to understand how these differences in terms of expressive power and factor complexity translate into compression power.

We defer the study of ℓ -variants to Section 7. We define them early, as some of our results relating ℓ to better-established measures in Sections 4 and 5 also apply for some of the ℓ -variants.

4 Breaking the δ -lower-bound for repetitiveness

It is known that the repetitiveness measure δ is a (strict) lower bound to all the other repetitiveness measures [17]. It is also known that δ is a lower bound to k -th order empirical entropy, which plays a role in several compressors [17]. This makes δ an asymptotic lower bound to the size of almost every existing compressor and compressibility measure.

Certainly, we cannot expect to find a reachable measure upper-bounded by $O(\delta)$ because δ -space is unreachable, as shown by Kociumaka et al. [12]. Still, it could be possible to

14:8 L-systems for Measuring Repetitiveness

design measures capturing repetitiveness and going below δ in some restricted but relevant scenarios. In this context, we raise the following question:

Can we find a competitive and reachable repetitiveness measure achieving space lower than δ on some restricted but relevant cases?

It is not difficult to design a trivial measure breaking the δ -lower-bound for some specific string families. We also require, however, this measure to make sense, be reachable, and be *competitive*, the latter meaning that it is at least as good as z , g , or r (i.e., the most popular reachable measures in practice) in terms of space.

The repetitiveness measure ℓ was designed with the conditions above in mind. As we already mentioned, ℓ cannot lower-bound δ because ℓ is a reachable measure.

► **Lemma 5** ([19, Theorem 4]). *There exist string families where $\ell = \Omega(\delta \log n)$.*

It was shown that ℓ is a competitive repetitiveness measure: the smallest L-system for a string is always asymptotically smaller than the smallest grammar (their proof applies to the variant ℓ_d as well). This shows that the measure ℓ is always reasonable for repetitive strings:

► **Lemma 6** ([19, Theorem 6]). *It always holds that $\ell = O(g)$.*

On the other hand, they [19] showed a string family satisfying that $\delta = \Omega(\ell \log n)$, and conjectured that this gap was the maximum possible, that is, that the lower bound $\ell = \Omega(\delta / \log n)$ holds for any string family. We now disprove this conjecture. We show a string family where δ is $\Theta(\sqrt{n})$ times bigger than the size ℓ of the smallest L-system.

► **Lemma 7.** *There exists a string family where $\delta = \Theta(\ell \sqrt{n})$.*

Proof. Consider a c-prolongable L-system $L_d = (\Sigma, \varphi, \tau, s, d, n)$, where

$$\begin{aligned}\Sigma &= \{\mathbf{a}, \mathbf{b}, \mathbf{c}\} \\ \varphi &= \{\mathbf{a} \rightarrow \mathbf{a}, \mathbf{b} \rightarrow \mathbf{ab}, \mathbf{c} \rightarrow \mathbf{cb}\} \\ \tau &= \{\mathbf{a} \rightarrow \mathbf{a}, \mathbf{b} \rightarrow \mathbf{b}, \mathbf{c} \rightarrow \mathbf{c}\} \\ s &= \mathbf{c} \\ n &= 1 + \frac{(d-1)d}{2} + d\end{aligned}$$

for any $d \geq 0$. By iterating the morphism φ we obtain the words

$$\begin{aligned}\varphi^0(\mathbf{c}) &= \mathbf{c} \\ \varphi^1(\mathbf{c}) &= \mathbf{cb} \\ \varphi^2(\mathbf{c}) &= \mathbf{cbab} \\ \varphi^3(\mathbf{c}) &= \mathbf{cbabaab} \\ \varphi^4(\mathbf{c}) &= \mathbf{cbabaabaaab} \\ \varphi^5(\mathbf{c}) &= \mathbf{cbabaabaaabaaaab}\end{aligned}$$

and so on, from which we extract as a prefix the whole word (depending on the value of d chosen). It is easy to check by induction that for each $d \geq 0$, the string generated by the system L_d is $s_d = \mathbf{c} \prod_{i=0}^{d-1} \mathbf{a}^i \mathbf{b}$ and it has length $1 + \frac{(d-1)d}{2} + d$.

It holds that ℓ is $\Theta(1)$ in this family. The system is essentially the same for every string in the family. The only changes are the integers d and n , which always fit in constant space.

$$s_3 = c \ b \ a \ b \ a \ a \ b$$

$$s_6 = c \ b \ a \ b \ a \ a \ \underline{b \ a \ a \ a \ b \ a \ a \ a \ b \ a \ a \ a \ a \ b}$$

■ **Figure 2** All the substrings of length 6 of the string s_6 of Lemma 7 starting inside some position $i \leq |s_3| = 7$ are distinct, because the runs of **a**'s considered have all different and increasing lengths, and d is big enough. The last of the substrings considered is underlined. Extending these substrings one position to the left yields $|s_3|$ different strings of length 7, so the claim holds for even and odd values of $d \geq 2$.

On the other hand, the first $|s_{\lfloor d/2 \rfloor}| = 1 + (\lfloor d/2 \rfloor - 1)(\lfloor d/2 \rfloor)/2 + \lfloor d/2 \rfloor$ substrings of length d of s_d (for $d \geq 2$) are completely determined by the **b**'s they cross, and the number of **a**'s at their extremes, so they are all distinct. An example can be seen in Figure 2.

This gives the lower bound $\delta = \Omega(d) = \Omega(\sqrt{n})$. The upper bound $O(\sqrt{n})$ holds trivially for run-length grammars as the strings considered have $\Theta(\sqrt{n})$ runs of **a**'s followed by **b**'s, so $\delta = \Theta(\sqrt{n})$. Thus $\delta = \Theta(\ell\sqrt{n})$ in this string family. ◀

This string of Lemma 7 is easy to describe yet hard to represent with copy-paste mechanisms. Intuitively, the simplicity of the sequence relies on the fact that many substrings can be described in terms of previous ones, so it is arguably highly repetitive, though not via copy-paste. The repetitiveness in this family is better captured by an L-system, instead.

5 Uncomparability of ℓ with other repetitiveness measures

As a corollary of Lemmas 5 and 7 (and also mentioned in previous work [19]), we obtain that ℓ and δ are uncomparable as repetitiveness measures.

► **Corollary 8.** *The measures ℓ and δ are uncomparable.*

Moreover, this is also true for the variant ℓ_p because, in Lemma 7, we considered a prolongable L-system with the identity function as the coding. As we prove later in Section 7, the variant ℓ_p is, in general, far from ideal for measuring repetitiveness, so the fact that δ is uncomparable to this weak variant is even more surprising.

A natural question is then to identify which other measures are also uncomparable to ℓ (and ℓ_p). We show in this section that this holds for almost any other repetitiveness measure. To do so, we first recall the string family defined by Kociumaka et al. [12], satisfying that it needs $\Omega(\log^2 n)$ bits to be represented with any method. This string family will be crucial in the following proofs.

► **Definition 9** ([12]). *The string family \mathcal{K} is formed by all the infinite strings s over $\{\mathbf{a}, \mathbf{b}\}$ constructed as follows:*

1. Let $s[1] = \mathbf{b}$.
2. For any $i \geq 2$, choose a position j_i in $[2 \cdot 4^{i-2} + 1, 4^{i-1}]$. Then, $s[j_i] = \mathbf{b}$.
3. If $j > 1$ and $j \neq j_i$ for any $i \geq 2$, then $s[j] = \mathbf{a}$.

The family \mathcal{K}_n for $n \geq 0$ is formed by all the prefixes of length n of some string in \mathcal{K} .

It is easy to see that the strings in the family \mathcal{K}_n have $\Theta(\log n)$ symbol **b**'s. Also, note that with the possible exception of the first two positions, there are no consecutive **b**'s.

Now we are ready to prove that, in general, it does not hold that $\ell = O(g_{rl})$, making L-systems uncomparable to RLSLPs.

14:10 L-systems for Measuring Repetitiveness

► **Lemma 10.** *There exists a string family where $\ell = \Omega(g_{rl} \log n / \log \log n)$.*

Proof. Consider the string family \mathcal{K}_n needing $\Omega(\log^2 n)$ bits (or $\Omega(\log n)$ space) to be represented with any method [12]. Strings in \mathcal{K}_n have $O(\log n)$ runs of **a**'s separated by **b**'s, so it is easy to see that $g_{rl} = O(\log n)$ in this family. Because of this, and because g_{rl} is a reachable measure, it holds that $g_{rl} = \Theta(\log n)$ in \mathcal{K}_n . On the other hand, the minimal L-system for a string in this family can be represented with $O(\ell \log |\Sigma| + \log n) \subseteq O(\ell \log \ell + \log n)$ bits, which must be in $\Omega(\log^2 n)$ bits because the L-system is also reachable. It follows that $\ell = \Omega(\log^2 n / \log \log n)$; otherwise,

$$\begin{aligned} \ell \log \ell &= o((\log^2 n / \log \log n) \log(\log^2 n / \log \log n)) \\ &= o(\log^2 n), \end{aligned}$$

which contradicts ℓ being reachable. Thus, $\ell = \Omega(g_{rl} \log n / \log \log n)$ in this string family. ◀

The same result holds for LZ-like parsings. Even the greedy LZ-End parsing (the largest of them) can be asymptotically smaller than ℓ in some string families.

► **Lemma 11.** *There exists a string family where $\ell = \Omega(z_e \log n / \log \log n)$.*

Proof. Take each string in \mathcal{K}_n and prepend \mathbf{a}^n to it. This new family of strings still needs $\Omega(\log^2 n)$ bits to be represented with any method because the size of the family is the same, and n just doubled. Just as in Lemma 10, it holds that $\ell = \Omega(\log^2 n / \log \log n)$ in this family. On the other hand, the LZ-End parsing needs $\Theta(\log n)$ phrases only to represent the prefix $\mathbf{a}^n \mathbf{b}$, and then for each run of **a**'s followed by **b**, its source is aligned with $\mathbf{a}^n \mathbf{b}$, so $z_e = \Theta(\log n)$. Thus, $\ell = \Omega(z_e \log n / \log \log n)$. ◀

The same result also holds for the number of equal-letter runs of the Burrows-Wheeler transform of a string.

► **Lemma 12.** *There exists a string family where $\ell = \Omega(r \log n / \log \log n)$.*

Proof. Consider the family \mathcal{K}_n again. Clearly $r = \Omega(\log n)$, because r is reachable. Because a string in this family has $O(\log n)$ **b**'s, its BWT has also $O(\log n)$ runs of **a**'s separated by **b**'s (or the unique \$). Thus, $r = \Theta(\log n)$ and $\ell = \Omega(r \log n / \log \log n)$ in this string family. ◀

We conclude that the measure ℓ is uncomparable to almost every other repetitiveness measure. We summarize these results in the following theorem.

► **Theorem 13.** *The measure ℓ is uncomparable to the repetitiveness measures $\delta, \gamma, b, v, c, g_{rl}, z, z_{no}, z_{end}, z_e$, and r .*

Proof. There exist string families where $\ell = o(\delta)$. In these families, it holds $\ell = o(\mu)$ where μ is any of the measures considered above, because δ is a lower bound to all of them. On the other hand, all the measures above are upper-bounded by at least one of z_e, g_{rl} , or r , which by Lemmas 10, 11, and 12, respectively, can be asymptotically smaller than ℓ for some string families. ◀

This shows that ℓ , although reachable and competitive as a repetitiveness measure, captures the regularities in strings in a form that is largely orthogonal to other repetitiveness measures. As the underlying regularities captured by ℓ and the other measures are apparently different, we try to combine them to obtain more powerful measures/compressors.

6 NU-systems and the measure ν

A *NU-system* [19] is a tuple $N = (V, R, \Gamma, s, d, n)$ that generates a unique string in a similar way to an L-system. The key difference is that on the right-hand side of its rules, a NU-system is permitted to have special symbols of the form $a(k)[i : j]$, whose meaning is to generate the k -th level from a , then extract the substring starting at position i and ending at position j , and finally apply the coding to the resulting substring.

The indices in a NU-system (e.g., levels, intervals) must be less or equal to n to fit in a $\Theta(\log n)$ -bits word. Also, the NU-system must not produce any loops when extracting a prefix from some level, which is decidable to detect. The size of a NU-system is defined analogously to the size of L-systems, with the extraction symbols $a(k)[i : j]$ being symbols of length 4. The measure ν is defined as the size of the smallest NU-system generating w .

It holds that $\nu = O(\ell)$ and $\nu = O(b)$ [19]. Moreover, there exist families where both asymptotic bounds are strict. We now show that NU-systems exploit the features of L-systems and macro schemes in a way that, for some string families, can reach sizes that are unreachable for both L-systems and macro schemes independently.

► **Theorem 14.** *There exists a family of strings where $\nu = o(\min(\ell, b))$.*

Proof. Let \mathcal{K}_m be the family of strings defined by Kociumaka et al. [12], needing $\Omega(\log^2 m)$ bits to be represented with any method, but over the alphabet $\{0, 1\}$. We construct a new family $\mathcal{F} = \{x \cdot \mathbf{y}[: m] \mid x \in \mathcal{K}_m\}$, where \mathbf{y} is the infinite fixed point generated by the c -prolongable L-system with the identity function as the coding, utilized in Lemma 7.

Let $n = 2m$. As shown in Lemma 10, it holds that $\ell = \Omega(\log^2 n / \log \log n)$ in this family. On the other hand, $b = \Omega(\sqrt{n})$, because $\delta = \Omega(\sqrt{n})$ on prefixes of \mathbf{y} , and the alphabets between the prefix in \mathcal{K}_m and $\mathbf{y}[: m]$ are disjoint.

Let x be a string in \mathcal{K}_m with k symbols 1. Let i_j be the number of 0's in x between the $(j-1)$ -th 1 and the j -th 1, for $j \in [2, k]$. Also, let i_1 and i_{k+1} be the number of 0's at the left and right extremes of x . We construct a NU-system $N = (V, R, \Gamma, s, d, n)$ as follows:

$$\begin{aligned} V &= \{0, 1, \mathbf{a}, \mathbf{b}, \mathbf{c}\} \\ R &= \{0 \rightarrow 00, 1 \rightarrow 1, \mathbf{a} \rightarrow \mathbf{a}, \mathbf{b} \rightarrow \mathbf{ab}, \mathbf{c} \rightarrow \mathbf{cb}\} \\ \Gamma &= \{0 \rightarrow 0, 1 \rightarrow 1, \mathbf{a} \rightarrow \mathbf{a}, \mathbf{b} \rightarrow \mathbf{b}, \mathbf{c} \rightarrow \mathbf{c}\} \\ s &= 0(m)[: i_1]10(m)[: i_2]1 \dots 0(m)[: i_k]10(m)[: i_{k+1}]c(m)[: m] \\ d &= 0 \\ n &= 2m \end{aligned}$$

By construction, this NU-system generates the string $x \cdot \mathbf{y}[: m]$ of length n , and its axiom has size $4(k+2) + k$, where $k = \Theta(\log n)$. Hence, it holds that ν is $O(\log n)$ for these strings. Thus, $\nu = o(\min(\ell, b))$ in the family \mathcal{F} we constructed. ◀

NU-systems can then be smaller representations than those produced by any other compression method exploiting repetitiveness. This shows that combining copy-paste mechanisms with iterated morphisms is an effective way of improving compression from a theoretical point of view.

On the other hand, though computable, no efficient decompression scheme has been devised for NU-systems. In turn, finding the smallest NU-system is very likely an NP-complete problem, and probably very difficult to even approximate.

7 ℓ -variants are weaker than ℓ

In this section we show that the features we include in the L-systems used in the definition of ℓ are necessary to obtain a competitive repetitiveness measure; removing any of them yields an inherent loss in compression power.

We start by showing that ℓ can be asymptotically strictly smaller than ℓ_m . That is, restricting L-systems to be prolongable yields a weaker measure.

► **Lemma 15.** *There exists a string family where $\ell = o(\ell_m)$.*

Proof. Let $\mathcal{F} = \{\mathbf{a}^{n-1}\mathbf{b} \mid n \geq 1\}$. Clearly, ℓ is constant in this string family: the L-system $L_n = (V, \varphi, \tau, s, d, n)$ where $V = \{\mathbf{a}, \mathbf{b}\}$, $\varphi = \{\mathbf{a} \rightarrow \mathbf{a}, \mathbf{b} \rightarrow \mathbf{ab}\}$, $\tau = \{\mathbf{a} \rightarrow \mathbf{a}, \mathbf{b} \rightarrow \mathbf{b}\}$, $s = \mathbf{b}$, and $d = n - 1$ produces each string in \mathcal{F} by changing only the value of n and d accordingly. Note that these L-systems are not prolongable on the axiom.

For the sake of contradiction, suppose that $\ell_m = O(1)$ in \mathcal{F} . Let $L_n = (\Sigma_n, \varphi_n, \tau_n, c, d_n, n)$ be the smallest c -prolongable system generating $\mathbf{a}^{n-1}\mathbf{b}$. Because $\ell_m = O(1)$, there exists a constant C satisfying that $|\Sigma_n| < C$ and $\text{width}(\varphi_n) < C$ for every n . Observe that it is only necessary to have one symbol $c' \in \Sigma_n$ with $\tau_n(c') = \mathbf{b}$ because there is only one \mathbf{b} in $\mathbf{a}^{n-1}\mathbf{b}$, so w.l.o.g. assume that $\mathbf{b} \in \Sigma_n$ and $\tau_n(\mathbf{b}) = \mathbf{b}$. As the system is c -prolongable, each level is a prefix of the next one. This implies that the morphism should be iterated until \mathbf{b} appears for the first time, and then we can safely extract the prefix. This must happen in the first C iterations of the morphism; otherwise, \mathbf{b} is not reachable from C (i.e., if an iteration does not yield a new symbol, then no new symbols will appear since then, and there are less than C symbols). But in the first C iterations, we cannot produce a string longer than the constant C^C . For sufficiently large n , this implies that the symbol \mathbf{b} , if it is reachable, will appear for the first time before the n -th position, which is a contradiction. ◀

Because we used the identity coding in the proof above, we can obtain the following corollary.

► **Corollary 16.** *There exists a string family where $\ell_d = o(\ell_m)$.*

We can prove a similar result for uniform morphisms.

► **Proposition 17.** *There exists a string family where $\ell_u = o(\ell_m)$.*

Proof. It is not difficult to see that ℓ_u is constant in the family $\{\mathbf{a}^{2^k}\mathbf{b} \mid k \geq 0\}$: consider the axiom $s = \mathbf{ab}$ and the rules $\mathbf{a} \rightarrow \mathbf{aa}$, $\mathbf{b} \rightarrow \mathbf{bb}$, the level $d = k$ and the prefix length $n = 2^k + 1$. A similar argument to the one of Lemma 15 yields that $\ell_u = o(\ell_m)$ for this other string family. ◀

Further, we can find a concrete asymptotic gap between ℓ and ℓ_m in the string family of the proof of the previous lemma.

► **Lemma 18.** *There exists a string family where $\ell_m = \Omega(\ell \log n / \log \log n)$.*

Proof. Let $\mathcal{F} = \{\mathbf{a}^{n-1}\mathbf{b} \mid n \geq 1\}$. Recall that $\ell = O(1)$ in this family. Let $k = |\Sigma|$ and $t = \text{width}(\varphi)$ obtained from the morphism of the smallest c -prolongable system generating $\mathbf{a}^{n-1}\mathbf{b}$ (we assume again that the only symbol mapped to \mathbf{b} by the coding is \mathbf{b}). In the first k iterations, \mathbf{b} must appear (as in the previous proof) and cannot be deleted in the following levels, so it cannot appear before position n . Hence, $t^k \geq n$, which implies $k \geq \log_t n$. By definition, $\ell_m \geq k \geq \log_t n$ and $\ell_m \geq t$, so $\ell_m \geq \max(t, \log_t n)$. The solution to the equation $t = \log_t n$ is the smallest value that $\max(t, \log_t n)$ can take for $t \in [2..n]$.

This value is $\Omega(\log n/W(\log n))$ where $W(x)$ is the Lambert W function, and it holds that $W(\log n) = \Theta(\log \log n)$. Therefore, $\ell_m = \Omega(\ell \log n / \log \log n)$ in this string family. ◀

As a corollary, we obtain the following result.

► **Corollary 19.** *There exists a string family where $\ell_m = \Omega(\ell_d \log n / \log \log n)$.*

We now show that if we remove the coding from prolongable L-systems, which corresponds to the variant ℓ_p , we end with a much worse measure. We change the usual alphabet for clarity of presentation.

► **Lemma 20.** *There exists a string family where $\ell_p = \Omega(\ell_m \sqrt{n})$.*

Proof. We prove that $\ell_p = \Theta(n)$ whereas $\ell_m = O(\sqrt{n})$ on $\mathcal{F} = \{0^{n-1}1 \mid n \geq 2\}$. Any prolongable morphism with an identity coding generating $0^{n-1}1$ must have the rule $0 \rightarrow 0^{n-1}1$, which implies $\ell_p = \Theta(n)$. The reason is that if the system is prolongable, but it has no coding, then the axiom must be 0, and in the prolongable rule $0 \rightarrow 0w$, if $|\varphi(0)| \leq n$, then the non-empty string w could only contain 0's and 1's, otherwise undesired symbols would appear in the final string because the starting level is a prefix of the final level. If w does not contain 1's, then 1 is unreachable from 0. If w contains a 1, then the first of them should be at position n .

On the other hand, we can construct an \mathbf{a} -prolongable morphism, with $\tau(1) = 1$ and $\tau(a) = 0$ for every other symbol $a \neq 1$ as follows: Let $n - 1 = k\lfloor\sqrt{n-1}\rfloor + j$ with $\lfloor\sqrt{n-1}\rfloor > 3, k > 1, 0 \leq j < \lfloor\sqrt{n-1}\rfloor$ (k and j integers). We can assume n is sufficiently big so the requirements are satisfied. Then, define the following rules

$$\begin{aligned} \mathbf{a} &\rightarrow \mathbf{ab} \\ \mathbf{b} &\rightarrow \mathbf{c}^{k-1}\mathbf{d} \\ \mathbf{c} &\rightarrow \mathbf{0}^{\lfloor\sqrt{n-1}\rfloor-1} \\ \mathbf{d} &\rightarrow \mathbf{0}^{\lfloor\sqrt{n-1}\rfloor-3+j}\mathbf{1}. \end{aligned}$$

The first four levels are

$$\begin{aligned} \varphi^0(\mathbf{a}) &= \mathbf{a} \\ \varphi^1(\mathbf{a}) &= \mathbf{ab} \\ \varphi^2(\mathbf{a}) &= \mathbf{abc}^{k-1}\mathbf{d} \\ \varphi^3(\mathbf{a}) &= \mathbf{abc}^{k-1}\mathbf{d0}^{(\lfloor\sqrt{n-1}\rfloor-1)(k-1)}\mathbf{0}^{\lfloor\sqrt{n-1}\rfloor-3+j}\mathbf{1}, \end{aligned}$$

and it holds that

$$|\varphi^3(\mathbf{a})| = 3 + (k-1) + (\lfloor\sqrt{n-1}\rfloor - 1)(k-1) + (\lfloor\sqrt{n-1}\rfloor - 3 + j) + 1 = n.$$

Hence, $\tau(\varphi^3(\mathbf{a})) = 0^{n-1}1$. The system $L = (\{\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{0}, \mathbf{1}\}, \varphi, \tau, \mathbf{a}, 3, n)$ generates $0^{n-1}1$ as required for n bigger than some constant. The size of the system is clearly $O(\sqrt{n})$. Thus, the claim holds. ◀

By using the same family above, the following corollary holds.

► **Corollary 21.** *There exists a string family where $\ell_p = \Omega(\ell_d n)$.*

It is surprising that this weak measure ℓ_p can be much smaller than δ for some string families. This can be deduced from Lemma 7. On the other hand, it does not hold that $\ell_p = O(g)$ for any string family, because $g = \Theta(\log n)$ on $\{0^{n-1}1 \mid n \geq 1\}$.

► **Corollary 22.** *The variant ℓ_p is uncomparable to the measures δ and g .*

If we restrict L-systems to be expanding, that is, with all its rules having a length of at least 2, we also end with a weaker measure. This shows that, in general, it is not possible to transform L-systems into expanding ones without incurring an increase in size.

► **Lemma 23.** *There exists a string family where $\ell = o(\ell_e)$.*

Proof. Let $\mathcal{F} = \{\mathbf{a}^k \mathbf{b} \mathbf{a}^{2^k} \mid k \geq 0\}$. Clearly ℓ is constant in \mathcal{F} : the L-system $(\{\mathbf{a}, \mathbf{b}, \mathbf{c}\}, \{\mathbf{a} \rightarrow \mathbf{a}\mathbf{a}, \mathbf{b} \rightarrow \mathbf{c}\mathbf{b}, \mathbf{c} \rightarrow \mathbf{c}\}, \{\mathbf{a} \rightarrow \mathbf{a}, \mathbf{b} \rightarrow \mathbf{b}, \mathbf{c} \rightarrow \mathbf{a}\}, \mathbf{b}\mathbf{a}, d = k, n = 2^k + k + 1)$ produces $\mathbf{a}^k \mathbf{b} \mathbf{a}^{2^k}$ and stays constant-size as k (and d and n) grows.

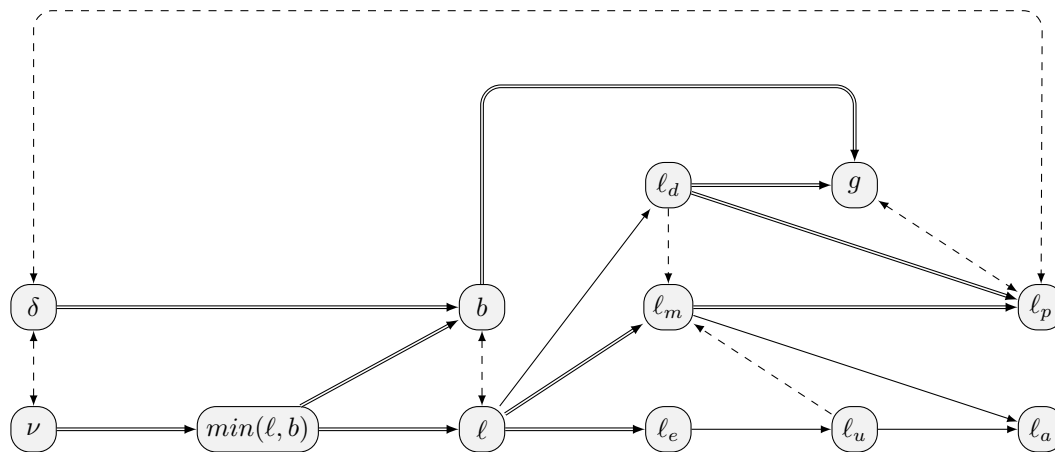
Suppose that ℓ_e is also constant in \mathcal{F} . Then there is a constant C such that the minimal expanding L-systems generating the strings in this family have at most C rules, each one of length at most C . Without loss of generality, assume that for each of these systems, the only symbol mapped to \mathbf{b} by the coding is \mathbf{b} . Also, assume that the axiom is a single symbol a_0 . Note that because the systems are expanding with rules of size at most C , their level must be $d \geq \log_C 2^k = \frac{k}{\log_2 C}$.

Let a_0, a_1, \dots, a_d be the sequence of the first symbols of $\varphi^i(a_0)$ for $i \leq d$. By the pigeonhole principle, for sufficiently big values of k (and consequently big values of d), this sequence has a period of length q starting from a_p , with $p + q \leq C \leq d$. Then there exist indexes t and j such that $t = d - jq$ and $p \leq t < p + q$. By the q -periodicity of the sequence starting at a_t , it is clear that $\varphi^q(a_t) = a_t w$ for some $w \neq \varepsilon$ (because the morphism is expanding), so φ^q is prolongable on a_t . This implies that $\varphi^{iq}(a_t)$ is a prefix of $\varphi^{jq}(a_t)$ for $i \leq j$. As before, if \mathbf{b} is reachable from a_t via φ^q , that must happen in the first C iterations, so $\varphi^{Cq}(a_t)$ contains a \mathbf{b} , and so does $\varphi^{jq}(a_t)$, which is a prefix of $\varphi^d(a_0)$. This implies that $\varphi^d(a_0)$ contains a \mathbf{b} before position C^{Cq} , which is bounded by C^{C^2} , a contradiction for sufficiently long strings in the family. So it has to be that \mathbf{b} is not reachable via φ^q from a_t , but this is also a contradiction for sufficiently long strings because $\varphi^{jq}(a_t)$ is a prefix of $\varphi^d(a_0)$ of length at least $2^{d-t} = \omega(k)$, yielding too many symbols not mapped to \mathbf{b} before the first \mathbf{b} at level d . Thus, ℓ_e cannot be $O(1)$ in \mathcal{F} . ◀

We summarize the results of this section in Figure 3. Overall, we have shown that imposing restrictions on the length of the rules of an L-system or forcing them to be prolongable wildly impacts their compression power. We have not yet found an example where ℓ_d could be asymptotically smaller than ℓ , which would prove that the coding contributes to the measure ℓ in a fundamental way (the purpose of the coding is to make ℓ constant in the case of prefixes of general morphic words, but it is unknown if it is really needed). We conjecture that such a family exists and the coding is necessary.

8 Conclusions and open questions

The measure ℓ is arguably a strong reachable repetitiveness measure, which can break the limits of δ (a measure considered a stable lower bound for repetitiveness) by a wide margin (a factor of \sqrt{n}). On the other hand, however, ℓ can be asymptotically weaker than the space reached by several compressors based on run-length context-free grammars, many Lempel-Ziv variants, and the Burrows-Wheeler transform. Only the size of context-free grammars is an upper bound to ℓ . This suggests that the self-similarity exploited by L-systems is mostly independent of the source of repetitiveness exploited by other compressors and measures, which build on copy-paste mechanisms. We also show that several attempts to simplify or restrict L-systems lead to weaker measures.



■ **Figure 3** Asymptotic relations between ℓ -variants and other relevant measures. A black arrow from v_1 to v_2 means that it always holds that $v_1 = O(v_2)$. A double black arrow from v_1 to v_2 means that it also exists a string family where $v_1 = o(v_2)$. A dashed arrow from v_1 to v_2 means that there exists a family where $v_1 = o(v_2)$.

A relevant question about L-systems is whether they can be useful for building compressed sequence representations that support direct access. More formally, can we build an $O(\ell)$ -space representation of a string $w[1 : n]$ providing random access to any position of the string in $O(\text{polylog } n)$ time? The closest result (as far as we know) is an algorithm designed by Shallit and Swart [25], which computes $\varphi^d(a)[i]$ in time bounded by a polynomial in $|\Sigma|, \text{width}(\varphi), \log d$ and $\log i$. It uses more space and takes more time than our aim. The main bottleneck is having to store the incidence matrix of the morphism and compute its powers. As suggested by Shallit and Swart, this could be solved by finding closed forms for the growth functions (recurrences) of each symbol. If this approach were taken, these formulas should be easily described within $O(\ell)$ space.

In terms of improving compression, on the other hand, the recent measure ν [19] aims to unify the repetitiveness induced by self-similarity and by explicit copies. This measure is the smallest size of a NU-system, a natural way to combine L-systems (with minimum size ℓ) with macro schemes (with minimum size $b \geq \delta$). In line with our finding that ℓ and δ are mostly orthogonal, we prove in this paper that ν is strictly more powerful than both ℓ and b , which makes ν the unique smallest reachable measure of repetitiveness to date.

There are several open questions related to NU-systems and ν . For example, does it hold that $\nu = \Omega(\ell \log \log n / \log n)$, or $\nu = \Omega(\delta / \sqrt{n})$, for every string family? Is $\nu = O(\gamma)$, or at least $o(\gamma \log(n/\gamma))$, for every string family? (recall that γ and $o(\gamma \log(n/\gamma))$ space is unknown to be reachable [9]). And towards having a practical compressor based on ν , can we decompress a NU-system efficiently?

In a more general perspective, this paper pushes a little further the discussion of what we understand by a repetitive string. Intuitively, repetitiveness is about copies, and macro schemes capture those copies pretty well, but there are other aspects in a text that could be repeated besides explicit copies, such as general patterns and the relative ordering of symbols. Macro schemes capture explicit copies, L-systems capture self-similarity, and NU-systems capture both. What other regularities could we exploit when compressing strings, keeping the representation (more or less) simple and the associated repetitiveness measure (hopefully efficiently) computable?

References

- 1 H. Bannai, M. Funakoshi, T. I, D. Köppl, T. Mieno, and T. Nishimoto. A separation of γ and b via Thue–Morse words. In *Proc. 28th International Symposium on String Processing and Information Retrieval (SPIRE)*, volume 12944 of *Lecture Notes in Computer Science (LNCS)*, pages 167–178, 2021.
- 2 M. Burrows and D. Wheeler. A block sorting lossless data compression algorithm. Technical Report 124, Digital Equipment Corporation, 1994.
- 3 M. Charikar, E. Lehman, Ding Liu, R. Panigrahy, M. Prabhakaran, A. Sahai, and A. Shelat. The smallest grammar problem. *IEEE Transactions on Information Theory*, 51(7):2554–2576, 2005.
- 4 T. Gagie, G. Navarro, and N. Prezza. Optimal-time text indexing in BWT-runs bounded space. In *Proc. 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1459–1477, 2018.
- 5 J. K. Gallant. *String Compression Algorithms*. PhD thesis, Princeton University, 1982.
- 6 S. Giuliani, S. Inenaga, Z. Lipták, N. Prezza, M. Sciortino, and A. Toffanello. Novel results on the number of runs of the Burrows-Wheeler-Transform. In *Proc. Theory and Practice of Computer Science (SOFSEM)*, pages 249–262, 2021.
- 7 A. Jež. Approximation of grammar-based compression via recompression. *Theoretical Computer Science*, 592:115–134, 2015.
- 8 D. Kempa and T. Kociumaka. Resolution of the Burrows-Wheeler Transform conjecture. *Communications of the ACM*, 65(6):91–98, 2022.
- 9 D. Kempa and N. Prezza. At the roots of dictionary compression: String attractors. In *Proc. 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, page 827–840, 2018.
- 10 T. Kida, T. Matsumoto, Y. Shibata, M. Takeda, A. Shinohara, and S. Arikawa. Collage system: a unifying framework for compressed pattern matching. *Theoretical Computer Science*, 298(1):253–272, 2003.
- 11 T. Kociumaka, G. Navarro, and F. Olivares. Near-optimal search time in δ -optimal space. In *Proc. 15th Latin American Symposium on Theoretical Informatics (LATIN)*, volume 13568 of *Lecture Notes in Computer Science (LNCS)*, pages 88–103, 2022.
- 12 T. Kociumaka, G. Navarro, and N. Prezza. Towards a definitive compressibility measure for repetitive sequences. *IEEE Transactions on Information Theory*, 69(4):2074–2092, 2023.
- 13 S. Krefl and G. Navarro. LZ77-like compression with fast random access. In *2010 Data Compression Conference (DCC)*, pages 239–248, 2010.
- 14 A. Lempel and J. Ziv. On the complexity of finite sequences. *IEEE Transactions on Information Theory*, 22(1):75–81, 1976.
- 15 A. Lindenmayer. Mathematical models for cellular interactions in development I. Filaments with one-sided inputs. *Journal of Theoretical Biology*, 18(3):280–299, 1968.
- 16 A. Lindenmayer. Mathematical models for cellular interactions in development II. Simple and branching filaments with two-sided inputs. *Journal of Theoretical Biology*, 18(3):300–315, 1968.
- 17 G. Navarro. Indexing highly repetitive string collections, part I: Repetitiveness measures. *ACM Computing Surveys*, 54(2):article 29, 2021.
- 18 G. Navarro, C. Ochoa, and N. Prezza. On the approximation ratio of ordered parsings. *IEEE Transactions on Information Theory*, 67(2):1008–1026, 2021.
- 19 G. Navarro and C. Urbina. On stricter reachable repetitiveness measures. In *Proc. 28th International Symposium on String Processing and Information Retrieval (SPIRE)*, volume 12944 of *Lecture Notes in Computer Science (LNCS)*, pages 193–206, 2021.
- 20 T. Nishimoto, T. I, S. Inenaga, H. Bannai, and M. Takeda. Fully dynamic data structure for LCE queries in compressed space. In *41st International Symposium on Mathematical Foundations of Computer Science (MFCS)*, volume 58 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 72:1–72:15, 2016.

- 21 J.-J. Pansiot. On various classes of infinite words obtained by iterated mappings. In *Automata on Infinite Words*, volume 192 of *Lecture Notes in Computer Science*, pages 188–197, 1984.
- 22 J.-J. Pansiot. Subword complexities and iteration. *Bulletin of the European Association for Theoretical Computer Science*, 26:55–62, 1985.
- 23 M Przeworski, RR Hudson, and A Di Rienzo. Adjusting the focus on human variation. *Trends in Genetics*, 16(7):296–302, 2000.
- 24 W. Rytter. Application of Lempel–Ziv factorization to the approximation of grammar-based compression. *Theoretical Computer Science*, 302(1):211–222, 2003.
- 25 J. Shallit and D. Swart. An efficient algorithm for computing the i th letter of $\varphi^n(a)$. In *Proc. 10th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 768–775, 1999.
- 26 J. A. Storer and T. G. Szymanski. Data compression via textual substitution. *Journal of the ACM*, 29(4):928–951, 1982.