On Stricter Reachable Repetitiveness Measures ⋆

Gonzalo Navarro and Cristian Urbina

CeBiB — Center for Biotechnology and Bioengineering
Departament of Computer Science, University of Chile

Abstract. The size b of the smallest bidirectional macro scheme, which is arguably the most general copy-paste scheme to generate a given sequence, is considered to be the strictest reachable measure of repetitiveness. It is strictly lower-bounded by measures like γ and δ, which are known or believed to be unreachable and to capture the entropy of repetitiveness. In this paper we study another sequence generation mechanism, namely compositions of a morphism. We show that these form another plausible mechanism to characterize repetitive sequences and define NU-systems, which combine such a mechanism with macro schemes. We show that the size $\nu \leq b$ of the smallest NU-system is reachable and can be $o(\delta)$ for some string families, thereby implying that the limit of compressibility of repetitive sequences can be even smaller than previously thought. We also derive several other results characterizing ν.

Keywords: Repetitiveness measures · Data compression · Combinatorics on words

1 Introduction

The study of repetitiveness measures, and of suitable measures of compressibility of repetitive sequences, has recently attracted interest thanks to the surge of repetitive text collections in areas like Bioinformatics, and versioned software and document collections. A recent survey [15] identifies a number of those measures, separating those that are *reachable* (i.e., any sequence can be represented within that space) from those that are not, which are still useful as lower bounds.

Reachable measures are, for example, the size g of the smallest context-free grammar that generates the sequence [8], the size c of the smallest *collage system* that generates the sequence [7] (which generalizes grammars), the number z of phrases of the Lempel-Ziv parse of the sequence [10], or the number b of phrases of a *bidirectional macro scheme* that represents the sequence [19]. Such a macro scheme cuts the sequence into phrases so that each phrase either is an explicit symbol or it can be copied from elsewhere in the sequence, in a way that no cyclic dependencies are introduced. As such, macro schemes are the ultimate measure of what can be obtained by "copy-paste" mechanisms, which characterize repetitive sequences well.

⋆ Funded in part by Basal Funds FB0001, Fondecyt Grant 1-200038, and a Conicyt Doctoral Scholarship, ANID, Chile.

Other measures are designed as lower bounds on the compressibility of repetitive sequences: γ is the size of the smallest *string attractor* for the sequence [6] and δ is a measure derived from the string complexity [17,3].

In asymptotic terms, it holds $\delta \leq \gamma \leq b \leq c \leq z \leq g$ and, except for $c \leq z$, there are string families where each measure is asymptotically smaller than the next. The recent result by Bannai et al. [2], showing that there exists a string family where $\gamma = o(b)$, establishes a clear separation between unreachable lower bounds (δ,γ) and reachable measures (b and the larger ones).

Concretely, Bannai et al. show that $b = \Theta(\log n)$ and $\gamma = O(1)$ for the Thue-Morse family, defined as $t_0 = 0$ and $t_{k+1} = t_k\overline{t_k}$, where $\overline{t_k}$ is t_k with 0s converted to 1s and vice versa. This family is a well-known example of the fixed point of a *morphism* ϕ, defined in this case by the rules $0 \rightarrow 01$ and $1 \rightarrow 10$. Then, t_k is simply $\phi^k(0)$. This representation of the words in the family is of size $O(1)$, and each word can be easily produced in optimal time by iterating the morphism.

Iterating a small morphism is arguably a relevant mechanism to define repetitive sequences. Intuitively, any short repetition $\alpha[1,k]$ that arises along the generation of a long string turns into a longer repetition $\phi^t(\alpha[1]) \cdots \phi^t(\alpha[k])$ in the final string, t steps later. More formally, if a morphism is k-uniform (i.e., all its rules are of fixed length k), then the resulting sequence is so-called k-*automatic* [1] and its prefixes have an attractor of size $\gamma = O(\log n)$ [18]. That is, many small morphisms lead to sequences with low measures of repetitiveness. Further, in the Thue-Morse family, morphisms lead to a reachable measure of repetitiveness that is $o(b)$, below what can be achieved with copy-paste mechanisms.

In this paper we further study this formalism. First, we define *macro systems*, a grammar-like extension that we prove equivalent to bidirectional macro schemes. We then study *deterministic Lindenmayer systems* [11,12], a grammar-like mechanism generating infinite strings via iterated morphisms; they are stopped at some level to produce a finite string. We combine both systems into what we call *NU-systems*. The size ν ("nu") of the smallest NU-system is always reachable and $O(b)$. Further, we show that there are string families where $\nu = o(\delta)$, thereby showing that δ is not anymore a lower bound for the compressibility of repetitive sequences if we include other plausible mechanisms to represent them. We present several other results that help characterize the new measure ν.

2 Basic Concepts

2.1 Terminology

Let Σ be a set of symbols, called the *alphabet*. A string w of length $|w| = n$ (also denoted as $w[1,n]$ when needed) is a concatenation of n symbols from Σ; in particular the string of length 0 is denoted by ε. The set of k-length concatenations of symbols from Σ is denoted Σ^k, and the set Σ^* of strings over Σ is defined as $\bigcup_{k\geq 0} \Sigma^k$; we also define $\Sigma^+ = \bigcup_{k\geq 1} \Sigma^k$. We juxtapose strings ($xy$) or combine them with the dot operator ($x \cdot y$) to denote their concatenation. A string x is a *prefix* of w if $w = xz$, a *suffix* of w if $w = yx$, and a *substring* of w if $w = yxz$, for some $y, z \in \Sigma^*$. Let $w[1, n]$ denote an n-length string. Then $w[i]$ is the i-th symbol of w, and $w[i, j]$ the substring $w[i]w[i+1] \cdots w[j]$ if $1 \le i \le j \le n$, and ε if $j < i$.

2.2 Parsing based schemes

Probably the most popular measure of repetitiveness is the number z of *phrases* in the so-called *Lempel-Ziv parse* of a word $w[1, n]$ [10]. In such a parse, w is partitioned optimally into phrases $w = x_1 \cdots x_z$, so that every x_k is either of length 1 or it appears starting to the left in w (so the phrase x_k is copied from some *source* at its left). This parsing can be computed in $O(n)$ time.

Storer and Szymanski [19] introduced *bidirectional macro schemes*, which allow sources appear to the left or to the right of their phrases, as long as circular dependencies are avoided. We follow the definition by Bannai et al. [2].

Let $w[1, n]$ be a string. A bidirectional macro scheme of size k for w is a sequence $B = (x_1, s_1), \ldots, (x_k, s_k)$ satisfying $w = x_1 \cdots x_k$ and $x_i = w[s_i, s_i + |x_i| - 1]$ if $|x_i| > 1$, and $s_i = \perp$ if $|x_i| = 1$. We denote the starting position of x_i in w by $p_i = 1 + \sum_{j=1}^{i-1} |x_j|$. The function $f : [1, n] \to [1, n] \cup \{\perp\}$,

$$f(i) = \begin{cases} \perp & : \text{ if } i = p_j, s_j = \perp \text{ for some } j \\ s_j + i - p_j : & \text{ if } p_j \le i < p_{j+1} \text{ for some } j \text{ and } s_j \ne \perp \end{cases}$$

is induced by the macro scheme. For B to be a valid bidirectional macro scheme it must hold that, for each i, there exists some r satisfying $f^r(i) = \perp$. Therefore, it suffices with the values $|x_i|$ and s_i, plus x_i where $s_i = \perp$, to recover w.

We call $b \le z$ the number k of elements in the smallest bidirectional macro scheme generating a given string $w[1, n]$. There are string families where $b = o(z)$ [16]. While z is computed in linear time, computing b is NP-hard.

2.3 Grammars and generalizations

The size g of the smallest context-free grammar generating (only) a word $w[1, n]$ [8] is a relevant measure of repetitiveness. Such a grammar has exactly one rule per nonterminal, and those can be sorted so that the right-hand sides mention only terminals and previously listed nonterminals. The size of the grammar is the sum of the lengths of the right-hand sides of the rules. The *expansion* of a nonterminal is the string of terminals it generates; the word defined by the grammar is the expansion of its last listed nonterminal.

More formally, a grammar over the alphabet of terminals Σ is a sequence of nonterminals $X_1, \ldots, X_r$, with a rule $X_k \to A_{k,1} \cdots A_{k,\ell_k}$, each $A_{k,r}$ being a terminal or some nonterminal in $X_1, \ldots, X_{k-1}$. The expansion of a terminal a is $exp(a) = a$, and that of a nonterminal X_k is $exp(X_k) = exp(A_{k,1}) \cdots exp(A_{k,\ell_k})$. The grammar represents the string $w = exp(X_r)$, and its size is $\sum_{k=1}^{r} \ell_k$.

Composition systems were introduced by Gasieniec et al. [5]. Those add the ability to reference any prefix or suffix of the expansion of a previous nonterminal (and, thus, substrings as prefixes of suffixes). Let us use the more general form, allowing terms $A_{k,r} = X_j[s, t]$ where $exp(X_j[s, t]) = exp(X_j)[s, t]$.

Kida et al. [7] extended composition systems with *run-length* terms of the form $(A_{k,r})^t$, so that $exp((A_{k,r})^t) = exp(A_{k,r})^t$, the expansion of $A_{k,r}$ concatenated t times. They called this extension a *collage system*. We call $c \leq g$ the size of the smallest collage system generating a word $w[1,n]$, and it always holds $b = O(c)$[1] and $c = O(z)$. There are string families where $b = o(c)$ [16], and where $z = o(g)$. Computing g (and, probably, c too) is NP-hard.

2.4 Lower bounds

Kempa and Prezza introduced the concept of *string attractor* [6], which yields an abstract measure that lower-bounds all the previous reachable measures.

Let $w[1,n]$ be a string. A string attractor for w is a set of positions $A \subseteq [1,n]$ where for every substring $w[i,j]$ there exists a copy $w[i',j']$ (i.e., $w[i,j] = w[i',j']$) and a position $k \in A$ with $i' \leq k \leq j'$. The measure γ is defined as the cardinality of the smallest of such attractors for a given string $w[1,n]$, and it always holds that $\gamma = O(b)$. Further, a string family where $\gamma = o(b)$ exists [2].

Kociumaka et al. [3] used the *string complexity* of $w[1,n]$ to define a measure called δ. Let $S_w(k)$ be the number of distinct substrings of length k in w. Then $\delta = \max\{S_w(k)/k \,|\, 1 \leq k \leq n\}$. This measure is computed in $O(n)$ time and it always holds that $\delta = O(\gamma)$; there are string families where $\delta = o(\gamma)$ [9]. While δ is unreachable in some string families, any string can be represented in $O(\delta \log(n/\delta))$ space [9]. Measure δ has been proposed as a lower bound on the compressibility of repetitive strings, which we question in this paper.

2.5 Morphisms over strings

We explain some general concepts about morphisms acting over strings [1,14]. A *monoid* $(M, *, e)$ is a set with an associative operation $*$ and a neutral element $e \in M$ satisfying $a * e = e * a = a$ for every $a \in M$. We write ab for $a * b$ and say that M is a monoid, instead of $(M, *, e)$. A *morphism of monoids* is a function $\phi : M_1 \to M_2$, where $(M_1, *_1, e_1)$ and $(M_2, *_2, e_2)$ are monoids, $\phi(a *_1 b) = \phi(a) *_2 \phi(b)$ for every $a, b \in M_1$, and $\phi(e_1) = e_2$.

Let Σ be a set of symbols, and $\cdot$ the concatenation of strings. Then $(\Sigma^*, \cdot, \varepsilon)$ is a monoid with string concatenation, called the *free monoid*. A *morphism of free monoids* $\phi : \Sigma^* \to \Delta^*$ is defined completely just by specifying ϕ on the symbols of Σ. If $\Sigma = \Delta$, then ϕ, is called an *automorphism*, and ϕ is *iterable*. We define the n-iteration (or composition) of ϕ over s as $\phi^n(s)$.

Let $\phi : \Sigma^* \to \Delta^*$ be a morphism of free monoids. We define $depth(\phi) = |\Sigma|$, $width(\phi) = \max_{a \in \Sigma} |\phi(a)|$, and $size(\phi) = \sum_{a \in \Sigma} |\phi(a)|$. We say ϕ is *expanding* if $|\phi(a)| > 1$, *non-erasing* if $|\phi(a)| > 0$, and *k-uniform* if $|\phi(a)| = k$, for every $a \in \Sigma$. A *coding* is a 1-uniform morphism. We say ϕ is *prolongable* on $a \in \Sigma$ if $\phi(a) = as$ for a non-empty string s.

Let ϕ be an automorphism on Σ^*. Let ϕ be prolongable on a, so $\phi(a) = as$. Then, $w = as\phi(s)\phi^2(s)\cdots$ is the unique infinite *fixed point* of ϕ starting with

[1] At least if the collage system is *internal*, that is, every $exp(X_k)$ appears in w.

a, that is, $\phi(w) = w$ [14]. Words built in this fashion are called *purely morphic words*. If we apply a coding to them, we obtain *morphic words*. A morphic word obtained from a k-uniform morphism is said to be *k-automatic* [1].

3 Macro Systems

Our first contribution is the definition of *macro systems*, a generalization of composition systems we prove to be as powerful as bidirectional macro schemes. That is, the smallest macro system generating a given string w is of size $O(b)$.

Definition 1. *A* macro system *is a tuple $M = (V, \Sigma, R, S)$, where V is a finite set of symbols called the variables, Σ is a finite set of symbols disjoint from V called the terminals, R is the set of rules (exactly one per variable)*

$$R : V \to (V \cup \Sigma \cup \{A[i,j] \mid A \in V, i, j \in \mathbb{N}\})^*,$$

and $S \in V$ is the initial variable. If $R(A) = \alpha$ is the rule for A, we also write $A \to \alpha$. The symbols $A[i,j]$ are called extractions. *The rule $A \to \varepsilon$ is permitted only for $A = S$. The* size *of a macro system is the sum of the lengths of the right-hand sides of the rules, $size(M) = \sum_{A \in V} |R(A)|$.*

We now define the string generated by a macro system as the expansion of its initial symbol, $exp(S)$. Such expansions are defined as follows.

Definition 2. *Let $M = (V, \Sigma, R, S)$ be a macro system. The* expansion *of a symbol is a string over Σ^* defined inductively as follows:*

- *If $a \in \Sigma$ then $exp(a) = a$.*
- *If $S \to \varepsilon$, then $exp(S) = \varepsilon$.*
- *If $A \to B_1 \cdots B_k$ is a rule, then $exp(A) = exp(B_1) \cdots exp(B_k)$.*
- *$exp(A[i,j]) = exp(A)[i,j]$ (this second $[i,j]$ denotes substring).*

We say that the macro system is valid *if there is a single solution $w \in \Sigma^*$ for $exp(S)$. We say that the macro system* generates *the string w.*

Note that a macro system looks very similar to a composition system, however, it does not impose an order so that each symbol references only previous ones. This algorithm determines the string generated by a macro system, if any:

1. Compute $|exp(A)|$ for every nonterminal A, using the rules:
   - If $a \in \Sigma$, then $|exp(a)| = 1$.
   - If $A \to B_1 \cdots B_k$, then $|exp(A)| = |exp(B_1)| + \cdots + |exp(B_k)|$.
   - $|exp(A[i,j])| = j - i + 1$.

   This must generate a system of equations without loops (otherwise the macro system is invalid), which is then trivially solved.
2. Replace every symbol $A[i,j]$ by $A[i] \cdots A[j]$; we use $A[r]$ to denote $A[r,r]$.
3. Replace every $A[r]$, if $A \to B_1 \cdots B_k$, iterating until obtaining a terminal:
   - Let $p_i = 1 + \sum_{j=1}^{i-1} |exp(B_j)|$, for $1 \leq i \leq k+1$.

   - Let s be such that $p_s \leq r < p_{s+1}$.
   - If $B_s \in \Sigma$, replace $A[r]$ by B_s.
   - Otherwise replace $A[r]$ by $B_s[r - p_s + 1]$.
4. If the process to replace any $A[r]$ falls in a loop (i.e., we return to $A[r]$), then the system has no unique solution and thus it is invalid. Otherwise, we are left with a classical context-free grammar without extractions, and compute $w = exp(S)$ in the classical way.

Note that a rule like $A \rightarrow B\ A[1, (t-1)|exp(B)|]$ solves only for $exp(A) = exp(B)^t$, just like the run-length symbol B^t of collage systems. For example, $A \rightarrow ab$ and $S \rightarrow A\ S[1, 4]$ generates $ababab$ as follows:

$$\begin{array}{l} A\ S[1]\ S[2]\ S[3]\ S[4] \\ A\ A[1]\ A[2]\ S[1]\ S[2] \\ A\ a\ b\ A[1]\ A[2] \\ A\ a\ b\ a\ b \\ a\ b\ a\ b\ a\ b \end{array}$$

This shows that macro systems are at least as powerful as collage systems. But they can be asymptotically smaller. For example, the smallest collage system generating the Fibonacci string F_k (where $F_1 = b$, $F_2 = a$, and $F_{k+2} = F_{k+1}F_k$) is of size $\Theta(\log |F_k|)$ [16, Thm. 32]. Instead, we can mimic a bidirectional macro scheme of size 4 [16, Lem. 35] with a constant-sized macro system generating F_k: $S \rightarrow S[f_{k-2}+1, f_k - 2]\ b\ a\ S[f_{k-2}+1, 2f_{k-2}]$ if k is odd and $S \rightarrow S[f_{k-2}+1, f_k - 2]\ a\ b\ S[f_{k-2}+1, 2f_{k-2}]$ if k is even (where $f_k = |F_k|$). For example, for F_7 the system is $S \rightarrow S[6, 11]\ b\ a\ S[6, 10]$ and we extract $F_7 = exp(S)$ as follows, using that $F_7[1, 6] = F_7[6, 11]$, $F_7[7] = b$, $F_7[8] = a$, and $F_7[9, 13] = F_7[6, 10]$:

$$\begin{array}{l} S[6]\ S[7]\ S[8]\ S[9]\ S[10]\ S[11]\ b\ a\ S[6]\ S[7]\ S[8]\ S[9]\ S[10] \\ S[11]\ b\ a\ S[6]\ b\ a\ b\ a\ S[11]\ b\ a\ S[6]\ b \\ a\ b\ a\ S[11]\ b\ a\ b\ a\ a\ b\ a\ S[11]\ b \\ a\ b\ a\ a\ b\ a\ b\ a\ a\ b\ a\ a\ b \end{array}$$

In general, we can prove that a restricted class of our macro systems is equivalent to bidirectional macro schemes.

Definition 3. *A macro system $M = (V, \Sigma, R, S)$ generating w is* internal *if $exp(A)$ appears in w for every $A \in V$. We use m to denote the size of the smallest internal macro system generating w.*

Theorem 1. *It always holds that $m \leq b$.*

Proof. Let $(x_1, s_1), \ldots, (x_b, s_b)$ be the smallest bidirectional macro scheme generating $w[1, n] = x_1 \cdots x_b$. We construct a macro system $M = (\{S\}, \Sigma, R, S)$ with a single rule $S \rightarrow A_1 \cdots A_b$, where A_i is the single terminal x_i if $s_i = \perp$, and the extraction symbol $S[s_i, s_i + |x_i| - 1]$ if not.

We now show that this macro system is valid. After we execute step 2 of our algorithm, the length of the resulting string (which we call W) is already n: it has only terminals and symbols of the form $W[i] = S[r]$. Note that this implies that $f(i) = r$ in the bidirectional macro scheme. In every step, we replace each such $S[r]$ by $W[r]$. Since the macro scheme is valid, for each i there is a finite k such that $f^k(i) = \perp$, and thus $W[i]$ becomes a terminal symbol after k steps. $\square$

Theorem 2. *For every internal macro system of size m there is a bidirectional macro scheme of size $b \leq m$.*

Proof. An internal macro system $M = (V, \Sigma, R, S)$ generating $w[1, n]$ can always be transformed into one with a single rule for the initial symbol. Let $A \in V$ be such that $w[i, j] = exp(A)$. We can then replace every occurrence of A by $S[i, j]$, and every occurrence of $A[i', j']$ by $S[i' + i - 1, j' + i - 1]$, on the right-hand sides of all the rules. In particular, the rule defining S will now contain terminals and symbols of the form $S[i, j]$, and thus all the other nonterminals can be deleted.

From the resulting macro system $S \to A_1 \cdots A_{m'}$, where $m' \leq m$, we can derive a bidirectional macro scheme $(x_1, s_1), \ldots, (x_{m'}, s_{m'})$, as follows: if A_t is a terminal, then x_t is that terminal and $s_t = \perp$. Otherwise, A_t is of the form $S[i, j]$ and then $x_t = w[i, j]$ and $s_t = i$. The resulting scheme is valid, because our algorithm extracts any $S[i]$ after a finite number k of steps, which is then the k such that $f^{k+1}(i) = \perp$. $\square$

That is, bidirectional macro schemes are equivalent to internal macro systems. General macro systems can be asymptotically smaller in principle, though we have not found an example where this happens.

4 Deterministic Lindenmayer Systems

In this section we study a mechanism for generating infinite sequences called *deterministic Lindenmayer Systems* [11,12], which build on morphisms. We adapt those systems to generate finite repetitive strings. Those systems are, in essence, grammars with only nonterminals, which typically generate longer and longer strings, in a levelwise fashion. For our purposes, we will also specify at which level d to stop the generation process and the length n of the string w to generate. The generated string $w[1, n]$ is then the n-length prefix of the sequence of nonterminals obtained at level d. We adapt, in particular, the variant called CD0L-systems, though we will use the generic name *L-systems* for simplicity.

Definition 4. *An* L-system *is a tuple $L = (V, R, S, \tau, d, n)$, where V is a finite set of symbols called* variables, *$R : V \to V^+$ is the set of* rules, *$S \in V^*$ is a sequence of variables called the* axiom, *$\tau : V \to V$ is a coding, $d \in \mathbb{N}$ is the level where to stop (or depth), and $n \in \mathbb{N}$ is the length of the string to generate.*

An L-system produces levels *of strings $L_i \in V^*$, starting from $L_0 = S$ at level 0. Each level replaces every variable A from the previous level by $R(A)$, that is, $L_{i+1} = R(L_i)$ if we identify R with its homomorphic extension. The generated string is $w[1, n] = \tau(L_d[1, n]) \in V^*$, seeing τ as its homomorphic extension.*

The size *of an L-system is $|S| + \sum_{A \in V} |R(A)|$. We call ℓ the size of the smallest L-system generating a string w.*

L-systems then represent strings by iterating a non-erasing automorphism. Somewhat surprisingly, we now exhibit a string family where $\delta = \Omega(\ell \log n)$, thus L-systems are a reachable mechanism to generate strings that can be asymptotically smaller than what was considered to be a stable lower bound.

Theorem 3. *There exist string families where $\delta = \Omega(\ell \log n)$.*

Proof. Consider the L-system $L = (V, R, S, \tau, d, n)$ where $V = \{0, 1\}$, $S = 0$, $R(0) = 001$, $R(1) = 1$, $\tau(0) = 0$, $\tau(1) = 1$, and $n = 2^{d+1} - 1$. The family of strings is formed by all those generated by the systems L, where $d \in \mathbb{N}$. It is clear that all the strings in this family share the value $\ell = 5$.

The first strings of the family generated by this system (i.e., its levels L_i) are 0, 001, 0010011, 001001100100111, and so on. It is easy to see by induction that level i contains 2^i 0s and $2^i - 1$ 1s, so the string L_i is of length $2^{i+1} - 1$.

More importantly, one can see by induction that levels $i \geq 2$ start with 00 and contain all the strings of the form 01^j0 for $1 \leq j < i$. This is true for level 2. Then, in level $i + 1$ the strings 01^j0 become 0011^j001, which contains $01^{j+1}0$, and the first 00 yields 001001, containing 010.

Consider now the number of d-length distinct substrings in L_d, for $d \geq 4$. Each distinct substring 01^j0, for $\lfloor d/2 \rfloor - 1 \leq j \leq d - 2$, yields at least $d - j - 1$ distinct d-length substrings (containing 01^j0 at different offsets; no single d-length substring may contain two of those). These add up to $d^2/8 + d/4$ distinct d-length substrings, and thus $\delta = \Omega(d) = \Omega(\log n)$ on the string $w = \tau(L_d)$. □

On the other hand, L-systems are always reachable, which yields the immediate result that δ and ℓ are incomparable.

Theorem 4. *There exist string families where $\ell = \Omega(\delta \log n)$.*

Proof. Kociumaka et al. [9, Thm. 2] exhibit a string family of $2^{\Theta(\log^2 n)}$ elements with $\delta = O(1)$, so it needs $\Omega(\log^2 n) = \Omega(\delta \log^2 n)$ bits, to be represented with any method. On the other hand, an L-system of size ℓ is described with $O(\ell \log n)$ bits. Therefore $\ell = \Omega(\log n) = \Omega(\delta \log n)$ in this family. □

Those strings are formed by n as, replacing them by bs at single arbitrary positions between $2 \cdot 4^{j-2} + 1$ and 4^{j-1} for every $j \geq 2$. While such a string is easily generated by a composition system of size $\Theta(\log n)$, we could only produce L-systems of size $\Theta(\log^2 n)$ generating it. We now prove bounds between L-systems and context-free grammars.

Theorem 5. *For any L-system $L = (V, R, S, \tau, d, n)$ of size ℓ generating w, there is a context-free grammar of size $(d + 1)\ell$ generating w. If the morphism represented by R is expanding, then the grammar is of size $O(\ell \log n)$.*

Proof. Consider the derivation tree for w in L: the root children are $S = L_0$ at level 0, and if A is a node at level i, then the children of A are the elements in $R(A)$, at level $i+1$. The nodes in each level i spell out L_i.

We create a grammar $G = (V', V, R', S')$ where V' contains the initial symbol S' and, for each variable $A \in V$ of the L-system, d nonterminals $A_0, \ldots, A_{d-1}$. The terminals of the grammar are the set of L-system variables, V. Then, for each L-system rule $A \to B_1 \cdots B_k$ appearing in level $0 \leq i \leq d-2$, we add the grammar rule $A_i \to (B_1)_{i+1} \cdots (B_k)_{i+1}$. Further, for each rule $A \to B_1 \cdots B_k$ appearing in level $d-1$, we add the grammar rule $A_{d-1} \to \tau(B_1) \cdots \tau(B_k)$. Finally, if $S = B_1 \cdots B_k$ is the L-system axiom, we add the grammar rule $S' \to (B_1)_0 \cdots (B_k)_0$ for its initial symbol.

It is clear that the grammar is of size at most $(d+1)\ell$ and it generates w. If every rule is of size larger than 1, and $d > \lg n$, then the prefix $w[1,n]$ of $\tau(L_d)$ is generated from the first symbol of $L_{d-\lceil \lg n \rceil}$, which can then be made the axiom and d reduced to $\lceil \lg n \rceil$. In this case, the grammar is of size $O(\ell \log n)$. $\square$

For example, consider our L-system $0 \to 001$ and $1 \to 1$. A grammar simulating a generation of $d = 3$ levels contains the rules $S' \to 0_0$, $0_0 \to 0_1 0_1 1_1$, $0_1 \to 0_2 0_2 1_2$, $1_1 \to 1_2$, $0_2 \to 001$, and $1_2 \to 1$. Note how the grammar uses the level subindices to control the point where the L-system should stop.

On the other hand, while we believe that composition systems can be smaller than L-systems, we can prove that L-systems are not larger than grammars.

Theorem 6. *It always holds that $\ell = O(g)$.*

Proof. Consider a grammar $G = (V, \Sigma, R, S)$ of height h generating $w[1,n]$. We define the L-system $L = (V \cup \Sigma, R', R(S), \tau, h, n)$, where R' contains all the rules in R except the one for S. We also include in R the rules $a \to a$ for all $a \in \Sigma$. The coding τ is the identity function.

It is clear that this L-system produces the same derivation tree of G, reaching terminals a at some level. Those remain intact up to the last level, h, thanks to the rules $a \to a$. At this point the L-system has derived $w[1,n]$.

The size of the L-system is that of G plus $|\Sigma|$, which is of the same order because every symbol $a \in \Sigma$ appears on some right-hand side (if not, we do not need to create the rule $a \to a$ for that symbol). $\square$

The following simple result characterizes a class of morphisms generating families with constant-sized L-systems.

Theorem 7. *Let $w \in \Sigma^*$, $\phi : \Sigma^* \to \Sigma^*$ be a non-erasing automorphism, and $\tau : \Sigma \to \Sigma$. Then $\ell = O(1)$ on the family $\{\tau(\phi^d(w)) \mid d > 0\}$.*

Proof. We can easily simulate ϕ on the L-system $L = (\Sigma, R, w, \tau, d, n)$ of fixed size, with $R(a) = \phi(a)$ and $n = |\phi^d(w)|$. The system generates $\tau(\phi^d(w))$ and, as d grows, it does not change its size. $\square$

This implies that $\ell = O(1)$ on families of n-iterations of the Thue-Morse morphism, the Fibonacci morphism, images of k-uniform morphisms (i.e., morphisms generating k-automatic words [1]), and standard Sturmian morphisms [4]. More generally, ℓ is $O(1)$ on the set of prefixes of any morphic word.

5 NU-Systems

We now define a mechanism that combines both macro systems and L-systems, yielding a computable measure that is reachable and strictly better than b.

Definition 5. *A* NU-system *is a tuple $N = (V, R, S, \tau, d, n)$, which is understood in the same way as L-systems, except that we extend rules with* extractions*, that is, $R : V \to (V \cup E)^+$ and*

$$E = \{A(l)[i, j] \mid A \in V, l, i, j \in \mathbb{N}\}.$$

The symbol $A(l)[i, j]$ means to expand variable A for l levels and then extract $\tau(A_l[i, j])$ from the string A_l at level l, recursively expanding extractions if necessary. This counts as a single expansion (one level) of a rule, that is, the levels L_i in the NU-system belong to V^*. We also use $A(l) = A(l)[1, |A_l|]$ to denote the whole level l of A. The size *of the NU-system is $size(N) = |S| + \sum_{A \in V} |R(A)|$. We call ν the size of the smallest NU-system generating a string $w[1, n]$.*

Just as macro systems, a NU-system is *valid* only if it does not introduce circular dependencies. Let $maxl$ be the maximum l value across every rule $A(l)[i, j]$ in the NU-system. The following algorithm determines the string generated by the system, if any:

1. Compute $|A_l|$ for every variable A and level $0 \le l \le maxl$, using the rules:
   - $|A_0| = 1$.
   - If $l > 0$ and $A \to B_1 \cdots B_k$, then $|A_l| = |(B_1)_{l-1}| + \cdots + |(B_k)_{l-1}|$.
   - Replace $|B(l)[i, j]| = j - i + 1$ on the previous summands $|(B_r)_{l-1}|$.

   This generates a system of equations without loops, which is trivially solved.
2. Replace every symbol $A(l)[i, j]$ in R by $A(l)[i] \cdots A(l)[j]$; we use $A(l)[r]$ to denote $A(l)[r, r]$.
3. Expand the rules, starting from the axiom, level by level as in L-systems. Handle the symbols $A(l)[r]$ as follows:
   (a) Replace every $A(0)[r]$ (so $r = 1$ if the NU-system is correct) by $\tau(A)$.
   (b) Replace every $A(l)[r]$, if $l > 0$ and $A \to B_1 \cdots B_k$, as follows:
      - Let $p_i = 1 + \sum_{j=1}^{i-1} |(B_j)_{l-1}|$, for $1 \le i \le k + 1$.
      - Let s be such that $p_s \le r < p_{s+1}$.
      - Replace $A(l)[r]$ by $B_s(l - 1)[r - p_s + 1]$.

   (c) Return to (a) until the extraction symbol disappears.

Note that the symbol B_s in step 3(b) can in turn be of the form $B_s = B(l')[r']$; we must then extract $B(l')[r']$ before continuing the extraction of $B_s(l - 1)[r - p_s + 1]$. If, along the expansion, we return again to the original $A(l)[r]$, then the system has no unique solution and thus it is invalid. This is computable because the number of possible combinations $A(l)[r]$ is bounded by $|V| \cdot maxl \cdot n$.

We now show that NU-systems are at least as powerful as macro systems and L-systems.

Theorem 8. *It always holds that $\nu = O(\min(\ell, m))$.*

Proof. It always holds $\nu \leq \ell$ because L-systems are a particular case of NU-systems. With respect to m, let $M = (V, \Sigma, R, S)$ be a minimal macro system generating $w[1,n]$. Then we construct a NU-System $N = (V \cup \Sigma, R', S, \tau, d, n)$ where τ is the identity and $d = |V|$, which upper-bounds the height of the derivation tree. Each level of N will simulate the sequence of extractions that lead from each $A[r]$ to its corresponding terminal in the macro system.

For each $a \in \Sigma$ we define the rule $a \to a$ in R'. For each rule $A \to B_1 \cdots B_m$ in R, we define the rule $A \to B'_1 \cdots B'_m$ in R', where $B'_i = B_i$ if $B_i \in V \cup \Sigma$, and $B'_i = A'(d)[j,k]$ if $B_i = A'[j,k]$. It is not hard to see that the NU-System N simulates the macro system M, and its size is $O(m)$. □

For example, consider our previous macro system $A \to ab$ and $S \to A\ S[1,4]$. The corresponding NU-system would have the rules $a \to a$, $b \to b$, $A \to ab$, and $S \to A\ S(2)[1,4]$. The derivation is then generated as follows:

$$\begin{array}{lll}
L_0 = S & \longrightarrow & A\ S(2)[1]\ S(2)[2]\ S(2)[3]\ S(2)[4] \\
 & & A\ A(1)[1]\ A(1)[2]\ (S(2)[1])(1)[1]\ (S(2)[2])(1)[1] \\
 & & A\ a\ b\ (A(1)[1])(1)[1]\ (A(1)[2])(1)[1] \\
L_1 = A\ a\ b\ a\ b & \longleftarrow & A\ a\ b\ a(1)[1]\ b(1)[1] \\
L_2 = a\ b\ a\ b\ a\ b. & &
\end{array}$$

Our new measure ν is then reachable, strictly better than b and incomparable with δ. It is likely, however, that computing ν (i.e., finding the smallest NU-system generating a given string $w[1,n]$) is NP-hard.

NU-systems easily allow us concatenating and composing automorphisms.

Theorem 9. *Let $N_1 = (V_1, R_1, S_1, \tau_1, d_1, n_1)$ and $N_2 = (V_2, R_2, S_2, \tau_2, d_2, n_2)$ be NU-systems generating w_1 and w_2, respectively. Then there are NU-systems of size $O(size(N_1) + size(N_2))$ that generate $w_1 \cdot w_2$ and the* composition *of w_1 and w_2, which is the string generated by N_2 with axiom w_1, $(V_2, R_2, w_1, \tau_2, d_2, n_2)$.*

Proof. Let $V'_1 = \{a_1 \mid a \in V_1\}$ and $V'_2 = \{a_2 \mid a \in V_2\}$ be disjoint copies of V_1 and V_2, respectively, and let R'_i, and S'_i be variants that operate on V'_i instead of V_i. We build a NU-system $N = (V, R, S, \tau, 1, n_1 + n_2)$ for $w_1 \cdot w_2$, where $V = V'_1 \cup V'_2 \cup V_1 \cup V_2 \cup \{Z_1, Z_2\}$, where Z_1 and Z_2 are new symbols. Let $R = R'_1 \cup R'_2 \cup \{Z_1 \to S'_1, Z_2 \to S'_2\}$, plus the rules $a \to a$ for $a \in V_1 \cup V_2$. The axiom is then $S = Z_1(d_1) \cdot Z_2(d_2)$. Finally, the mapping on V'_i is $\tau(a_i) = \tau_i(a)$, and $\tau(a) = a$ for $a \in V_1 \cup V_2$. It is easy to see that N generates $w_1 \cdot w_2$.

To generate the composition, V_2 should contain the image of V_1 by τ_1, but still V'_1 is disjoint from V'_2. The axiom is $Z_1(d_1)$. The mapping is $\tau(a_1) = \tau_1(a)_2$ on V'_1, $\tau(a_2) = \tau_2(a)$ on V'_2, and $\tau(a) = a$ on $V_1 \cup V_2$. The depth is $d = 1 + d_2$. □

The theorem allows a family $\mathcal{F}$ to have $\nu = O(1)$, by finding a finite collection of families generated by fixed non-erasing automorphisms, and then joining them using a finite number of set unions, concatenations and morphism compositions.

6 Future work

We leave a number of open questions. We know $\nu = O(m) = O(b) = O(\delta \log(n/\delta))$, but it is unknown if $\nu = O(\gamma)$; if so, then γ would be reachable. We know $\ell = O(g)$, but it is unknown if $\ell = O(c)$; we suspect it is not, but in general we lack mechanisms to prove lower bounds on ℓ or ν. We also know $m = O(b)$, but not if it can be strictly better. We also do not know if these measures are monotone, and if they are actually NP-hard to compute (they are likely so).

We could prove that $g = O(\ell \log n)$, and thus the lower bound $\ell = \Omega(g/\log n)$, if every L-system could be made expanding, but this is also unknown. This, for example, would prove that the stretch $\ell = O(\delta/\log n)$ we found for a family of strings is the maximum possible.

7 Conclusions

Extending the study of repetitiveness measures, from parsing-based to morphism-based mechanisms, opens a wide number of possibilities for the study of repetitiveness. There is already a lot of theory behind morphisms, waiting to be exploited on the quest for a golden measure of repetitiveness.

We first generalized composition systems to macro systems, showing that a restriction of them, called internal macro systems, are equivalent to bidirectional macro schemes, the lowest reachable measure of repetitiveness considered in the literature. It is not yet known if general macro systems are more powerful.

We then showed how morphisms, and measures based on mechanisms capturing that concept called L-systems (and variations), can be strictly better than δ for some string families, thereby questioning the validity of δ as a lower bound for reachable repetitiveness measures. L-systems are never larger than context-free grammars, but probably not always as small as composition systems.

Finally, we proposed a novel mechanism of compression aimed at unifying parsing and morphisms as repetitiveness sources, called NU-systems, which builds on macro systems and L-systems. NU-systems can combine copy-paste, recurrences, and concatenations and compositions of morphisms. The size ν of the smallest NU-system generating a string is a relevant measure of repetitiveness because it is reachable, computable, always in $O(b)$ and sometimes $o(\delta)$.

A simple lower bound capturing the idea of recurrence on a string, and lower bounding ℓ, just like δ captures the idea of copy-paste and strictly lower bounds b, would be of great interest when studying morphism-based measures. For infinite strings, there exist concepts like *recurrence constant* and *appearance constant* [1], but an adaptation, or another definition, is needed for finite strings. Besides, like Lindenmayer systems, NU-systems could be used to model other repetitive structures beyond strings that appear in biology, like the growth of plants and fractals. In this sense, they can be compared with tree grammars; the relation between NU-systems and TSLPs [13], for example, deserves further study.

References

1. Allouche, J.P., Shallit, J.: Automatic Sequences: Theory, Applications, Generalizations. Cambridge University Press (2003)
2. Bannai, H., Funakoshi, M., I, T., Koeppl, D., Mieno, T., Nishimoto, T.: A separation of γ and b via thue–morse words. CoRR 2104.09985 (2021)
3. Christiansen, A.R., Ettienne, M.B., Kociumaka, T., Navarro, G., Prezza, N.: Optimal-time dictionary-compressed indexes. ACM Trans. Alg. **17**(1), art. 8 (2020)
4. de Luca, A.: Standard Sturmian morphisms. Theor. Comp. Sci. **178**(1), 205–224 (1997)
5. Gasieniec, L., Karpinski, M., Plandowski, W., Rytter, W.: Efficient algorithms for Lempel-Ziv encoding. In: Proc. SWAT. pp. 392–403 (1996)
6. Kempa, D., Prezza, N.: At the roots of dictionary compression: String attractors. In: Proc. 50th STOC. p. 827–840 (2018)
7. Kida, T., Matsumoto, T., Shibata, Y., Takeda, M., Shinohara, A., Arikawa, S.: Collage system: a unifying framework for compressed pattern matching. Theor. Comp. Sci. **298**(1), 253–272 (2003)
8. Kieffer, J.C., Yang, E.H.: Grammar-based codes: A new class of universal lossless source codes. IEEE Trans. Inf. Theory **46**(3), 737–754 (2000)
9. Kociumaka, T., Navarro, G., Prezza, N.: Towards a definitive measure of repetitiveness. In: Proc. 14th LATIN. pp. 207–219 (2020)
10. Lempel, A., Ziv, J.: On the complexity of finite sequences. IEEE Trans. Inf. Theory **22**(1), 75–81 (1976)
11. Lindenmayer, A.: Mathematical models for cellular interactions in development I. Filaments with one-sided inputs. J. Theor. Biol. **18**(3), 280–299 (1968)
12. Lindenmayer, A.: Mathematical models for cellular interactions in development II. Simple and branching filaments with two-sided inputs. J. Theor. Biol. **18**(3), 300–315 (1968)
13. Lohrey, M.: Grammar-based tree compression. In: Proc. DLT. pp. 46–57 (2015)
14. Lothaire, M.: Algebraic Combinatorics on Words. Cambridge University Press (2002)
15. Navarro, G.: Indexing highly repetitive string collections, part I: Repetitiveness measures. ACM Comp. Surv. **54**(2), article 29 (2021)
16. Navarro, G., Ochoa, C., Prezza, N.: On the approximation ratio of ordered parsings. IEEE Trans. Inf. Theory **67**(2), 1008–1026 (2021)
17. Raskhodnikova, S., Ron, D., Rubinfeld, R., Smith, A.: Sublinear algorithms for approximating string compressibility. Algorithmica **65**(3), 685–709 (2013)
18. Shallit, J.: String attractors for automatic sequences. CoRR 2012.06840 (2020)
19. Storer, J.A., Szymanski, T.G.: Data compression via textual substitution. J. ACM **29**(4), 928–951 (1982)