

Article

# Approximation Ratios of RePair, LongestMatch and Greedy on Unary Strings <sup>†</sup>

Danny Hucke and Carl Philipp Reh <sup>\*</sup>

Department Elektrotechnik und Informatik, Universität Siegen, D-57068 Siegen, Germany; dannyhucke@gmail.com

<sup>\*</sup> Correspondence: reh@eti.uni-siegen.de

<sup>†</sup> This paper is an extended version of our paper published in the proceedings of SPIRE 2019.

**Abstract:** A grammar-based compressor is an algorithm that receives a word and outputs a context-free grammar that only produces this word. The approximation ratio for a single input word is the size of the grammar produced for this word divided by the size of a smallest grammar for this word. The worst-case approximation ratio of a grammar-based compressor for a given word length is the largest approximation ratio over all input words of that length. In this work, we study the worst-case approximation ratio of the algorithms Greedy, RePair and LongestMatch on unary strings, i.e., strings that only make use of a single symbol. Our main contribution is to show the improved upper bound of  $\mathcal{O}((\log n)^8 \cdot (\log \log n)^3)$  for the worst-case approximation ratio of Greedy. In addition, we also show the lower bound of  $1.34847194 \dots$  for the worst-case approximation ratio of Greedy, and that RePair and LongestMatch have a worst-case approximation ratio of  $\log_2(3)$ .

**Keywords:** data compression; grammar-based compression; approximation algorithm; addition chain



**Citation:** Hucke, D.; Reh, C.P. Approximation Ratios of RePair, LongestMatch and Greedy on Unary Strings. *Algorithms* **2021**, *14*, 65. <https://dx.doi.org/10.3390/a14020065>

Academic Editor: Alberto Policriti

Received: 20 January 2021  
Accepted: 18 February 2021  
Published: 20 February 2021

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

The goal of grammar-based compression is to represent a word  $w$  by a small context-free grammar that produces exactly  $\{w\}$ . Such a grammar is called a straight-line program (SLP) for  $w$ . In the best case, one gets an SLP of size  $\Theta(\log n)$  for a word of length  $n$ , where the size of an SLP is the total length of all right-hand sides of the rules of the grammar. A grammar-based compressor is an algorithm that produces an SLP for a given word  $w$ . There are various grammar-based compressors that can be found in many places in the literature. A well-known example is the classic LZ78-compressor of Lempel and Ziv [1]. Although it was not introduced as a grammar-based compressor, it is straightforward to compute from the LZ78-factorization of  $w$  an SLP for  $w$  of roughly the same size. Other examples include BISECTION [2] and SEQUITUR [3]. In this work, we study the global grammar-based compressors Greedy [4–6], RePair [7] and LongestMatch [8], to which we will also refer to as global algorithms. Global algorithms are important in practice because they show excellent compression results in various fields. For example, Greedy is used in [5] to compress DNA sequences. Among all global algorithms, RePair is probably the most used one. Examples include compressing web graphs [9], searching compressed text [10], suffix array compression [11] and compressing XML [12]. A key concept of global compressors are maximal strings. A maximal string of an SLP  $\mathbb{A}$  is a word that has length at least two and occurs at least twice without overlap as a factor of the right-hand sides of the rules of  $\mathbb{A}$ . Furthermore, no strictly longer word appears at least as many times without overlap as a factor of the right-hand sides of  $\mathbb{A}$ . For an input word  $w$ , a global grammar-based compressor starts with the SLP that has a single rule  $S \rightarrow w$ , where  $S$  is the start nonterminal of the grammar. The SLP is then recursively updated by choosing a maximal string  $\gamma$  of the current SLP and replacing a maximal set of pairwise nonoverlapping occurrences of  $\gamma$  by a new nonterminal  $X$ . Additionally, a new rule  $X \rightarrow \gamma$  is introduced. The algorithm stops when the obtained SLP has no maximal string. In the

case of Greedy the chosen maximal string minimizes the size of the SLP in each round, while RePair selects in each round a most frequent maximal string, and LongestMatch chooses a longest maximal string. Please note that the Greedy algorithm as originally presented in [4–6] is different from the version studied in this work as well as in [13]: The original Greedy algorithm only considers the right-hand side of the start rule for the choice and the replacement of the maximal string. In particular, all other rules do not change after they are introduced.

In [13] the worst-case approximation ratio of grammar-based compressors is studied. For a grammar-based compressor  $\mathcal{C}$  that computes an SLP  $\mathcal{C}(w)$  for a given word  $w$ , one defines the approximation ratio of  $\mathcal{C}$  on  $w$  as the quotient of the size of  $\mathcal{C}(w)$  and the size  $g(w)$  of a smallest SLP for  $w$ . The approximation ratio  $\alpha_{\mathcal{C}}(n)$  is the maximal approximation ratio of  $\mathcal{C}$  among all words of length  $n$ . In [13] the authors provide upper and lower bounds for the approximation ratios of several grammar-based compressors (among them are all compressors mentioned so far), but for none of the compressors the lower and upper bounds match. For LZ78 and BISECTION these gaps were closed in [14]. For all global algorithms the best upper bound on the approximation ratio is  $\mathcal{O}((n/\log n)^{2/3})$  [13], while the best known lower bounds so far are  $\Omega(\log n / \log \log n)$  for RePair [15],  $\Omega(\log \log n)$  for LongestMatch and  $5/(3 \log_3(5)) = 1.137\dots$  for Greedy [13]. In general, the achieved bounds “leave a large gap of understanding surrounding the global algorithms” as the authors in [13] conclude.

Unary words have the form  $a^n$  for some symbol  $a$  and integer  $n \geq 1$ . Grammar-based compression on unary words is strongly related to the field of addition chains, which has been studied for decades (see [16] (Chapter 4.6.3) for a survey) and still is an active topic due to the strong connection to public key cryptosystems (see [17] for a review from that point of view). An addition chain for an integer  $n$  of size  $m$  is a sequence of integers  $1 = k_1, k_2, \dots, k_m = n$  such that for each  $d$  ( $2 \leq d \leq m$ ), there exists  $i, j$  ( $1 \leq i, j < d$ ) such that  $k_i + k_j = k_d$ . It is straightforward to compute from an addition chain for an integer  $n$  of size  $m$  an SLP for  $a^n$  of size  $2m - 2$ . Vice versa, an SLP for  $a^n$  of size  $m$  yields an addition chain for  $n$  of size  $m$ . Therefore, grammar-based compressors on unary inputs can also be thought of as addition chain solvers, i.e., algorithms that find a (small) addition chain for a given integer.

The worst-case approximation ratio for global algorithms is difficult to analyze. A good starting point is therefore to analyze them on unary words because of their simplicity. Even though unary words are not interesting to compress, it is still interesting to look at how global algorithms perform on them. The improved upper bound we show for Greedy uses unary words and is the first improvement that happened in 15 years.

We show the worst-case approximation ratio of RePair and LongestMatch for unary words to be  $\log_2(3)$ . Both algorithms are basically identical to the binary method that produces an addition chain for  $n$  by creating powers of two using repeated squaring, and then the integer  $n$  is represented as the sum of those powers of two that correspond to a one in the binary representation of  $n$ . Based on that information, we show that for any unary input  $w$  the produced SLPs of RePair and LongestMatch have size at most  $\log_2(3) \cdot g(w)$  and we also provide a lower bound.

We improve the upper bound for the approximation ratio of Greedy on unary words to  $\mathcal{O}((\log n)^8 \cdot (\log \log n)^3)$ . In [18], which is the previous version of this article, the authors showed an upper bound for the approximation ratio of Greedy of  $\mathcal{O}(n^{1/4} / \log n)$  for unary inputs, by only analyzing the first three rounds. Here, we present a more in-depth analysis that makes use of every round. We can prove that Greedy produces an SLP of size  $\mathcal{O}((\log n)^9 \cdot (\log \log n)^3)$  on input  $a^n$ , which together with the fact that a smallest SLP for  $a^n$  has size  $\Omega(\log n)$  then yields the improved upper bound of  $\mathcal{O}((\log n)^8 \cdot (\log \log n)^3)$ . To prove the size bound on the SLP produced by Greedy, we distinguish unary and nonunary nonterminals. A nonterminal  $X$  is called unary if its right-hand side is of the form  $X \rightarrow Z^d$  when it is first introduced. Otherwise, it is nonunary. We bound the total number of occurrences of all unary and nonunary nonterminals in the grammar

produced by Greedy separately. For the unary nonterminals, we bound their total number to be  $\mathcal{O}((\log n)^9)$ , while each of them contributes a size of  $\mathcal{O}(\log \log n)$ , which yields a total size contribution of  $\mathcal{O}((\log n)^9 \cdot \log \log n)$ . We then bound the number of occurrences of nonunary nonterminals using the already established number of unary nonterminals, which comes out to be  $\mathcal{O}((\log n)^9 \cdot (\log \log n)^3)$ . Thus, we obtain the desired upper bound on the size of the grammar.

We also show the lower bound of  $1.34847194\dots$  for the approximation ratio of Greedy. The key to achieve this bound is the sequence  $y_k = y_{k-1}^2 + 1$  with  $y_0 = 2$ , which has been studied in [19] (among other sequences), where it is shown that  $y_k = \lfloor \gamma^{2^k} \rfloor$  for  $\gamma = 2.258\dots$ . To prove the lower bound, we show that the SLP produced by Greedy on input  $a^{y_k}$  has size  $3 \cdot 2^k - 1$ , while a smallest SLP for  $a^{y_k}$  has size  $3 \cdot \log_3(\gamma) \cdot 2^k + o(2^k)$  (this follows from a construction used to prove the lower bound for Greedy in [13]).

This paper is an extended version of our paper published in the proceedings of SPIRE 2019 [18].

### Related Work

One of the first appearances of straight-line programs in the literature are [20,21], where they are called word chains (since they generalize addition chains from numbers to words). In [20] it is shown that the function  $g(k, n) = \max\{g(w) \mid w \in \{1, \dots, k\}^n\}$  is in  $\Theta(n / \log_k n)$ . Recall that  $g(w)$  is the size of a smallest SLP for the word  $w$  and thus  $g(k, n)$  measures the worst-case SLP-compression over all words of length  $n$  over a  $k$ -letter alphabet.

The smallest grammar problem is the problem of computing a smallest SLP for a given input word. It is known from [13,22] that in general no grammar-based compressor can solve the smallest grammar problem in polynomial time unless  $P = NP$ . Even worse, unless  $P = NP$  one cannot compute in polynomial time for a given word  $w$  an SLP of size at most  $\frac{8569}{8568} \cdot g(w)$  [13]. One should mention that the constructions to prove these hardness results use alphabets of unbounded size. Although in [13] it is remarked that the construction in [22] works for words over a ternary alphabet, in [23] it is argued that this is not clear at all and a construction for fixed alphabets of size at least 24 is given. However, for grammar-based compression on unary strings as studied in this work (as well as for the problem of computing a smallest addition chain), there is no NP-hardness result, so there might be an optimal polynomial-time algorithm even though it is widely believed that there is none.

Other notable systematic investigations of grammar-based compression are provided in [8,24]. However, in [8], grammar-based compressors are used for universal lossless compression (in the information-theoretical sense), it is shown in [24] that the size of so-called irreducible SLPs (that include SLPs produced by global algorithms) can be upper-bounded by the (unnormalized)  $k$ -th order empirical entropy of the produced string plus some lower order terms.

## 2. Preliminaries

For  $i, j \in \mathbb{N}$  we write  $[i, j] = \{i, i + 1, \dots, j\}$  for  $i \leq j$  and  $[i, j] = \emptyset$  otherwise. For  $m, n \in \mathbb{N}$  we denote by  $m \operatorname{div} n$  the integer division of  $m$  and  $n$ . We denote by  $m \operatorname{mod} n$  the modulo of  $m$  and  $n$ , i.e.,  $m \operatorname{mod} n \in [0, n - 1]$  and

$$m = (m \operatorname{div} n) \cdot n + (m \operatorname{mod} n).$$

If  $m/n$  or  $\frac{m}{n}$  is used, then this refers to the standard division over  $\mathbb{R}$ . Please note that  $m \operatorname{div} n = \lfloor m/n \rfloor$  and  $(m \operatorname{div} n) + (m \operatorname{mod} n) \geq m/n$ .

An alphabet  $\Sigma$  is a finite set of symbols. For a word or string  $w = a_1 \cdots a_n$  over  $\Sigma$  with  $a_1, \dots, a_n \in \Sigma$  and  $n \geq 0$  we write  $|w| = n$  to denote  $w$ 's length. The set of  $\Sigma^*$  consists of all words over  $\Sigma$  and  $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$ , where  $\varepsilon$  is the word of length 0. A unary word is a

word of the form  $a^n$  with  $a \in \Sigma$  and  $n \in \mathbb{N}$ . All other words  $w \in \Sigma^+$  are called *nonunary*. For words  $w, v \in \Sigma^+$  we say that  $v$  is a *factor* of  $w$  if there are  $x, y \in \Sigma^*$  such that  $w = xvy$ .

A *context-free grammar* is a tuple  $(N, \Sigma, P, S)$ , where  $N$  is the finite set of nonterminals,  $\Sigma$  is the alphabet with  $\Sigma \cap N = \emptyset$ ,  $P$  is the set of productions of the form  $X \rightarrow v$ , where  $X \in N$  and  $v \in (\Sigma \cup N)^+$ , and  $S \in N$  is the start symbol. An *SLP* is a context-free grammar  $\mathbb{A} = (N, \Sigma, P, S)$ , where

- every  $X \in N$  has exactly one production, i.e.,

$$|\{(X \rightarrow v) \in P \mid v \in (\Sigma \cup N)^+\}| = 1, \text{ and}$$

- the relation  $\{(A, B) \in N \times N \mid (A \rightarrow v) \in P, B \text{ occurs in } v\}$  is acyclic.

This way, for every  $X \in N$  there exists a unique word  $w \in \Sigma^+$  with  $X \rightarrow^+ w$ . We say that  $\mathbb{A}$  *produces*  $w$  if  $S \rightarrow^+ w$ . Please note that some authors require SLPs to be in Chomsky Normal Form (CNF), i.e., every production is of the form  $A \rightarrow BC$ , where  $B, C \in N$  or  $A \rightarrow a$ , where  $a \in \Sigma$ . We do not make this assumption here because, in general, grammar-based compressors produce SLPs that are not in CNF. Furthermore, every SLP can easily be transformed into an SLP that is in CNF, produces the same word and has roughly the same size. A grammar-based compressor  $\mathcal{C}$  is an algorithm that given an input  $w \in \Sigma^+$  outputs an SLP  $\mathbb{A}$  that produces  $w$ . The size of an SLP is defined as  $|\mathbb{A}| = \sum_{(X \rightarrow v) \in P} |v|$ . For a word  $w \in \Sigma^+$  we write  $g(w)$  for the size of a smallest SLP that produces  $w$ . The *worst-case approximation ratio*  $\alpha_{\mathcal{C}}(k, n)$  of  $\mathcal{C}$  is the maximal approximation ratio over all words of length  $n$  over an alphabet of size  $k$ :

$$\alpha_{\mathcal{C}}(k, n) = \max \left\{ \frac{|\mathcal{C}(w)|}{g(w)} \mid w \in [1, k]^n \right\}.$$

For a given SLP  $\mathbb{A}$ , a word  $\gamma$  is called a *maximal string* of  $\mathbb{A}$  if

- $|\gamma| \geq 2$ ,
- $\gamma$  appears at least twice without overlap as a factor of the right-hand sides,
- and no strictly longer word appears at least as many times as a factor of the right-hand sides without overlap.

**Example 1.** Let  $\mathbb{A} = (\{S, X, Y, Z\}, \{a, b\}, P, S)$  such that  $P$  contains

- $S \rightarrow aXXXXbbYZYZ,$
- $X \rightarrow YbbYZYZa,$
- $Y \rightarrow bbbZZaZ,$  and
- $Z \rightarrow abb.$

The maximal strings of  $\mathbb{A}$  are  $bb, YZ$  and  $bbYZYZ$ . The factors  $bb$  and  $YZ$  occur four times on the right-hand sides without overlap and  $bbYZYZ$  occurs twice without overlap.

A *global grammar-based compressor* (or simply *global algorithm*) starts on input  $w$  with the SLP  $\mathbb{A}_0 = (\{S\}, \Sigma, \{S \rightarrow w\}, S)$ . In each round  $i \geq 1$ , the algorithm selects a maximal string  $\gamma$  of  $\mathbb{A}_{i-1}$  and updates  $\mathbb{A}_{i-1}$  to  $\mathbb{A}_i$  by replacing a largest set of pairwise nonoverlapping occurrences of  $\gamma$  in  $\mathbb{A}_{i-1}$  by a new nonterminal  $X$ . Additionally, the algorithm introduces the rule  $X \rightarrow \gamma$  in  $\mathbb{A}_i$ . The algorithm stops when no maximal string occurs. Please note that the replacement is not unique, e.g., the word  $a^5$  has a unique maximal string  $\gamma = aa$ , which yields SLPs with rules  $S \rightarrow XXa, X \rightarrow aa$  or  $S \rightarrow XaX, X \rightarrow aa$  or  $S \rightarrow aXX, X \rightarrow aa$ . We assume the first variant here, i.e., maximal strings are replaced from left to right. The compressor Greedy that we study in this work chooses a maximal string in each round  $i \geq 1$  such that the size of  $\mathbb{A}_i$  is minimal.

**Example 2 (Greedy).** Let  $w = aaaaaabbabbbbbaabb$ . We have

- $\mathbb{A}_0: S \rightarrow aaaaaabbabbbbbaabb,$
- $\mathbb{A}_1: S \rightarrow aaaaXabXbaaX, X \rightarrow abb,$

$$\begin{aligned} \mathbb{A}_2: S &\rightarrow YYXabXbYX, X \rightarrow abb, Y \rightarrow aa, \\ \mathbb{A}_3: S &\rightarrow YYXZXbYX, X \rightarrow Zb, Y \rightarrow aa, Z \rightarrow ab, \\ \mathbb{A}_4: S &\rightarrow YAZXbA, X \rightarrow Zb, Y \rightarrow aa, Z \rightarrow ab, A \rightarrow YX. \end{aligned}$$

Please note that in the first round, instead of the maximal string  $abb$  the algorithm could also choose the maximal string  $aaabb$ , because both choices yield SLPs of minimal size 15. In the second round, instead of  $aa$  the algorithm could also choose  $aaX$ , because both choices yield SLPs of size 14. Finally, the order of the choices  $ab$  (round 3) and  $YX$  (round 4) could be swapped because both choices yield SLPs of unchanged size 14.

The following lemma from [13] provides a lower bound on the size of an SLP for a word of length  $n$ .

**Lemma 1** ([13] (Lemma 1)). *For every word  $w \in \Sigma^+$  of length  $n$ , we have  $g(w) \geq 3 \log_3(n) - 3$ .*

### 3. Upper Bound for Greedy

To show our improved upper bound for the approximation ratio of Greedy on unary words, we are first going to prove that the size of the SLP produced by Greedy for the input  $a^n$  is upper-bounded by  $\mathcal{O}((\log n)^9 \cdot (\log \log n)^3)$ .

**Proposition 1.** *For all  $n$ , we have*

$$|\text{Greedy}(a^n)| \in \mathcal{O}((\log n)^9 \cdot (\log \log n)^3).$$

First, we need to prove several lemmas that are fulfilled for any global algorithm. When we apply specific arguments to Greedy, we draw attention to it. For better readability, we will use  $X_0 = a$ , i.e., the input is  $X_0^n$ . Furthermore, let  $\mathbb{A}_i = (N_i, \{X_0\}, P_i, S)$  be the SLP obtained by the global algorithm on input  $X_0^n$  after  $i$  rounds. Please note that until the algorithm stops, we have  $|N_i \setminus \{S\}| = i$  since exactly one new nonterminal is introduced in each round. If we quantify over the rounds of the algorithm, we always implicitly mean that the statements hold until the algorithm stops. If  $i$  is mentioned without a quantification, then the statement holds for any  $\mathbb{A}_i$  constructed after some round  $i$  of the algorithm.

**Lemma 2.** *For every  $i$ , there is a fixed order  $X_i > X_{i-1} > \dots > X_1$  of the nonterminals in  $N_i \setminus \{S\}$  such that every right-hand side of a rule  $(X \rightarrow v) \in P_i$  satisfies*

$$v \in X_i^* X_{i-1}^* \dots X_1^* X_0^*.$$

**Proof.** We prove this property by induction. Initially, the property holds for the SLP  $\mathbb{A}_0$  since  $N_0 \setminus \{S\} = \emptyset$  and the only rule  $S \rightarrow X_0^n$  satisfies  $X_0^n \in X_0^*$ . Now assume the claim is true for  $\mathbb{A}_i$ , i.e., each right-hand side of a rule in  $P_i$  is a word from  $X_i^* X_{i-1}^* \dots X_1^* X_0^*$ . Please note that any nonempty factor of such a right-hand side is a word from  $X_k^* \dots X_{j+1}^* X_j^+$  for some  $i \geq k > j \geq 0$ . Therefore, assume the global algorithm chooses a maximal string  $\gamma \in X_k^* \dots X_{j+1}^* X_j^+$  in round  $i + 1$  and  $(X \rightarrow \gamma) \in P_{i+1}$  is the corresponding new rule. We show that  $v \in X_i^* X_{i-1}^* \dots X^* X_j^* \dots X_1^* X_0^*$  for all rules  $(Y \rightarrow v) \in P_{i+1}$ , i.e., the order of the nonterminals after round  $i + 1$  is obtained by inserting the new nonterminal  $X$  directly before  $X_j$  in the previous order. First, this is obviously true for the new rule  $X \rightarrow \gamma$  as well as for all rules that have not been modified during round  $i + 1$ . It remains to check the rules  $(Y \rightarrow v) \in P_{i+1}$  that are obtained from a rule  $(Y \rightarrow v') \in P_i$  by replacing a largest set of pairwise nonoverlapping occurrences of  $\gamma$  in  $v'$  by the new nonterminal  $X$ . If  $\gamma = X_j^d$  ( $d \geq 2$ ) is unary and  $X_j^\ell$  is the single maximal  $X_j$ -block that occurs in  $v'$ , then replacing occurrences of  $\gamma$  from left to right yields  $X^{\ell \text{ div } d} X_j^{\ell \bmod d}$  as the new maximal blocks of  $X$  and  $X_j$  in  $w$ . It follows that  $v \in X_i^* X_{i-1}^* \dots X^* X_j^* \dots X_1^* X_0^*$ . If otherwise  $\gamma$  is not a unary word, i.e.,  $\gamma \in X_k^+ \dots X_{j+1}^* X_j^+$  for  $i \geq k > j \geq 0$ , then  $v'$  has exactly one occurrence of  $\gamma$  as

a factor. It follows that  $v \in X_i^* X_{i-1}^* \cdots X_k^* X X_j^* \cdots X_1^* X_0^*$  and thus  $v$  satisfies the claim. This finishes the induction.  $\square$

In other words, there is at most one maximal block for each symbol on each right-hand side and the order of these blocks is the same for all rules. Similar to the case distinction in the last steps of the proof of Lemma 2, we will distinguish two types of nonterminals. Let  $X \rightarrow \gamma$  be the introduced rule in some round of the algorithm. If  $\gamma$  is unary, then we call  $X$  a unary nonterminal. Otherwise, we call  $X$  nonunary. We categorize  $X_0$  as a unary nonterminal, although formally  $X_0$  is not a nonterminal. Please note that the type of a nonterminal is decided when it is introduced and does not change later, i.e., even if the right-hand side of a unary nonterminal becomes nonunary during the execution of the algorithm, the type of the nonterminal stays the same. Our strategy to prove Theorem 1 is to bound the total number of occurrences of unary nonterminals and nonunary nonterminals on right-hand sides independently. It follows from Lemma 2 that every factor that occurs more than once on the right-hand side of a single rule is unary. The following lemma is a direct consequence of that fact.

**Lemma 3.** *Every nonunary nonterminal occurs at most once on the right-hand side of each rule at any time of the algorithm.*

**Corollary 1.** *If a unary nonterminal  $X$  is introduced and  $X \rightarrow Z^d$  with  $d \geq 2$  is the corresponding rule for  $X$ , then  $Z$  is a unary nonterminal.*

Next, we bound the number of rules that contain a unary nonterminal  $X$  on the right-hand side. For a nonterminal  $X$ , including  $X_0$ , let  $\#_i(X)$  be the number of rules of  $\mathbb{A}_i$  where  $X$  occurs on the right-hand side, or more formally

$$\#_i(X) = |\{(Y \rightarrow v) \in P_i \mid X \text{ occurs in } v\}|.$$

The next two lemmas describe how  $\#_i$  evolves depending on the type of the introduced nonterminal.

**Lemma 4.** *If a nonunary nonterminal  $X$  is introduced in some round  $i + 1$ , then for every  $X' \neq X$  (including  $X_0$ ), we have  $\#_{i+1}(X') \leq \#_i(X')$ .*

**Proof.** To prove this point, we use that all rules  $(Y \rightarrow v) \in P_i$  satisfy  $v \in X_i^* X_{i-1}^* \cdots X_1^* X_0^*$  (Lemma 2). Since  $X$  is nonunary, the chosen maximal string  $\gamma$  satisfies  $\gamma \in X_k^+ \cdots X_{j+1}^* X_j^+$  for  $i \geq k > j \geq 0$ . If a nonterminal  $X'$  does not occur in  $\gamma$ , then  $\#_{i+1}(X') = \#_i(X')$ . So, assume the nonterminal  $X'$  occurs in  $\gamma$ , i.e.,  $X' = X_t$  for  $t \in [j, k]$ . Note first that  $X_t$  occurs on the right-hand side of the new rule  $(X \rightarrow \gamma) \in P_{i+1}$ . It follows that to prove the claimed result, we must show that for at least one rule  $(Y \rightarrow v) \in P_i$  such that  $X_t$  occurs in  $v$ , all occurrences of  $X_t$  must disappear, i.e.,  $X_t$  does not occur in  $v'$  for  $(Y \rightarrow v') \in P_{i+1}$ . Let

$$M = \{Y \in N_i \mid (Y \rightarrow v) \in P_i \text{ and } \gamma \text{ is a factor of } v\} \subseteq N_i$$

be the set of nonterminals of  $\mathbb{A}_i$  where the corresponding rule is modified in round  $i + 1$ . Please note that  $|M| \geq 2$  since a maximal string occurs at least twice on all right-hand sides and  $\gamma$  occurs at most once as factor of each rule since  $X$  is nonunary (Lemma 3). If  $k < t < j$  then for all  $Y \in M$  and  $(Y \rightarrow v') \in P_{i+1}$ , the nonterminal  $X_t$  does not occur in  $v'$  anymore since the complete  $X_t$ -block (among other symbols) has been replaced. This means that  $\#_{i+1}(X_t) < \#_i(X_t)$  since one new rule contains  $X_t$  on the right-hand side, while for at least two rules the occurrences of  $X_t$  have been removed in round  $i + 1$ . If otherwise  $t = k$  or  $t = j$ , then the same argument fails since for  $Y \in M$  and  $(Y \rightarrow v') \in P_{i+1}$ , the right-hand side  $v'$  could still contain  $X_t$  since it is not necessarily true that the complete  $X_t$ -block has been replaced. However, due to the properties of a maximal string, we show that  $X_t$  does

not occur in  $v'$  for at least one  $Y \in M$  and  $(Y \rightarrow v') \in P_{i+1}$ . Towards a contradiction, assume  $X_t$  occurs in  $v'$  for all  $Y \in M$  and  $(Y \rightarrow v') \in P_{i+1}$ . This means that for all  $Y \in M$  and  $(Y \rightarrow v) \in P_i$ , the length of the maximal  $X_t$ -block that occurs in  $v$  is strictly larger than the length of the maximal  $X_t$ -block that occurs in  $\gamma$ . If  $t = k$ , it follows that  $X_k\gamma$  is a factor of  $v$  for all  $Y \in M$  and  $(Y \rightarrow v) \in P_i$ , and symmetrically, if  $t = j$  then  $\gamma X_j$  is a factor of  $v$  for all  $Y \in M$  and  $(Y \rightarrow v) \in P_i$ . This contradicts the property that no strictly longer string than  $\gamma$  occurs at least as often on the right-hand sides of the rules. It follows that in this case  $\#_{i+1}(X_t) \leq \#_i(X_t)$ , which finishes the proof.  $\square$

**Lemma 5.** *If a unary nonterminal  $X$  is introduced in some round  $i + 1$  and  $X \rightarrow Z^d$  with  $d \geq 2$  is the corresponding rule, then  $\#_{i+1}(X) \leq \#_i(Z)$  and  $\#_{i+1}(Z) \leq \#_i(Z) + 1$ .*

**Proof.** Both points are straightforward: A rule  $(Y \rightarrow v) \in P_{i+1}$  only contains  $X$  on the right-hand side if  $Z^d$  is a factor of  $v'$  for  $(Y \rightarrow v') \in P_i$ , which shows that  $\#_{i+1}(X) \leq \#_i(Z)$ . For the second point, note that if  $(Y \rightarrow v) \in P_i$  does not contain  $Z$  on the right-hand side, then the same is true for (the unchanged rule)  $(Y \rightarrow v) \in P_{i+1}$ . The only rule where  $Z$  occurs new is the new rule  $X \rightarrow Z^d$ , and thus  $\#_{i+1}(Z) \leq \#_i(Z) + 1$ .  $\square$

So far, we have shown that when a unary nonterminal  $X$  is introduced and  $X \rightarrow Z^d$  with  $d \geq 2$  is the corresponding rule, then  $Z$  is a unary nonterminal as well (Corollary 1). Furthermore, we argued that introducing a nonunary nonterminal does not increase the number of rules where a unary nonterminal occurs on the right-hand side (Lemma 4). It follows that we can upper-bound the number of unary nonterminals and the number of rules where those nonterminals occur on right-hand sides independently of the nonunary nonterminals.

To do so, we inductively define a binary tree  $T_i$  that describes how the unary nonterminals evolve until  $\mathbb{A}_i$  is reached. All nodes in the tree are labeled with  $(X, k)$ , where  $X$  is a unary nonterminal and  $k$  is an upper bound on  $\#_j(X)$  for some  $j$ .

- (1) Initially, the tree  $T_0$  only contains a single node that is labeled with  $(X_0, 1)$ .
- (2) If a rule  $X \rightarrow Z^d$  is introduced in round  $i + 1$  for some  $d \geq 2$ , then we update  $T_i$  to  $T_{i+1}$  by adding two children to the unique leaf that is labeled with  $(Z, k)$  for some  $k$ . The new left child is labeled with  $(Z, k + 1)$  and the new right child is labeled with  $(X, k)$  as depicted on the left of Figure 1.
- (3) If otherwise a nonunary nonterminal is introduced in round  $i + 1$ , then  $T_{i+1} = T_i$ , i.e., nonunary nonterminals are ignored.

The initial tree  $T_0$  reflects that the only unary nonterminal of  $\mathbb{A}_0$  is  $X_0$  and  $\#_0(X_0) = 1$ . If the tree is modified according to point (2) of the definition, this refers to Lemma 5, where  $\#_{i+1}(X) \leq \#_i(Z)$  and  $\#_{i+1}(Z) \leq \#_i(Z) + 1$  is shown when a rule  $X \rightarrow Z^d$  for  $d \geq 2$  is introduced.

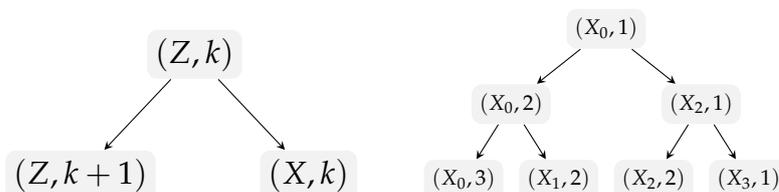
The *level* of a node is the length of the path from the node to the root. For a unary nonterminal  $X \in N_i$ , we denote by  $\text{level}_i(X)$  the level of the unique leaf of  $T_i$  that is labeled with  $(X, k)$  for some  $k$ .

**Example 3.** *Assume that the first three rules introduced by a global algorithm are  $X_2 \rightarrow X_0^{d_1}$  in the first round,  $X_1 \rightarrow X_0^{d_2}$  in the second round and  $X_3 \rightarrow X_2^{d_3}$  in the third round for  $d_1, d_2, d_3 \geq 2$ . The tree  $T_3$  that corresponds to this introduced rules is depicted on the right of Figure 1. The indices for the introduced nonterminals are chosen such that the ordering of the nonterminals in  $N_3 \setminus \{S\}$  (see Lemma 2) is  $X_3 > X_2 > X_1$ , i.e., all right-hand sides of rules are contained in  $X_3^* X_2^* X_1^* X_0^*$ .*

The corresponding SLPs  $\mathbb{A}_0, \mathbb{A}_1, \mathbb{A}_2$  and  $\mathbb{A}_3$  are depicted next, where we simply use  $*$  instead of the exact exponents of the symbols due to better readability.

$$\begin{array}{lll}
 \mathbb{A}_0: S \rightarrow X_0^* & \mathbb{A}_2: S \rightarrow X_2^* X_1^* X_0^* & \mathbb{A}_3: S \rightarrow X_3^* X_2^* X_1^* X_0^* \\
 \mathbb{A}_1: S \rightarrow X_2^* X_0^* & X_2 \rightarrow X_1^* X_0^* & X_2 \rightarrow X_1^* X_0^* \\
 X_2 \rightarrow X_0^* & X_1 \rightarrow X_0^* & X_1 \rightarrow X_0^* \\
 & & X_3 \rightarrow X_2^*
 \end{array}$$

Please note that in this example, we have  $\#_3(X_0) \leq 3, \#_3(X_2) \leq 2, \#_3(X_1) \leq 2$  and  $\#_3(X_3) \leq 1$ , which is exactly the information contained in the second components of the leaf labels in  $T_3$ . Furthermore, we have  $\text{level}_3(X_i) = 2$  for  $i \in [0, 3]$  in this example.



**Figure 1.** On the left, the general pattern that is applied during the construction of  $T_i$  is illustrated, where the split refers to a rule  $X \rightarrow Z^d$  that has been introduced by the global algorithm. On the right, the tree  $T_3$  that corresponds to the introduced rules of Example 3 is shown.

The following lemma is a direct consequence of the fact that the maximal  $k$  that occurs for some label  $(X, k)$  is incremented from one level to the next level (as described in Lemma 5).

**Lemma 6.** For each node of  $T_i$  at level  $m$  that is labeled with  $(X, k)$  for some unary nonterminal  $X \in N_i$ , we have  $k \leq m + 1$ . Also, let  $X \in N_i$  be a unary nonterminal. Then we have  $\#_i(X) \leq \text{level}_i(X) + 1$ .

So far, we provided information about the number of rules where a unary nonterminal occurs. Next, we move on to the total number of occurrences of a unary nonterminal on all right-hand sides. We denote by  $t_i(X)$  the total number of occurrences of  $X$  on right-hand sides of rules in  $\mathbb{A}_i$ . We have  $\#_i(X) \leq t_i(X)$  by the definition of both functions and for a nonunary nonterminal  $X$ , we have  $\#_i(X) = t_i(X)$  due to Lemma 3.

**Lemma 7.** Let  $X \rightarrow \gamma$  be the rule that is introduced in some round  $i + 1$  and let  $M = \{Y \in N_i \mid Y \text{ occurs in } \gamma\}$ . We have

- (1)  $t_{i+1}(Y) \leq t_i(Y)$  for all  $Y \in N_i = N_{i+1} \setminus \{X\}$ , and
- (2)  $\sum_{Y \in M} t_{i+1}(Y) + t_{i+1}(X) \leq \sum_{Y \in M} t_i(Y)$ .

**Proof.** Point (1) is straightforward: For  $Y \in M$  let  $Y^\ell$  be the maximal  $Y$ -block that occurs as a factor in  $\gamma$  for some  $\ell \geq 1$ . Replacing  $\gamma$  on right-hand sides yields that at least two occurrences of  $Y^\ell$  are eliminated while only  $Y^\ell$  is added as a part of the new rule  $X \rightarrow \gamma$ . If otherwise  $Y \notin M$ , then  $t_{i+1}(Y) = t_i(Y)$  because the occurrences of  $Y$  are not affected by the new rule.

Point (2) is also based on a simple observation. Please note that  $\sum_{Y \in M} t_i(Y)$  describes the part of the SLP  $\mathbb{A}_i$  that is affected by the replacement of  $\gamma$  in round  $i + 1$ , and  $\sum_{Y \in M} t_{i+1}(Y) + t_{i+1}(X)$  is the size of that part in  $\mathbb{A}_{i+1}$  after the occurrences of  $\gamma$  are replaced by  $X$  plus the new occurrences in the introduced rule. All other parts of  $\mathbb{A}_i$  are not affected by the new rule. Now the properties of a maximal string ensure that  $|\mathbb{A}_{i+1}| \leq |\mathbb{A}_i|$ . The extreme case where  $\gamma$  has length two and occurs only twice without

overlap on the right-hand sides of rules in  $\mathbb{A}_i$  satisfies  $|\mathbb{A}_{i+1}| = |\mathbb{A}_i|$ . All other cases even satisfy  $|\mathbb{A}_{i+1}| < |\mathbb{A}_i|$ . Point (2) directly follows.  $\square$

Our next goal is to bound  $t_i(X)$  depending on  $\text{level}_i(X)$  for a unary nonterminal  $X$ . To do so, we now apply arguments specific to Greedy. Recall that Greedy selects a maximal string that minimizes the size of the obtained SLP in each round.

**Lemma 8.** *Let  $Z \in N_i \cup \{X_0\}$  be a unary nonterminal and assume a rule  $X \rightarrow Z^d$  for some  $d \geq 2$  is introduced by Greedy in round  $i + 1$ . We have*

$$t_{i+1}(X) + t_{i+1}(Z) \leq 2\sqrt{t_i(Z)}\sqrt{\#_i(Z) + 1} + 1.$$

**Proof.** If a unary nonterminal  $X$  and a rule  $X \rightarrow Z^d$  with  $d \geq 2$  are introduced in round  $i + 1$ , then the choice of  $d$  only depends on the maximal  $Z$ -blocks occurring on all right-hand sides of rules in  $\mathbb{A}_i$  since the remaining part of  $\mathbb{A}_i$  does not change. Assume that  $\#_i(Z) = k$  and let  $\ell_1, \dots, \ell_k$  be the lengths of the maximal  $Z$ -blocks occurring on right-hand sides of  $\mathbb{A}_i$ , i.e.,  $\sum_{j=1}^k \ell_j = t_i(Z)$ . Then Greedy minimizes  $t_{i+1}(X) + t_{i+1}(Z) = d + \sum_{j=1}^k (\ell_j \text{div } d) + (\ell_j \text{mod } d)$ , where  $d$  is the size of the new rule  $X \rightarrow Z^d$  and for each  $j \in [1, k]$  a maximal block  $Z^{\ell_j}$  on the right-hand side of a rule in  $\mathbb{A}_i$  is transformed into  $X^{\ell_j \text{div } d} Z^{\ell_j \text{mod } d}$ . Due to the greedy nature of the algorithm, the following equation holds for all  $d \geq 1$ :

$$\begin{aligned} t_{i+1}(X) + t_{i+1}(Z) &\leq d + \sum_{j=1}^k (\ell_j \text{div } d) + (\ell_j \text{mod } d) \\ &\leq d + \sum_{j=1}^k \frac{\ell_j}{d} + k(d - 1) \\ &= d + \frac{t_i(Z)}{d} + k(d - 1). \end{aligned}$$

Please note that the chosen maximal string has length at least 2, but the upper bound also holds for  $d = 1$  since in this case we have  $t_{i+1}(X) + t_{i+1}(Z) \leq t_i(Z)$  due to Lemma 7 (point (2)). If we apply  $d = \lceil \sqrt{t_i(Z)} / \sqrt{k + 1} \rceil$ , we get

$$\begin{aligned} t_{i+1}(X) + t_{i+1}(Z) &\leq \left\lceil \frac{\sqrt{t_i(Z)}}{\sqrt{k + 1}} \right\rceil + \frac{t_i(Z)}{\left\lceil \frac{\sqrt{t_i(Z)}}{\sqrt{k + 1}} \right\rceil} + k \left( \left\lceil \frac{\sqrt{t_i(Z)}}{\sqrt{k + 1}} \right\rceil - 1 \right) \\ &\leq \frac{\sqrt{t_i(Z)}}{\sqrt{k + 1}} + 1 + \frac{t_i(Z)}{\frac{\sqrt{t_i(Z)}}{\sqrt{k + 1}}} + k \left( \frac{\sqrt{t_i(Z)}}{\sqrt{k + 1}} \right) \\ &= (k + 1) \frac{\sqrt{t_i(Z)}}{\sqrt{k + 1}} + \frac{t_i(Z) \cdot \sqrt{k + 1}}{\sqrt{t_i(Z)}} + 1 \\ &= 2\sqrt{t_i(Z)}\sqrt{k + 1} + 1. \end{aligned}$$

Together with  $k = \#_i(Z)$  this proves the lemma.  $\square$

The following lemma is essential for the proof of Theorem 1 since we bound the total number of occurrences of a unary nonterminal depending on its level.

**Lemma 9.** *Let  $X \in N_i \cup \{X_0\}$  be a unary nonterminal with  $\text{level}_i(X) = m$ . We have*

$$t_i(X) \leq 2^{2-2^{1-m}} n^{2-m} \prod_{j=1}^m (m + 2 - j)^{2^{-j}}. \tag{1}$$

**Proof.** We prove the lemma by induction on  $m = \text{level}_i(X)$  and we start with  $m = 0$ . The only SLP  $\mathbb{A}_i$  that contains a unary nonterminal  $X$  such that  $\text{level}_i(X) = 0$  is the initial SLP  $\mathbb{A}_0$  and the unary nonterminal is  $X = X_0$ . Please note that the maximal string  $\gamma$  chosen by any global algorithm in the first round on input  $X_0^n$  trivially satisfies  $\gamma \in X_0^*$  and thus the two unary nonterminals of  $\mathbb{A}_1$  have level one. We have  $t_0(X_0) = n$  and this is exactly what we obtain when  $m = 0$  is used on the right side of Equation (1) (the empty product is considered to be 1).

Now assume any unary nonterminal that has level  $m$  satisfies the claimed bound and we consider a unary nonterminal  $X$  such that  $\text{level}_i(X) = m + 1 > 0$  for some  $i$ . It follows from the definition that there is a leaf node at level  $m + 1$  in  $T_i$  that is labeled with  $(X, k)$  for some  $k$ . There are two cases that need to be distinguished. Either this leaf is a left child or a right child of its parent node. Assume that  $(X, k)$  is the label of a right child and let  $(Z, k + 1)$  be the label of the left sibling of that node. To prove both cases simultaneously, we prove the upper bound for  $X$  and for  $Z$ , i.e., we use  $Z$  to cover the second case where the node is a left child. The parent node of  $(Z, k + 1)$  and  $(X, k)$  is labeled with  $(Z, k)$  (see Figure 1 on the left). Let  $i' < i$  be the maximal  $i'$  such that  $\text{level}_{i'}(Z) = m$ , i.e.,  $X \rightarrow Z^d$  for  $d \geq 2$  is the introduced rule in round  $i' + 1$  and  $(Z, k)$  is the label of a leaf at level  $m$  in  $T_{i'}$ . By induction, we have  $t_{i'}(Z) \leq 2^{2-2^{1-m}} n^{2^{-m}} \prod_{j=1}^m (m + 2 - j)^{2^{-j}}$ . Now by Lemma 8, we have

$$t_{i'+1}(X) + t_{i'+1}(Z) \leq 2\sqrt{t_{i'}(Z)}\sqrt{\#_{i'}(Z)} + 1 + 1.$$

Together with  $\#_{i'}(Z) \leq m + 1$  (Lemma 6), this yields

$$\begin{aligned} t_{i'+1}(X) + t_{i'+1}(Z) &\leq 2 \left( 2^{2-2^{1-m}} n^{2^{-m}} \prod_{j=1}^m (m + 2 - j)^{2^{-j}} \right)^{\frac{1}{2}} (m + 2)^{\frac{1}{2}} + 1 \\ &= 2^{2-2^{-m}} n^{2^{-m-1}} \prod_{j=1}^m (m + 2 - j)^{2^{-j-1}} (m + 2)^{2^{-1}} + 1 \\ &= 2^{2-2^{-m}} n^{2^{-m-1}} \prod_{j=2}^{m+1} (m + 2 - j + 1)^{2^{-j}} (m + 2)^{2^{-1}} + 1 \\ &= 2^{2-2^{-m}} n^{2^{-m-1}} \prod_{j=1}^{m+1} (m + 2 - j + 1)^{2^{-j}} + 1. \end{aligned}$$

Using the fact that  $t_{i'+1}(X) \geq 2$  (there are at least two nonoverlapping occurrences of a maximal string) and  $t_{i'+1}(Z) \geq 2$  (the new rule contains  $Z$  at least twice) yields the claimed upper bound on  $t_{i'+1}(X)$  and  $t_{i'+1}(Z)$ . Finally, this upper bound holds for  $i \geq i' + 1$  due to Lemma 7 (point (1)).  $\square$

**Corollary 2.** Let  $X \in N_i \cup \{X_0\}$  be a unary nonterminal with  $\text{level}_i(X) = m$ . We have

$$t_i(X) \leq 4n^{2^{-m}}(m + 2).$$

**Proof.** Please note that  $2^{2-2^{-m}} \leq 4$  for all  $m \geq 0$ . We upper-bound the right side of Equation (1) (Lemma 9) as follows:

$$\begin{aligned} 2^{2-2^{1-m}} n^{2^{-m}} \prod_{j=1}^m (m + 2 - j)^{2^{-j}} &\leq 4n^{2^{-m}} \prod_{j=1}^m (m + 2)^{2^{-j}} \\ &= 4n^{2^{-m}} (m + 2)^{\sum_{j=1}^m 2^{-j}} \\ &\leq 4n^{2^{-m}} (m + 2) \end{aligned}$$

$\square$

What we achieved so far is to bound the total size  $t_i(X)$  that a unary nonterminal  $X$  contributes on right-hand sides of the rules depending on  $\text{level}_i(X)$ . Next, we bound the size that nonunary nonterminals contribute to  $|\mathbb{A}_i|$  depending on the levels of all unary nonterminals. To do so, we need the following definitions. Let  $R_i(X)$  be the number of distinct right neighbors of  $X$  (which are not equal to  $X$ ) on right-hand sides plus the number of occurrences of  $X$  as the last symbol of a right-hand side in  $\mathbb{A}_i$ , i.e.,

$$R_i(X) = |\{A \in N_i \cup \{X_0\} \mid A \neq X, XA \text{ occurs on a right-hand side in } \mathbb{A}_i\}| + |\{(Y \rightarrow vX) \in P_i \mid Y \in N_i, v \in (N_i \cup \{X_0\})^*\}|.$$

Let  $L_i(X)$  be the number of distinct left neighbors of  $X$  (which are not equal to  $X$ ) on right-hand sides plus the number of occurrences of  $X$  as the first symbol of a right-hand side in  $\mathbb{A}_i$ , i.e.,

$$L_i(X) = |\{A \in N_i \cup \{X_0\} \mid A \neq X, AX \text{ occurs on a right-hand side in } \mathbb{A}_i\}| + |\{(Y \rightarrow Xv) \in P_i \mid Y \in N_i, v \in (N_i \cup \{X_0\})^*\}|.$$

Also, let  $f_i(X) = \#_i(X) - R_i(X)$  and  $g_i(X) = \#_i(X) - L_i(X)$ . Please note that  $R_i(X) \leq \#_i(X)$  and  $L_i(X) \leq \#_i(X)$  since for each right-hand side of a rule there is at most one right (respectively, left) neighbor  $A \neq X$  for some occurrence of  $X$  due to Lemma 2 and each right-hand side can contain  $X$  at most once as the last (respectively, first) symbol. Also,  $\#_i(X) = R_i(X)$  means that all maximal  $X$ -blocks on right-hand sides are either at the end of the right-hand side or are followed by a distinct symbol. Similarly,  $\#_i(X) = L_i(X)$  means that all maximal  $X$ -blocks on right-hand sides are either at the beginning of the right-hand side or are preceded by a distinct symbol. The following lemmas describe how the functions  $f_i(X)$  and  $g_i(X)$  evolve.

**Lemma 10.** *If  $X \in N_i \cup \{X_0\}$  then  $f_{i+1}(X) \leq f_i(X)$ . If a nonunary, maximal string  $\gamma = Xv$  is selected in round  $i + 1$  for some  $v \in (N_i \cup \{X_0\})^+$ , then  $f_{i+1}(X) < f_i(X)$ .*

**Proof.** Let  $Y \rightarrow \gamma$  be the introduced rule in round  $i + 1$ . If  $X$  does not occur in  $\gamma$ , then it is straightforward to see that  $f_{i+1}(X) \leq f_i(X)$  since  $\#_{i+1}(X) = \#_i(X)$  and  $R_{i+1}(X) \geq R_i(X)$ . The new nonterminal  $Y$  could be a new right neighbor for some occurrences of  $X$ , but all occurrences of  $X$  which have this new right neighbor  $Y$  in  $\mathbb{A}_{i+1}$  shared the same right neighbor in  $\mathbb{A}_i$  (the first symbol of  $\gamma$ ).

If otherwise  $X$  occurs in  $\gamma$ , then first assume that  $\gamma = X^d$  for some  $d \geq 2$ . Please note that replacing an occurrence of  $\gamma$  on the right-hand side of a rule  $(Z \rightarrow u) \in P_i$  either removes all occurrences of  $X$  on this right-hand side (in case  $u$  contains a maximal  $X$ -block of length  $k \cdot d$  for some integer  $k \geq 1$ ) or the right neighbor of the maximal  $X$ -block in  $u$  does not change in the modified rule  $(Z \rightarrow u') \in P_{i+1}$  since occurrences of  $\gamma$  are replaced from left to right. It follows that the only way to obtain  $R_{i+1}(X) < R_i(X)$  is to remove all occurrences of  $X$  on a right-hand side, but then  $\#_i(X)$  decreases by the same value. Additionally, the new rule  $Y \rightarrow X^d$  adds a new right-hand side to  $\#_{i+1}(X)$ , but since  $X$  is the last symbol on this right-hand side it follows that  $R_{i+1}(X)$  is incremented as well. Together this yields  $f_{i+1}(X) \leq f_i(X)$  in this case.

The case remains where  $\gamma$  is nonunary and  $X$  occurs in  $\gamma$ . Here, we have  $\#_{i+1}(X) \leq \#_i(X)$  due to Lemma 4 and  $\gamma$  occurs at most once on each right-hand side due to Lemma 2. However, again, the only way to reduce  $R_{i+1}(X)$  compared to  $R_i(X)$  is to remove all occurrences of  $X$  on a right-hand side, but then again  $\#_{i+1}(X)$  decreases by the same value. This yields  $f_{i+1}(X) \leq f_i(X)$ .

Assume now that a nonunary, maximal string  $\gamma = Xv$  for some  $v \in (N_i \cup \{X_0\})^+$  is selected in round  $i + 1$ . We show that  $f_{i+1}(X) < f_i(X)$ . If  $X$  only occurs in the new rule in  $\mathbb{A}_{i+1}$  after occurrences of  $\gamma$  are replaced on all right-hand sides in  $\mathbb{A}_i$ , i.e., all modified rules do not contain  $X$  anymore, then  $\#_{i+1}(X) < \#_i(X)$  because at least two right-hand sides do not contain  $X$  as a factor anymore while only the new rule  $Y \rightarrow \gamma$  adds a new right-hand

side which contains  $X$  to  $\#_{i+1}(X)$ . Also, we have  $R_{i+1}(X) = R_i(X)$  in this case and thus  $f_{i+1}(X) < f_i(X)$  because all rules where the maximal  $X$ -block is removed shared the same right neighbor due to the fact that  $\gamma = Xv$  is the chosen maximal string. However,  $Xv$  still occurs in the new rule of  $\mathbb{A}_{i+1}$  and thus  $R_{i+1}(X) = R_i(X)$ . If otherwise at least one of the modified rules still contains  $X$  on the right-hand side, then  $XY$  is a factor of this right-hand side in  $\mathbb{A}_{i+1}$  after the replacement of  $\gamma$ . It follows that  $R_{i+1}(X) > R_i(X)$  in this case and thus  $f_{i+1}(X) < f_i(X)$  because each distinct right neighbor of  $X$  in  $\mathbb{A}_i$  is still a right neighbor of  $X$  in  $\mathbb{A}_{i+1}$  as argued above, but additionally  $XY$  is new since  $Y$  is a new nonterminal.  $\square$

The same result does not hold for  $g_i(X)$ . In particular,  $g_{i+1}(X) > g_i(X)$  is possible when a rule  $Y \rightarrow X^d$  is introduced in round  $i + 1$  for some  $d \geq 2$  due to the assumption that global algorithms replace occurrences of the maximal string from left to right. For example, assume that  $AX^4$ ,  $BX^7$  and  $CX^{10}$  are the maximal  $X$ -blocks on right-hand sides of  $\mathbb{A}_i$  including distinct left neighbors for each  $X$ -block ( $A$ ,  $B$  and  $C$ ). Therefore, we have  $\#_i(X) = 3$ ,  $L_i(X) = 3$  and thus  $g_i(X) = 0$  in this example. If now a rule  $Y \rightarrow X^3$  is introduced, then this yields  $AYX$ ,  $BY^2X$  and  $CY^3X$  after replacing  $X^3$ . Hence we have  $\#_{i+1}(X) = 4$ ,  $L_{i+1}(X) = 2$  and thus  $g_{i+1}(X) = 2$ . We show in the following lemma that this is the only case where  $g_{i+1}(X) > g_i(X)$  occurs.

**Lemma 11.** *Let  $X \in N_i \cup \{X_0\}$ . If a rule  $Y \rightarrow \gamma$  is introduced in round  $i + 1$  such that  $\gamma \notin X^+$ , then  $g_{i+1}(X) \leq g_i(X)$ . If a nonunary, maximal string  $\gamma = Xv$  is selected in round  $i + 1$  for some  $v \in (N_i \cup \{X_0\})^+$ , then  $g_{i+1}(X) < g_i(X)$*

**Proof.** The arguments are similar to the corresponding cases in Lemma 10. Let  $Y \rightarrow \gamma$  be the introduced rule in round  $i + 1$ . If  $X$  does not occur in  $\gamma$ , then  $g_{i+1}(X) \leq g_i(X)$  since  $\#_{i+1}(X) = \#_i(X)$  and  $L_{i+1}(X) \geq L_i(X)$ . The new nonterminal  $Y$  could be a new left neighbor for some occurrences of  $X$  in  $\mathbb{A}_{i+1}$ , but all these occurrences of  $X$  shared the same left neighbor in  $\mathbb{A}_i$  (the last symbol of  $\gamma$ ).

If otherwise  $\gamma$  is nonunary and contains  $X$ , then we have  $\#_{i+1}(X) \leq \#_i(X)$  due to Lemma 4 and  $\gamma$  occurs at most once on each right-hand side due to Lemma 2. The only way to obtain  $L_{i+1}(X) < L_i(X)$  is again to remove all occurrences of  $X$  on a right-hand side, but then  $\#_{i+1}(X)$  decreases by the same value. Please note that due to the assumption that  $\gamma$  is nonunary, it is not possible to modify two (or more) rules such that the maximal  $X$ -blocks have different left neighbors in  $\mathbb{A}_i$  and after the replacement these  $X$ -blocks share the same left neighbor in  $\mathbb{A}_{i+1}$ . This yields  $g_{i+1}(X) \leq g_i(X)$ .

Assume now that a nonunary, maximal string  $\gamma = vX$  is selected in round  $i + 1$  for some word  $v \in (N_i \cup \{X_0\})^+$ . We show that  $g_{i+1}(X) < g_i(X)$ . If  $X$  only occurs in the new rule in  $\mathbb{A}_{i+1}$ , i.e.,  $X$  does not occur in the modified rules, then we have  $\#_{i+1}(X) < \#_i(X)$  because at least two right-hand sides do not contain  $X$  as a factor anymore while only the new rule  $Y \rightarrow \gamma$  adds a new right-hand side which contains  $X$  to  $\#_{i+1}(X)$ . Moreover, we have  $L_{i+1}(X) = L_i(X)$  in this case because all rules where the maximal  $X$ -block is removed shared the same left neighbor since  $\gamma = vX$  is the selected nonunary, maximal string. However,  $vX$  still occurs on the right-hand side of the new rule of  $\mathbb{A}_{i+1}$ . It follows that  $g_{i+1}(X) < g_i(X)$ . If otherwise at least one of the modified rules still contains  $X$  on the right-hand side, then  $YX$  is a factor of this right-hand side in  $\mathbb{A}_{i+1}$  after the replacement of  $\gamma$ . It follows that  $L_{i+1}(X) > L_i(X)$  and thus  $g_{i+1}(X) < g_i(X)$  because each distinct left neighbor of  $X$  in  $\mathbb{A}_i$  is still a left neighbor of  $X$  in  $\mathbb{A}_{i+1}$  for some occurrence of  $X$ . Additionally,  $Y$  is a new left neighbor.  $\square$

In the following proof, we use the notation

$$U_i = \{X \in N_i \cup \{X_0\} \mid X \text{ is a unary nonterminal}\}$$

for all unary nonterminals that appear in  $\mathbb{A}_i$  and  $M_i = N_i \setminus U_i$  for all nonunary nonterminals that appear in  $\mathbb{A}_i$ .

**Lemma 12.** *We have*

$$\sum_{X \in M_i} t_i(X) \leq \sum_{X \in U_i} (\text{level}_i(X) + 1) \cdot (\text{level}_i(X) + 1 + (\text{level}_i(X) + 1)^2).$$

**Proof.** Let  $s(i) = \sum_{X \in M_i} t_i(X)$  be the total size that all nonunary nonterminals contribute to the size of  $\mathbb{A}_i$ . We first bound the number of rounds where the function  $s$  increases, i.e., we bound  $|\{j \in [0, i - 1] \mid s(j + 1) > s(j)\}|$ . If a unary nonterminal is introduced in some round  $j + 1$ , then  $s(j + 1) = s(j)$ , i.e., we can ignore these rules. So, consider some round  $j + 1$  where a nonunary nonterminal  $X$  is introduced and let  $X \rightarrow \gamma$  be the introduced rule. Let  $M = \{Z \in N_j \mid Z \text{ occurs in } \gamma\}$  be the set of nonterminals that occur at least once in  $\gamma$ . We first show that if  $k := |M_j \cap M| \geq 2$ , then  $s(j + 1) \leq s(j)$ . In other words, if two nonunary nonterminals occur in  $\gamma$ , then  $s(j + 1) \leq s(j)$ . Let  $r$  be the number of rules  $(Z \rightarrow v) \in P_j$  such that  $\gamma$  is a factor of  $v$ . Recall that nonunary factors and nonterminals occur at most once on the right-hand side of a single rule (Lemma 3). We have  $s(j + 1) - s(j) = k + r - k \cdot r$  because the new nonunary nonterminal  $X$  occurs now on  $r$  right-hand sides,  $\gamma$  contains  $k$  nonunary nonterminals which occur exactly once in  $\gamma$  each, and the replacement of  $\gamma$  on right-hand sides deletes these  $k$  nonterminals on  $r$  right-hand sides. We have  $r \geq 2$  (due to the properties of a maximal string) which together with  $k \geq 2$  yields  $s(j + 1) - s(j) \leq 0$ . Hence we can assume that  $k = |M_j \cap M| \leq 1$ . The maximal string  $\gamma$  has length  $|\gamma| \geq 2$  and is not unary, so the first and the last symbol of  $\gamma$  are different and at least one of them is unary due to our assumption that at most one nonunary nonterminal occurs in  $\gamma$ . Let  $Y$  be this unary nonterminal and assume that  $Y$  is the first symbol, i.e., the nonunary, maximal string is  $\gamma = Yv$  for some (nonempty)  $v$ . Afterwards we discuss the case where  $Y$  is the last symbol, i.e.,  $\gamma = vY$ .

We bound the number of rounds where a nonunary, maximal string  $\gamma = Yv$  is selected for some  $v$ . Let  $j_0 \leq i$  be the round where the unary nonterminal  $Y$  has been introduced. We have

$$f_{j_0}(Y) \leq \#_{j_0}(Y) \leq \text{level}_{j_0}(Y) + 1 \leq \text{level}_i(Y) + 1$$

due to Lemma 6 and  $\text{level}_j(Y) \leq \text{level}_i(Y)$  for all  $j \leq i$ . We also have  $f_{j+1}(Y) \leq f_j(Y)$  for  $j \in [j_0, i - 1]$  by Lemma 10. In addition to that, if  $Yv$  is the selected nonunary, maximal string in round  $j + 1$  for some  $v$ , then we have  $f_{j+1}(Y) < f_j(Y)$  again by Lemma 10. It follows that after at most  $\text{level}_i(Y) + 1$  many rounds where the chosen maximal string is nonunary and has the form  $Yv$  for some (nonempty)  $v$ , we have  $f_i(Y) = 0$ . In this case, all maximal  $Y$ -blocks have distinct right neighbors or occur at the end of a right-hand side. Hence there is no possibility to select a nonunary, maximal string  $Yv$  anymore.

Now we similarly bound the number of rounds such that a nonunary, maximal string  $\gamma = vY$  is selected for some  $v$ . However, care must be taken in this case, because it is possible that  $g_{j+1}(Y) > g_j(Y)$  when a rule  $X' \rightarrow Y^d$  for  $d \geq 2$  is introduced in round  $j + 1$  as explained above. Fortunately, rules of this form (the selected maximal string is from  $Y^+$ ) are introduced at most  $\text{level}_i(Y)$  many times up to round  $i$  by the definition of  $\text{level}_i(Y)$ . Let  $j_0 \leq i$  be the round where the unary nonterminal  $Y$  has been introduced. We have

$$g_j(Y) \leq \#_j(Y) \leq \text{level}_j(Y) + 1 \leq \text{level}_i(Y) + 1$$

for each  $j \in [j_0, i]$  due to Lemma 6 and  $\text{level}_j(Y) \leq \text{level}_i(Y)$  for all  $j \leq i$ . Furthermore, if the selected maximal string in round  $j + 1$  ( $j \geq j_0$ ) is not from  $Y^+$ , we have  $g_{j+1}(Y) \leq g_j(Y)$  due to Lemma 11. Moreover, if the maximal string  $\gamma$  is nonunary and  $\gamma = vY$  for some (nonempty)  $v$ , then  $g_{j+1}(Y) < g_j(Y)$ . It follows that between two rounds where maximal strings from  $Y^+$  are selected, there are at most  $\text{level}_i(Y) + 1$  many rounds where a nonunary, maximal string of the form  $vY$  is chosen because then  $g_j(Y) = 0$  is reached (for some  $j$ ) and thus all maximal  $Y$  blocks have distinct left neighbors or occur at the beginning of a right-hand side. Hence no nonunary string of the form  $vY$  for some  $v$  occurs twice on right-hand sides. Since maximal strings from  $Y^+$  are chosen at most  $\text{level}_i(Y)$  many times

up to round  $i$ , it follows that the number of rounds where a nonunary, maximal string of the form  $vY$  for some  $v$  is selected is at most  $(\text{level}_i(Y) + 1)^2$ .

Furthermore, the maximal increase  $\max\{s(j + 1) - s(j) \mid j \in [0, i - 1]\}$  in a single round is at most  $\text{level}_i(Y) + 1$ , because the new nonunary nonterminal occurs in  $\mathbb{A}_{j+1}$  on at most  $\#_j(Y) \leq \text{level}_j(Y) + 1 \leq \text{level}_i(Y) + 1$  many right-hand sides of rules for any  $j \leq i$  and the total number of occurrences of all other (nonunary) nonterminals does not increase (Lemma 7, point (1)).

We conclude that for each unary nonterminal  $Y$ , at most

$$\text{level}_i(Y) + 1 + (\text{level}_i(Y) + 1)^2$$

many rules are introduced such that the nonunary, maximal string  $\gamma$  satisfies  $\gamma = Yv$  or  $\gamma = vY$  for some  $v$  and each of those rules increases the total size that nonunary nonterminals contribute by at most  $\text{level}_i(Y) + 1$ . In all other cases, we showed that the size that nonunary nonterminals contribute does not increase.  $\square$

Now we are able to prove Proposition 1.

**Proof of Proposition 1.** Let  $\mathbb{A}_f = \text{Greedy}(X_0^n)$  be the final SLP obtained by Greedy, i.e., after  $f$  rounds the algorithm stops because  $\mathbb{A}_f$  has no maximal string. First, we want to bound the level of unary nonterminals occurring in  $\mathbb{A}_f$ . Assume there is a unary nonterminal  $X$  such that  $\text{level}_i(X) = \lceil \log \log n \rceil$  after some round  $i \leq f$  of the algorithm. By Corollary 2, we have

$$t_i(X) \leq 4n^{2^{-\log \log n}} (\log \log n + 3) \leq 8(\log \log n + 3).$$

Consider the unique leaf node  $v_X$  in the tree  $T_i$  which has level  $\lceil \log \log n \rceil$  and label  $(X, k)$  for some  $k$ . If in some round  $j \in [i + 1, f]$  two children with labels  $(X, k + 1)$  and  $(Y, k)$  are attached to  $v_X$ , i.e., the introduced rule in round  $j$  is  $Y \rightarrow X^d$  for some  $d \geq 2$ , then we have  $t_j(X) + t_j(Y) \leq t_{j-1}(X) \leq t_i(X)$  by Lemma 7. To be more specific, if the length of the chosen maximal string  $X^d$  is exactly  $d = 2$  and this maximal string  $XX$  occurs exactly twice without overlap in  $\mathbb{A}_{j-1}$ , then we have  $t_j(X) + t_j(Y) = t_{j-1}(X)$  (and  $|\mathbb{A}_j| = |\mathbb{A}_{j-1}|$ ). Please note that in this case, there does not exist a maximal string  $X^d$  or  $Y^d$  of  $\mathbb{A}_k$  for all  $k \in [j, f]$  (since  $Y$  occurs only twice and  $XX$  does not occur on right-hand sides of  $\mathbb{A}_j$ ), i.e., the children of the node  $v_X$  in  $T_k$  are leaves for  $k \in [j, f]$ . Otherwise, if the maximal string has length  $d \geq 3$  or occurs at least three times without overlap, then we have  $t_j(X) + t_j(Y) < t_{j-1}(X)$  since  $|\mathbb{A}_j| < |\mathbb{A}_{j-1}|$  holds in this setting. This means that when a new branch occurs in the tree  $T_j$  for some  $j$ , then the new children of the branching node are either leaves of the final tree  $T_f$  or the corresponding nonterminals contribute strictly less to the size of the current SLP than the nonterminal which corresponds to the parent node did before the branch. We can iterate this argument for the children of the children of  $v_X$  and so on, i.e., if we consider the subtree rooted at  $v_X$  in  $T_f$ , then from level to level the size that the nonterminals contribute decreases until only leaves occur at some level. Since  $t_i(X) \leq 8(\log \log n + 3)$ , it follows that the subtree of  $T_f$  rooted at  $v_X$  has depth at most  $8(\log \log n + 3) + 1$  and thus the maximal level of any unary nonterminal in  $\mathbb{A}_f$  is bounded by

$$\lceil \log \log n \rceil + 8(\log \log n + 3) + 1 \leq 9 \log \log n + 26.$$

Consequently, the number of unary nonterminals (the number of leaves of  $T_f$ ) is bounded by  $\mathcal{O}((\log n)^9)$  since  $T_f$  is a binary tree of depth at most  $9 \log \log n + 26$ . Furthermore, each unary nonterminal  $X$  in  $\mathbb{A}_f$  satisfies  $t_f(X) \leq \mathcal{O}(\log \log n)$ . If there is a round  $i \leq f$  such that  $\text{level}_i(X) = \lceil \log \log n \rceil$  we obtain  $t_f(X) \leq t_i(X) \leq 8(\log \log n + 3)$  by Corollary 2 and Lemma 7, point (1). Otherwise, let  $m = \text{level}_f(X) < \lceil \log \log n \rceil$ . Then  $t_f(X) \leq m + 3 \leq \log \log n + 3$ , because there is at most one nonoverlapping occurrence of  $XX$  on right-hand sides of  $\mathbb{A}_f$  (otherwise there would exist a maximal string of  $\mathbb{A}_f$ ) and the number of rules where  $X$  occurs on the right-hand side is  $\#_f(X) \leq m + 1$  by

**Lemma 6.** To be more precise, a single right-hand side of  $\mathbb{A}_f$  could have a maximal  $X$ -block of length 3 and all other right-hand sides must have at most one occurrence of  $X$  since two different right-hand sides where  $X$ -blocks of length 2 occur as well as one right-hand side where an  $X$ -block of length 4 occurs would contradict the fact that  $\mathbb{A}_f$  has no maximal string. It follows that the size which unary nonterminals contribute to  $\mathbb{A}_f$  is  $\mathcal{O}((\log n)^9 \cdot \log \log n)$ . By Lemma 12, we can bound the size that nonunary nonterminals contribute by  $\mathcal{O}((\log n)^9 \cdot (\log \log n)^3)$  since there are at most  $\mathcal{O}((\log n)^9)$  many unary nonterminals and each has level at most  $\mathcal{O}(\log \log n)$  as argued above. It follows that  $|\mathbb{A}_f| \leq \mathcal{O}((\log n)^9 \cdot (\log \log n)^3)$ , which proves the proposition.  $\square$

The following theorem follows directly from Proposition 1 and Lemma 1, where  $g(w) \geq \Omega(\log n)$  is shown for words  $w$  of length  $n$ .

**Theorem 1.** For all  $n$ , we have

$$\alpha_{\text{Greedy}}(1, n) \leq \mathcal{O}((\log n)^8 \cdot (\log \log n)^3).$$

#### 4. Lower Bound for Greedy

We proceed with the lower bound on the approximation ratio of Greedy. The best lower bound [13] (Theorem 11) that was known previously was

$$\alpha_{\text{Greedy}}(k, n) \geq \frac{5}{3 \log_3(5)} = 1.13767699 \dots$$

for all  $k \geq 1$  and infinitely many  $n$ . This bound was shown using unary words, which we will also use in our proof for the improved lower bound. A key concept for doing this is the sequence  $x_n$  described in the following lemma by [19]:

**Lemma 13** ([19] (Example 2.2)). Let  $x_{n+1} = x_n^2 + 1$  with  $x_0 = 1$  and

$$\beta = \exp\left(\sum_{i=1}^{\infty} \frac{1}{2^i} \log\left(1 + \frac{1}{x_i^2}\right)\right).$$

We have  $x_n = \lfloor \beta^{2^n} \rfloor$ .

In this work, we use the shifted sequence  $y_n = x_{n+1}$ , i.e., we start with  $y_0 = 2$ . It follows that  $y_n = \lfloor \gamma^{2^n} \rfloor$ , where  $\gamma = \beta^2 = 2.25851845 \dots$ . Additionally, we need the following lemma:

**Lemma 14.** Let  $m \geq 1$  be an integer. Let  $f_m: \mathbb{R}_{>0} \rightarrow \mathbb{R}$  with

$$f_m(x) = x + \frac{m^2 + 1}{x}.$$

We have  $f_m(x) > 2m$  for all  $x > 0$ .

**Proof.** The unique minimum of  $f_m(x)$  is  $2\sqrt{m^2 + 1}$  for  $x = \sqrt{m^2 + 1}$ . It follows that  $f_m(x) \geq 2\sqrt{m^2 + 1} > 2\sqrt{m^2} = 2m$ .  $\square$

Now we can prove the new lower bound for Greedy:

**Theorem 2.** For all  $k \geq 1$  and infinitely many  $n$ , we have

$$\alpha_{\text{Greedy}}(k, n) \geq \frac{1}{\log_3(\gamma)} = 1.34847194 \dots$$

**Proof.** Let  $\Sigma = \{a\}$  be a unary alphabet. We define  $w_k = a^{y_k}$ . By Lemma 13, we have  $|w_k| \leq \gamma^{2^k}$ . Applying Lemma 1 yields

$$g(w_k) \leq 3 \cdot \log_3(\gamma) \cdot 2^k + o(2^k).$$

In the remaining proof we show that on input  $w_k$ , Greedy produces an SLP of size  $3 \cdot 2^k - 1$ , which directly implies  $\alpha_{\text{Greedy}}(1, n) \geq 3/(3 \log_3(\gamma))$ . We start with the SLP  $\mathbb{A}_0$  which has the single rule  $S \rightarrow a^{y_k}$ . Consider now the first round of the algorithm, i.e., we need to find a maximal string  $a^x$  of  $\mathbb{A}_0$  such that the grammar  $\mathbb{A}_1$  with rules

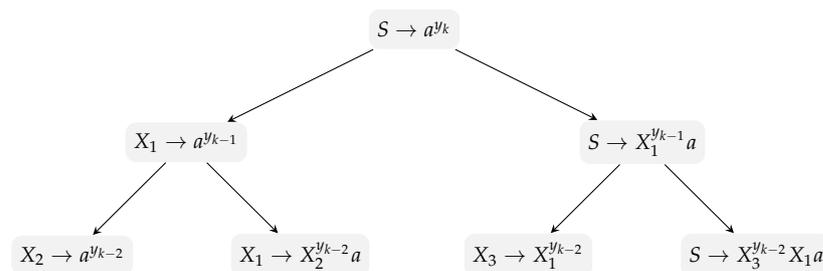
$$X_1 \rightarrow a^x, S \rightarrow X_1^{y_k \operatorname{div} x} a^{y_k \operatorname{mod} x}$$

has minimal size. We have  $|\mathbb{A}_1| = x + (y_k \operatorname{div} x) + (y_k \operatorname{mod} x) \geq x + y_k/x$ . By the definition of  $y_k$  we have  $|\mathbb{A}_1| \geq x + (y_{k-1}^2 + 1)/x$ . Applying Lemma 14 yields  $|\mathbb{A}_1| \geq 2y_{k-1} + 1$ . Please note that for  $x = y_{k-1}$  this minimum is achieved, i.e., we can assume that Greedy selects the maximal string  $a^{y_{k-1}}$  and  $\mathbb{A}_1$  is

$$X_1 \rightarrow a^{y_{k-1}}, S \rightarrow X_1^{y_{k-1}} a.$$

Each maximal string of  $\mathbb{A}_1$  is either a unary word over  $X$  or a unary word over  $a$ , i.e., we can analyze the behavior of Greedy on both rules independently. The rule  $X_1 \rightarrow a^{y_{k-1}}$  is obviously treated similarly as the initial SLP  $\mathbb{A}_0$ , so we continue with analyzing  $S \rightarrow X_1^{y_{k-1}} a$ . However, again, the same arguments as above show that Greedy introduces a rule  $X_3 \rightarrow X_1^{y_{k-2}}$  which yields  $S \rightarrow X_3^{y_{k-2}} X_1 a$  as the new start rule. This process can be iterated using the same arguments for the leading unary strings of length  $y_i$  for some  $i \in [1, k]$ .

The reader might think of this process as a binary tree, where each node is labeled with a rule (the root is labeled with  $S \rightarrow a^{y_k}$ ) and the children of a node are the two rules obtained by Greedy when the rule has been processed. We assume that the left child represents the rule for the chosen maximal string and the right child represents the parent rule where all occurrences of the maximal string are replaced by the new nonterminal. In Figure 2 this binary tree is depicted for the steps we discussed above.



**Figure 2.** Three rounds of Greedy on input  $a^{y_k}$ .

Please note that when a rule is processed, the longest common factor of the two new rules has length 1 (the remainder). More generally, after each round there is no word of length at least two that occurs as a factor in two different rules, since a possibly shared remainder has length 1 and otherwise only new nonterminals are introduced. It follows that we can iterate this process independently for each rule until no maximal string occurs. This is the case when each rule starts with a unary string of length  $y_0 = 2$  or, in terms of the interpretation as a binary tree, when a full binary tree of height  $k$  is produced. Each right branch occurring in this tree adds a new remainder to those remainders that already occur in the parent rule and a left branch introduces a new (smaller) instance of the start problem. We show by induction that at level  $i \in [0, k]$  of this full binary tree of height  $k$ , there is one rule of size  $y_{k-i} + i$  and  $2^{i-j-1}$  many rules of size  $y_{k-i} + j$  for  $j \in [0, i - 1]$ . At level 0, this is true since there is only a single rule of size  $y_k + 0$ . Assuming that our claim is true at level

$i < k$ , we derive from each rule at level  $i$  two new rules at level  $i + 1$ : A right branch yields a rule that starts with a leading unary string of size  $y_{k-i-1}$  and adds a new remainder to the parent rule. A left branch yields a rule that contains only a unary string of size  $y_{k-i-1}$ . If we first consider the left branches, we derive that each of the  $2^i$  many rules at level  $i$  adds a rule of size  $y_{k-i-1}$  at level  $i + 1$ . For the right branches, the single rule of size  $y_{k-i} + i$  at level  $i$  yields a rule of size  $y_{k-i-1} + i + 1$  at level  $i + 1$ . Furthermore, each of the  $2^{i-j-1}$  many rules of size  $y_{k-i} + j$  ( $j \in [0, i - 1]$ ) yields a rule of size  $y_{k-i-1} + j + 1$ . When we put everything together, we get that at level  $i + 1$  there is a single rule of size  $y_{k-i-1} + i + 1$  and  $2^{i-j}$  many rules of size  $y_{k-i-1} + j$  for  $j \in [0, i]$ . That finishes the induction. It follows that the final SLP (which consists of the rules at level  $k$ ) has a single rule of size  $y_0 + k = 2 + k$  and  $2^{k-j-1}$  many rules of size  $2 + j$  for  $j = 0, \dots, k - 1$ . This gives a total size of

$$\begin{aligned}
 2 + k + \sum_{j=0}^{k-1} 2^{k-j-1}(2 + j) &= 2 + k + 2^k \sum_{j=0}^{k-1} 2^{-j} + 2^k \sum_{j=0}^{k-1} 2^{-j-1}j \\
 &= 2 + k + 2^k(2 - 2^{-k+1}) + 2^k(-2^{-k}k - 2^{-k} + 1) \\
 &= 2 + k + 2^{k+1} - 2 - k - 1 + 2^k \\
 &= 2^{k+1} + 2^k - 1 \\
 &= 3 \cdot 2^k - 1.
 \end{aligned}$$

□

## 5. RePair and LongestMatch

In this section, we analyze the global grammar-based compressors RePair and LongestMatch. In each round  $i$ , RePair selects a most frequent maximal string of  $\mathbb{A}_{i-1}$  and LongestMatch selects a longest maximal string of  $\mathbb{A}_{i-1}$ .

We will abbreviate the approximation ratio  $\alpha_{\text{LongestMatch}}$  by  $\alpha_{\text{LM}}$  for better readability. We will first show that RePair and LongestMatch produce SLPs of equal size for unary inputs  $a^n$  and we show what the exact size of these SLPs depending on  $n$  is. We then use this information to obtain our result for  $\alpha_{\text{RePair}}(1, n)$  and  $\alpha_{\text{LM}}(1, n)$ , respectively. Fix an integer  $n \geq 2$  and consider the binary representation

$$n = \sum_{i=0}^{\lfloor \log_2 n \rfloor} b_i \cdot 2^i \quad (2)$$

of  $n$ , where  $b_i \in \{0, 1\}$  for  $i \in [0, \lfloor \log_2 n \rfloor]$ . We denote by  $\nu(n)$  the number of 1's in the binary representation of  $n$ , i.e.,

$$\nu(n) = \sum_{i=0}^{\lfloor \log_2 n \rfloor} b_i.$$

For example, we have  $11 = 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$  and thus  $b_0 = b_1 = b_3 = 1$ ,  $b_2 = 0$  and  $\nu(11) = 3$ .

**Proposition 2.** For  $n \geq 2$ , let  $\mathbb{A}$  be the SLP produced by RePair on input  $a^n$  and  $\mathbb{B}$  be the SLP produced by LongestMatch on input  $a^n$ . We have

$$|\mathbb{A}| = |\mathbb{B}| = 2 \lfloor \log_2 n \rfloor + \nu(n) - 1.$$

**Proof.** If  $n = 2$  or  $n = 3$  then  $a^n$  has no maximal string and thus the final SLP of any global algorithm has a single rule  $S \rightarrow a^n$ . The reader can easily verify the claimed result for these cases.

We now assume that  $n \geq 4$ . Let  $m = \lfloor \log_2 n \rfloor - 1$ . We prove the claim for RePair first and for LongestMatch after. On input  $a^n$ , RePair runs for exactly  $m$  rounds and creates rules

$X_1 \rightarrow aa$  and  $X_i \rightarrow X_{i-1}X_{i-1}$  for  $i \in [2, m]$ , i.e., the nonterminal  $X_i$  produces the string  $a^{2^i}$ . These rules have a total size of  $2m$ . After these steps, the start rule is

$$S \rightarrow X_m X_m X_m^{b_m} X_{m-1}^{b_{m-1}} \dots X_1^{b_1} a^{b_0},$$

where the  $b_i$ 's are the coefficients occurring in the binary representation of  $n$ , see equation (2). In other words, the symbol  $a$  only occurs in the start rule if the least significant bit is  $b_0 = 1$ , and the nonterminal  $X_i$  ( $i \in [1, m - 1]$ ) occurs in the start rule if and only if  $b_i = 1$ . Since RePair only replaces words with at least two occurrences, the most significant bit  $b_{m+1} = 1$  is represented by  $X_m X_m$ . A third  $X_m$  occurs in the start rule if and only if  $b_m = 1$ . The size of the start rule is  $2 + \sum_{i=0}^m b_i$ . It follows that the total size of the SLP produced by RePair on input  $a^n$  is  $2m + 2 + \sum_{i=0}^m b_i$ , which together with  $m = \lfloor \log_2 n \rfloor - 1$  and  $b_{\lfloor \log_2 n \rfloor} = 1$  (the most significant bit is always 1) yields the claimed size.

Now we prove the same result for LongestMatch. In the first round, the chosen maximal string is  $a^{\lfloor n/2 \rfloor}$ , which yields rules  $X_1 \rightarrow a^{\lfloor n/2 \rfloor}$  and  $S \rightarrow X_1 X_1 a^{b_0}$ , i.e., the symbol  $a$  occurs in the start rule if and only if  $n$  is odd and thus the least significant bit is  $b_0 = 1$ . Assuming that  $n \geq 8$ , this procedure is now repeated for the rule  $X_1 \rightarrow a^{\lfloor n/2 \rfloor}$  (for  $n < 8$  there is no maximal string, and the algorithm stops after the first round). This yields  $X_2 \rightarrow a^{\lfloor n/4 \rfloor}$ ,  $X_1 \rightarrow X_2 X_2 a^{b_1}$  and  $S \rightarrow X_1 X_1 a^{b_0}$  (note that  $\lfloor (\lfloor n/2 \rfloor)/2 \rfloor = \lfloor n/4 \rfloor$ ). After  $m = \lfloor \log_2 n \rfloor - 1$  steps, the iteration of this process results in the final SLP with rules  $S \rightarrow X_1 X_1 a^{b_0}$ ,  $X_i \rightarrow X_{i+1} X_{i+1} a^{b_i}$  for  $i \in [1, m - 1]$  and  $X_m \rightarrow a a^{b_m}$ . The size of this SLP is  $2 \cdot (m + 1) + \sum_{i=0}^m b_i$ , which directly implies the claimed result for LongestMatch.  $\square$

Using Proposition 2, we prove the matching bounds for  $\alpha_{\text{RePair}}(1, n)$  and  $\alpha_{\text{LM}}(1, n)$ :

**Theorem 3.** For all  $n$ , we have  $\alpha_{\text{RePair}}(1, n) = \alpha_{\text{LM}}(1, n) \leq \log_2(3)$ .

**Proof.** As a consequence of Proposition 2, RePair and LongestMatch produce on input  $a^n$  SLPs of size at most  $3 \log_2 n$ , since  $\nu(n) - 1 \leq \log_2 n$ . By Lemma 1, we have  $g(a^n) \geq 3 \log_3 n - 3$ . The equality  $\log_2 n / \log_3 n = \log_2(3)$  finishes the proof.  $\square$

**Theorem 4.** For infinitely many  $n$ , we have  $\alpha_{\text{RePair}}(1, n) = \alpha_{\text{LM}}(1, n) \geq \log_2(3)$ .

**Proof.** Let  $w_k = a^{2^k - 1}$ . We have  $2^k - 1 = \sum_{i=0}^{k-1} 2^i$  and thus  $\nu(2^k - 1) = k$ . By Proposition 2, the size of the SLPs produced by RePair and LongestMatch is  $3k - 3$ . By Lemma 1, we have

$$g(w_k) \leq 3 \log_3(2^k - 1) + o(\log(2^k - 1)) \leq 3 \log_3(2) \cdot k + o(k).$$

The equality  $1 / \log_3(2) = \log_2(3)$  finishes the proof.  $\square$

## 6. Discussion

Although we improved the upper bound on the approximation ration for Greedy, the lower bound remains quite weak. We might be able to improve it by finding a similar sequence such that Greedy produces larger remainders in each round. However, care must be taken since for larger remainders it is not true anymore that the rules can be analyzed independently because the rules could share factors of length greater than one. Concerning the upper bound, we conjecture that Greedy achieves logarithmic compression for all unary inputs and thus the approximation ratio is constant, but we have not been able to prove this so far. For arbitrary alphabets, a nonconstant lower bound for Greedy as well as an improvement of the upper bound of  $\mathcal{O}((n / \log n)^{2/3})$  for any global algorithm seems to be natural starting points for future work.

**Author Contributions:** Conceptualization, D.H.; methodology, D.H.; validation, D.H. and C.P.R.; formal analysis, D.H.; writing—original draft preparation, D.H.; writing—review and editing, C.P.R. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by the DFG research project LO 748/10-2.

**Data Availability Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

## References

1. Ziv, J.; Lempel, A. Compression of individual sequences via variable-rate coding. *IEEE Trans. Inf. Theory* **1978**, *24*, 530–536. [[CrossRef](#)]
2. Kieffer, J.C.; Yang, E.; Nelson, G.J.; Cosman, P.C. Universal lossless compression via multilevel pattern matching. *IEEE Trans. Inf. Theory* **2000**, *46*, 1227–1245. [[CrossRef](#)]
3. Nevill-Manning, C.G.; Witten, I.H. Identifying Hierarchical Structure in Sequences: A linear-time algorithm. *CoRR* **1997**, *7*, 67–82. [[CrossRef](#)]
4. Apostolico, A.; Lonardi, S. Some Theory and Practice of Greedy Off-Line Textual Substitution. In Proceedings of the Data Compression Conference, DCC 1998, IEEE Computer Society, Snowbird, UH, USA, 30 March–1 April 1998; pp. 119–128. [[CrossRef](#)]
5. Apostolico, A.; Lonardi, S. Compression of Biological Sequences by Greedy Off-Line Textual Substitution. In Proceedings of the Data Compression Conference, DCC 2000, IEEE Computer Society, Snowbird, UH, USA, 28–30 March 2000; pp. 143–152. [[CrossRef](#)]
6. Apostolico, A.; Lonardi, S. Off-line compression by greedy textual substitution. *Proc. IEEE* **2000**, *88*, 1733–1744. [[CrossRef](#)]
7. Larsson, N.J.; Moffat, A. Offline Dictionary-Based Compression. In Proceedings of the Data Compression Conference, DCC 1999, IEEE Computer Society, Snowbird, UH, USA, 29–31 March 1999; pp. 296–305. [[CrossRef](#)]
8. Kieffer, J.C.; Yang, E. Grammar-based codes: A new class of universal lossless source codes. *IEEE Trans. Inf. Theory* **2000**, *46*, 737–754. [[CrossRef](#)]
9. Claude, F.; Navarro, G. Fast and Compact Web Graph Representations. *ACM Trans. Web* **2010**, *4*, 16:1–16:31. [[CrossRef](#)]
10. Kida, T.; Matsumoto, T.; Shibata, Y.; Takeda, M.; Shinohara, A.; Arikawa, S. Collage system: a unifying framework for compressed pattern matching. *Theor. Comput. Sci.* **2003**, *298*, 253–272. [[CrossRef](#)]
11. González, R.; Navarro, G. Compressed Text Indexes with Fast Locate. In Proceedings of the Combinatorial Pattern Matching, 18th Annual Symposium, CPM 2007, London, ON, Canada, 9–11 July 2007; In *Lecture Notes in Computer Science*; Ma, B., Zhang, K., Eds.; Springer: Berlin, Germany, 2007; Volume 4580, pp. 216–227. [[CrossRef](#)]
12. Lohrey, M.; Maneth, S.; Mennicke, R. XML tree structure compression using RePair. *Inf. Syst.* **2013**, *38*, 1150–1167. [[CrossRef](#)]
13. Charikar, M.; Lehman, E.; Liu, D.; Panigrahy, R.; Prabhakaran, M.; Sahai, A.; Shelat, A. The smallest grammar problem. *IEEE Trans. Inf. Theory* **2005**, *51*, 2554–2576. [[CrossRef](#)]
14. Hucke, D.; Lohrey, M.; Reh, C.P. The Smallest Grammar Problem Revisited. In Proceedings of the String Processing and Information Retrieval—23rd International Symposium, SPIRE 2016, Beppu, Japan, 18–20 October 2016; In *Lecture Notes in Computer Science*; Inenaga, S., Sadakane, K., Sakai, T., Eds.; Springer International Publishing: Cham, Switzerland, 2016, Volume 9954, pp. 35–49. [[CrossRef](#)]
15. Hucke, D.; Jez, A.; Lohrey, M. Approximation ratio of RePair. *arXiv* **2017**, arXiv:1703.06061.
16. Knuth, D.E. *The Art of Computer Programming, Volume II: Seminumerical Algorithms*, 3rd ed.; Addison-Wesley: Boston, MA, USA, 1998.
17. Noma, A.M.; Muhammed, A.; Mohamed, M.A.; Zulkarnain, Z.A. A Review on Heuristics for Addition Chain Problem: Towards Efficient Public Key Cryptosystems. *J. Comput. Sci.* **2017**, *13*, 275–289. [[CrossRef](#)]
18. Hucke, D. Approximation Ratios of RePair, LongestMatch and Greedy on Unary Strings. In Proceedings of the String Processing and Information Retrieval—26th International Symposium, SPIRE 2019, Segovia, Spain, 7–9 October 2019. In *Lecture Notes in Computer Science*; Brisaboa, N.R., Puglisi, S.J., Eds.; Springer: Berlin, Germany, 2019; Volume 11811, pp. 3–15. [[CrossRef](#)]
19. Aho, A.V.; Sloane, N.J.A. Some doubly exponential sequences. *Fibonacci Quart.* **1973**, *11*, 429–437.
20. Berstel, J.; Brlek, S. On the Length of Word Chains. *Inf. Process. Lett.* **1987**, *26*, 23–28. [[CrossRef](#)]
21. Diwan, A.A. *A New Combinatorial Complexity Measure for Languages*; Tata Institute: Bombay, India, 1986.
22. Storer, J.A.; Szymanski, T.G. Data compression via textual substitution. *J. ACM* **1982**, *29*, 928–951. [[CrossRef](#)]
23. Casel, K.; Fernau, H.; Gaspers, S.; Gras, B.; Schmid, M.L. On the Complexity of Grammar-Based Compression over Fixed Alphabets. In Proceedings of the 43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, Rome, Italy, 11–15 July 2016; Chatzigiannakis, I., Mitzenmacher, M., Rabani, Y., Sangiorgi, D., Eds.; Schloss Dagstuhl—Leibniz-Zentrum für Informatik, 2016, Volume 55, pp. 122:1–122:14. [[CrossRef](#)]
24. Ochoa, C.; Navarro, G. RePair and All Irreducible Grammars are Upper Bounded by High-Order Empirical Entropy. *IEEE Trans. Inf. Theory* **2019**, *65*, 3160–3164. [[CrossRef](#)]