On the Complexity of the Smallest Grammar Problem over Fixed Alphabets

Katrin Casel · Henning Fernau · Serge Gaspers · Benjamin Gras · Markus L. Schmid

Abstract In the smallest grammar problem, we are given a word w and we want to compute a preferably small context-free grammar G for the singleton language $\{w\}$ (where the size of a grammar is the sum of the sizes of its rules, and the size of a rule is measured by the length of its right side). It is known that, for unbounded alphabets, the decision variant of this problem is NP-hard and the optimisation variant does not allow a polynomial-time approximation scheme, unless P = NP. We settle the long-standing open problem whether these hardness results also hold for the more realistic case of a constant-size alphabet. More precisely, it is shown that the smallest grammar problem remains NP-complete (and its optimisation version is APX-hard), even if the alphabet is fixed and has size of at least 17. The corresponding reduction is

This work represents an extended version of the paper "On the Complexity of Grammar-Based Compression over Fixed Alphabets" presented at the 43^{rd} *International Colloquium on Automata, Languages, and Programming* (ICALP 2016) and published in *LIPIcs - Leibniz International Proceedings in Informatics* (http://dx.doi.org/10.4230/LIPIcs.ICALP.2016.122)

Katrin Casel
Hasso Plattner Institute, University of Potsdam, Potsdam, Germany
E-mail: Casel@informatik.uni-trier.de

Henning Fernau
Trier University, Fachbereich IV – Abteilung Informatikwissenschaften, Trier, D-54296, Germany
E-mail: Fernau@uni-trier.de

Serge Gaspers
UNSW Australia, Data61 (formerly: NICTA), CSIRO, Sydney, Australia
E-mail: sergeg@cse.unsw.edu.au

Benjamin Gras
École Normale Superieure de Lyon, Département Informatique, Lyon, France
E-mail: benjamin.gras@ens-lyon.fr

Markus L. Schmid
Humboldt-Universität zu Berlin, Unter den Linden 6, D-10099 Berlin, Germany,
E-mail: MLSchmid@MLSchmid.de

robust in the sense that it also works for an alternative size-measure of grammars that is commonly used in the literature (i. e., a size measure also taking the number of rules into account), and it also allows to conclude that even computing the number of rules required by a smallest grammar is a hard problem. On the other hand, if the number of nonterminals (or, equivalently, the number of rules) is bounded by a constant, then the smallest grammar problem can be solved in polynomial time, which is shown by encoding it as a problem on graphs with interval structure. However, treating the number of rules as a parameter (in terms of parameterised complexity) yields W[1]-hardness. Furthermore, we present an $\mathcal{O}(3^{|w|})$ exact exponential-time algorithm, based on dynamic programming. These three main questions are also investigated for 1-level grammars, i. e., grammars for which only the start rule contains nonterminals on the right side; thus, investigating the impact of the "hierarchical depth" of grammars on the complexity of the smallest grammar problem. In this regard, we obtain for 1-level grammars similar, but slightly stronger results.

Keywords Grammar-Based Compression · Smallest Grammar Problem · Straight-Line Programs · NP-Completeness · Exact Exponential-Time Algorithms

1 Introduction

Context-free grammars are among the most classical concepts in theoretical computer science. Their wide range of applications, both of theoretical and practical nature, is well-known and usually forms an integral part of academic undergraduate courses in computer science. In this paper, we are concerned with grammars G that describe singleton languages $\{w\}$ (or, by slightly abusing notation, grammars describing single words).[1]

1.1 Grammars as Inference Tools and Compressors

Although, from a formal languages point of view, describing a single word by a context-free grammar seems excessive, there are at least two evident motivations:

- *Compression Perspective*:[2] The grammar G is a *compressed representation* of the word w.
- *Inference Perspective*: The grammar G identifies the *hierarchical structure* of the word w.

The inference perspective can be traced back to the work of Nevill-Manning and Witten [50, 49],[3] in which the authors consider algorithmic possibilities

[1] Such context-free grammars are also called *straight-line programs* in the literature.

[2] In this work, the term "compression" always refers to lossless data compression.

[3] The work [49] also considers the compression perspective.

of extracting (hierarchical) structure from sequential data, such as texts (in a natural or formal language), music or DNA, by constructing a grammar for a given sequence. The hypothesis that *small* grammars are to be preferred can be considered as an application of Occam's razor (note that the size of a grammar is the sum of the sizes of its rules, where the size of a rule is measured by the length of its right side). In a more general sense, Nevill-Manning and Witten's approach embarks on the quest of inferring the intrinsic *information* content of a given sequence, which is a central problem in learning theory and algorithmic information theory (especially Kolmogorov complexity, as mentioned below). In Nevill-Manning's PhD-thesis [49], a multitude of connections between the compression perspective of computing grammars for single words and other core topics of mathematics and theoretical computer science are discussed (e. g., the minimum description length principle in learning theory, information theory, data compression). The inference perspective of computing grammars for single words has been applied in two more PhD-theses, namely by de Marcken [48] in order to investigate whether analysing the structure of small grammars for large English texts could help understanding the structure of the language itself, and by Gallé [23] in order to infer hierarchical structures in DNA. Moreover, Lanctot et al. [37] contribute to the work on estimating the entropy of DNA-sequences (see the references in [37]), by using an algorithm first proposed by Kieffer and Yang [35] to compute grammars for DNA-sequences.

While in the above mentioned work, grammars are mainly used as an inference tool, the obvious connections to data compression are often highlighted as well (e. g., in [49]). The work of Kieffer et al. [36,35,62] directly approaches the concept of representing words by grammars from a traditional data compression perspective, i. e., we want to compute a *small* grammar representing a *large* given word w (in the following, we denote the general concept of compressing a single word by a context-free grammar as *grammar-based compression*). Besides the above mentioned papers by Nevill-Manning and Witten, the work by Kieffer et al. is usually stated as the second origin of using grammars for single words, but a closer look into the older literature reveals that the *external pointer macro scheme* (*without overlapping and with pointer size* 1) defined by Storer and Szymanski [58,57] is also equivalent to grammar-based compression.

Another motivation is that grammar-based compression, like any lossless data compression scheme, provides a computable upper bound of the *Kolmogorov complexity* (see [40]). Since this central measure in algorithmic information theory is generally incomputable, such computable approximations are important and, in this regard, grammars are of relevance, since, in comparison to other practically applied compression schemes, they achieve high compression rates and therefore yield a better approximation of the Kolmogorov complexity (in this regard, note that many practically relevant compression schemes, e. g., some of the ones mentioned in Section 1.3, allow fast compression and decompression, but cannot achieve exponential compression rates).

1.2 Algorithmics on Compressed Strings

The original motivations outlined so far are still relevant, but the actual reason why grammar-based compression has experienced a renaissance and thrives today as an independent and important field of research on its own are the following. While in the early days of computer science, the most important requirements for compression schemes were fast (i. e., linear or near linear time) compression and decompression, nowadays the investigation regarding whether they are suitable for solving problems directly on the compressed data without prior decompression forms a vibrant research area.[4] This area is usually subsumed under the term *algorithmics on compressed strings*, and grammar-based compression is particularly well suited for this purpose.

The success of grammars with respect to algorithmics on compressed strings is due to the fact that they cover many compression schemes from practice (most notably, the family of Lempel-Ziv encodings) and that they are mathematically easy to handle (see Lohrey [41] for a survey on the role of grammar-based compression for algorithmics on compressed strings). Many basic problems on strings, e. g., comparison, pattern matching, membership in a regular language, retrieving subwords, etc. can all be solved in polynomial time directly on the grammars [41]. In addition, grammar-based compression has been successfully applied in combinatorial group theory (see the textbook [42] by Lohrey) and to prove problems in computational topology to be polynomial-time solvable [41]. Grammars as compression schemes have also been extended to more complicated objects, e. g., trees (see [1,43,44,45,28], and [27,28] for applications in term unification) and two-dimensional words (see [8]). It is also worth pointing out the successful applications of compression-techniques for solving word equations (see, e. g., [53,34]).

A rather recent result is that any context-free grammar for a single word can be transformed in linear time into an equivalent one that is balanced in the sense that the depth of its derivation tree is logarithmic in the size of the represented word (see [24]). This result has a direct impact on basic algorithmic problems on grammar-compressed data, e. g., the random access problem (i. e., accessing in the compressed string the symbol at a given position).

1.3 The Smallest Grammar Problem

For grammar-based compression, the central computational problem is that of computing a smallest (or at least small) grammar for a given word, which is called the *smallest grammar problem*,[5] and the respective literature is mainly

[4] There is a Dagstuhl seminar series concerned with algorithmics on compressed sequences that so far took place in 2008 [10], 2013 [46] and 2016 [9].

[5] A concept of *grammar complexity* has also been introduced and is investigated in the area of descriptional complexity of formal languages (see [3,20,12,18,30,32]). However, this differs from the topic of this paper, since there, grammars for finite languages are investigated and the complexity measure under interest is the number of rules (note that in [30,32], the size of grammars is also considered).

about approximation algorithms:[6] LZ78 [63], LZW [61], Bisection [36], Sequitur [49,50] and Sequential [62], Longest Match [35], Greedy [4], Re-Pair [38] (the names of algorithms in this list are according to [14,39]). These algorithms share the benefit of being rather simple and fast, and their approximation ratios have been studied thoroughly by Charikar et al. in [14], by Lehman in his PhD-thesis [39] and some bounds have recently been further improved by Hucke et al. [33]. Unfortunately, none of the approximation ratios are constant and the currently best achieved approximation ratio is $\mathcal{O}\left(\log\left(\frac{|w|}{m^*}\right)\right)$, where m^* is the size of a smallest grammar (i. e., it is still open whether an approximation algorithm with a constant approximation-ratio exists, or equivalently, whether the problem is in APX). This result is due to the algorithms by Rytter [54] and Charikar et al. [14,39], which have been developed independently from each other and are not mentioned in the above list. On the other hand, assuming $\mathsf{P} \neq \mathsf{NP}$, it has been shown in [14,39] that an approximation ratio better than $\frac{8569}{8568} \approx 1.0001$ is not possible (thus, ruling out a polynomial-time approximation scheme (PTAS)). However, the research seems to have stagnated at this huge gap between lower and upper bound and still neither an approximation algorithm with a constant approximation ratio nor stronger inapproximability results are known.

The strong bias towards approximation algorithms is usually justified by the general NP-hardness of the smallest grammar problem, but, as explained next, this theoretical justification is seriously flawed. The NP-completeness can be shown by a reduction from vertex cover (see [14,39]), but in the reduction, an unbounded number of symbols in the underlying alphabet is needed. This means nothing less than that the hardness-reduction is invalid for any realistic scenario, where we deal with a constant alphabet (even more, if the alphabet is rather small, as it is the case in practical applications). Consequently, since the motivation for the approximation algorithms mentioned above is of a rather practical kind (i. e., string compression in real-world scenarios), this theoretical foundation falls apart (in particular, note that an unbounded alphabet is also necessary for the inapproximability result of [14,39]). One reason for this situation is probably that in [5], it is claimed that the hardness for alphabets of size 3 follows from [57], but a closer look into [57] does not confirm this (we elaborate on this claim in Section 2.4). Consequently, the NP-hardness of the smallest grammar problem for fixed alphabets is essentially open (for well over 30 years, taking [58,57] as the first reference, which investigates hardness and complexity questions).

[6] Most of these algorithms were originally designed as compression algorithms (with slightly different purposes than solving the smallest grammar problem), but they can also be regarded as approximation algorithms for the smallest grammar problem and have also been investigated in this regard in [14,39].

1.4 Our Contribution

The main result of this paper is a reduction that proves the smallest grammar problem for fixed alphabets to be NP-complete, at least for alphabet sizes of 17 or larger. As explained above, this closes an important gap in the literature and therefore puts the previous work on grammar-based compression on a more solid theoretical foundation.

Moreover, it also follows that the optimisation version of the smallest grammar problem is APX-hard; thus, the impossibility of a PTAS, previously only known for unbounded alphabets, carries over to the more realistic case of bounded alphabets. By a minor modification of this reduction, we can also show that these two hardness results hold for a slightly different (but frequently used) size measure of grammars, i. e., the *rule-size*, which equals the size of a grammar as defined above plus the number of its rules (both these measures are formally defined Section 2.2).

Given these negative complexity results, we move on to the question of whether smallest grammars can be efficiently computed, if certain parameters (e. g., levels of the derivation tree, number of rules) are bounded. In this regard, we show that smallest grammars can be computed in polynomial time, provided that the size of the nonterminal alphabet (i. e., number of rules) is bounded. This result, which is due to an encoding of the smallest grammar problem as a problem on graphs with interval structure, raises two follow-up questions: (1) is the problem fixed-parameter tractable with respect to the number of rules, (2) is it possible to efficiently compute, how many rules are at least necessary for a smallest grammar? Both of these questions are answered in the negative, by showing W[1]-hardness and NP-hardness, respectively.

Finally, we investigate exact exponential-time algorithms which are not yet considered in the literature. We consider this a relevant topic, since grammars are particularly suitable for solving basic problems directly on the compressed representation without decompression, which motivates scenarios, where an extensive running time is invested only once, in order to obtain an optimal compression, which is then stored and worked with. While brute-force algorithms with running time $\mathcal{O}^*(c^{|w|})$, for a constant c, can be easily found, we present a dynamic programming algorithm with running time $\mathcal{O}^*(3^{|w|})$.

The exploitation of hierarchical structure is one of the main features of grammars (making them suitable tools for structural inference, and also allowing exponential compression rates) and is reflected in the number of levels of the corresponding derivation tree. Hence, from a (parameterised) complexity point of view, it is natural to measure the impact of this “hierarchical depth” of grammars with respect to the complexity of the smallest grammar problem. To this end, we investigate the above mentioned questions also for 1*-level grammars*, i. e., grammars in which only the start rule contains nonterminals and, surprisingly, our results suggest that computing general grammars is, if at all, only insignificantly more difficult than computing 1-level grammars. More precisely, the smallest grammar problem for 1-level grammars is NP-hard for alphabets of size 5 (also with respect to the rule size measure), W[1]-hard if parameterised by the number of rules, it can be solved in polynomial time if the number of rules is bounded by a constant and there is an $\mathcal{O}^*(1.8392^{|w|})$ exact algorithm. Moreover, the exact exponential-time algorithm for the general case works incrementally in the sense that in the process of producing a smallest grammar, it also produces a smallest 1-level grammar, a smallest 2-level grammar and so on.

1.5 Outline of the Paper

In Section 2, we give basic definitions, we define the smallest grammar problem, we illustrate it with several examples and also illustrate in detail the connections between grammar-based compression and the related macro schemes by Storer and Szymanski [58]. The next section contains the hardness results mentioned above, where the 1-level and the multi-level case is treated separately in Sections 3.1 and 3.2, respectively (in Section 3.3, we define and discuss possible extensions of the hardness reductions). The second main part of the paper is Section 4, where we show that the smallest grammar problem can be solved in polynomial time, if the number of nonterminals is bounded (in Section 4.1, we discuss some related questions). In the last part, Section 5, we first present a (simple) exact exponential-time algorithm for the 1-level case and then, in Section 5.2, we define the dynamic programming algorithm for the multi-level case. Finally, in Section 6, we summarise our results, point out open problems and mention further research tasks.

2 Preliminaries

In this section, we first introduce some general mathematical definitions and terminology about strings, and some basic concepts from graph theory and complexity theory. Then we define grammars and the smallest grammar problem and illustrate it by several examples. We conclude this section by a discussion of Storer and Szymanski's external pointer macro scheme already mentioned in Section 1.

Let $\mathbb{N} = \{1, 2, 3, \ldots\}$ denote the natural numbers. By $|A|$, we denote the cardinality of a set A. Let Σ be a finite alphabet of *symbols*. A *word* or *string* (over Σ) is a sequence of symbols from Σ. For any word w over Σ, $|w|$ denotes the length of w and ε denotes the *empty word*, i. e., $|\varepsilon| = 0$. The symbol Σ^+ denotes the set of all non-empty words over Σ and $\Sigma^* = \Sigma^+ \cup \{\varepsilon\}$. For the *concatenation* of two words w_1, w_2 we write $w_1 \cdot w_2$ or simply w_1w_2. For every symbol $a \in \Sigma$, we denote by $|w|_a$ the number of occurrences of symbol a in w. We say that a word $v \in \Sigma^*$ is a *factor* of a word $w \in \Sigma^*$ if there are $u_1, u_2 \in \Sigma^*$ such that $w = u_1vu_2$. If $u_1 = \varepsilon$ (or $u_2 = \varepsilon$), then v is a *prefix* (or a *suffix*, respectively) of w. Furthermore, $\mathsf{F}(w) = \{u \colon u \text{ is a factor of } w\}$ and $\mathsf{F}_{\geq 2}(w) = \{u \colon u \in \mathsf{F}(w), |u| \geq 2\}$. For a position j, $1 \leq j \leq |w|$, we refer to the symbol at position j of w by the expression $w[j]$ and $w[j..j'] =$

$w[j]w[j+1]\dots w[j']$, $j \leq j' \leq |w|$. By w^R, we denote the *reversal* of w, i. e., $w^R = w[n]w[n-1]\dots w[1]$, where $|w| = n$.

A *factorisation* of a word w is a tuple $(u_1, u_2, \dots, u_k)$ with $u_i \neq \varepsilon$, $1 \leq i \leq k$, such that $w = u_1 u_2 \dots u_k$.

2.1 Basic Concepts of Graph Theory and Complexity Theory

We use undirected graphs, which are represented as pairs (V, E), where V is the set of vertices and E is the set of edges. For the sake of convenience, we write edges $\{u, v\} \in E$ also as (u, v) or (v, u). For a vertex $v \in V$, $N(v) = \{u \colon (v, u) \in E\}$ is the (*open*) *neighbourhood* (of v), $N[v] = N(v) \cup \{v\}$ is the *closed neighbourhood* (of v) and, furthermore, we extend the notation of closed neighbourhood to sets $C \subseteq V$ in the obvious way, i. e., $N[C] = \bigcup_{v \in C} N[v]$. A graph is *cubic* (or *subcubic*) if, for every $v \in V$, $|N(v)| = 3$ (or $|N(v)| \leq 3$, respectively).

A set $C \subseteq V$ is

- an *independent set* if, for every $u, v \in C$, $(u, v) \notin E$,
- a *dominating set* if $N[C] = V$,
- an *independent dominating set* if it is both an independent and a dominating set,
- a *vertex cover* if, for every $(u, v) \in E$, $\{u, v\} \cap C \neq \emptyset$.

We are concerned with the corresponding problems of deciding, for a given graph G and a $k \in \mathbb{N}$, whether there is a vertex cover (or an independent dominating set) of cardinality at most k. It is a well-known fact that these decision problems are NP-complete problems (see [25]).

For $k \in \mathbb{N}$, a graph $G = (V, E)$, with $|V| = n$, is a *k-interval graph*, if there are intervals $I_{i,j}$, $1 \leq i \leq |V|$, $1 \leq j \leq k$, on the real line, such that G is isomorphic to $(\{v_i \colon 1 \leq i \leq |V|\}, \{(v_i, v_{i'}) \colon \bigcup_{j=1}^{k} I_{i,j} \cap \bigcup_{j=1}^{k} I_{i',j} \neq \emptyset\})$. For 1-interval graphs (which are also just called interval graphs), it is possible to compute minimal independent dominating sets in linear time (see [19]; note that a perfect elimination ordering (that is part of the input of Farber's algorithm) can be easily computed in our applications, because the intervals are clear).

We assume the reader to be familiar with the basic concepts of complexity theory (for unexplained notions, see Papadimitriou [52]) and the theory of NP-completeness (see [52] and [25]).

As usual, for our running-time estimations, we mainly use the $\mathcal{O}$-notation, but sometimes also the $\mathcal{O}^*$-notation (ignoring polynomial factors). The latter is appropriate, if we are dealing with exponential-time algorithms (see Section 5).

Since we also wish to discuss some of our results from the parameterised complexity point of view, we shall briefly mention the concepts relevant for us (for detailed explanations on parameterised complexity, the reader is referred to the textbooks [17,21,15]). A *parameterised problem* is a decision problem with instances (x, k), where x is the actual input and $k \in \mathbb{N}$ is the *parameter*.

By XP, we denote the class of parameterised problems that are solvable in time $\mathcal{O}(n^{f(k)})$ (where n is the size of the instance) and FPT denotes the class of *fixed-parameter tractable* problems, i. e., problems having an algorithm with running-time $\mathcal{O}(g(k) \cdot f(n))$, for a computable function g and polynomial f.

In order to argue about fixed-parameter intractability, we need the following kind of reductions. A (classical) many-one reduction R from a parameterised problem to another is an *fpt-reduction*, if the parameter of the target problem is bounded in terms of the parameter of the source problem, i. e., there is a recursive function $h\colon \mathbb{N} \to \mathbb{N}$ such that $R(x, k) = (x', k')$ implies $k' \leq h(k)$.

We shall use two different kinds of fixed-parameter intractability. First, if a parameterised problem is NP-hard if the parameter is fixed to a constant, then it is not in FPT, unless $\mathsf{P} = \mathsf{NP}$. As a slightly weaker form of fixed-parameter intractability, the framework of parameterised complexity provides the classes of the so-called W-hierarchy, for which the hard problems (with respect to fpt-reductions) are considered fixed-parameter intractable, i. e., they are not in FPT (under some complexity theoretical assumptions). For a detailed definition of the W-hierarchy, we refer to the textbooks [17,21,15]; in this paper, we only use the first level of this hierarchy, i. e., the class $\mathsf{W}[1]$, and our respective intractability results are $\mathsf{W}[1]$-hardness results.

A minimisation problem[7] P is a triple (I, S, m) with I being the set of instances, S being a function that maps instances $x \in I$ to the set of feasible solutions for x, and m being the objective function that maps pairs (x, y) with $x \in I$ and $y \in S(x)$ to a positive rational number. For every $x \in I$, we denote $m^*(x) := \min\{m(x, y) \colon y \in S(x)\}$. For two minimisation problems P_1, P_2 with P_j given by (I_j, S_j, m_j), $j \in \{1, 2\}$, an *L-reduction* from P_1 to P_2 is a quadruple (f, g, β, γ) such that

- f is a polynomial-time computable function from I_1 to I_2 that satisfies, for every $x \in I_1$ with $S_1(x) \neq \emptyset$, $S_2(f(x)) \neq \emptyset$.
- g is a polynomial-time computable function that, for every $x \in I_1$ and $y \in S_2(f(x))$, maps (x, y) to a solution in $S_1(x)$.
- β is a constant such that $m_2^*(f(x)) \leq \beta \cdot m_1^*(x)$ for each $x \in I_1$.
- γ is a constant such that $m_1(x, g(x, y)) - m_1^*(x) \leq \gamma \cdot (m_2(f(x), y) - m_2^*(f(x)))$ for each $x \in I_1$ and $y \in S_2(f(x))$.

We shall use L-reductions in order to show hardness for APX, the class of optimisation problems for which there exists an approximation algorithm with a constant approximation ratio. Note that, unless $\mathsf{P} = \mathsf{NP}$, an APX-hard problem does not have a polynomial-time approximation scheme (see [6] for detailed information of approximation hardness).

[7] As we are not considering maximisation problems, we define the relevant terminology only for minimisation problems.

2.2 Grammars

A *context-free grammar* is a tuple $G = (N, \Sigma, R, S)$, where N is the set of *nonterminals*, Σ is the *terminal alphabet*, $S \in N$ is the *start symbol* and $R \subseteq N \times (N \cup \Sigma)^+$ is the set of *rules* (as a convention, we write rules $(A, w) \in R$ also in the form $A \to w$). A context-free grammar $G = (N, \Sigma, R, S)$ is a *singleton grammar* if R is a total function $N \to (N \cup \Sigma)^+$ and the relation $\{(A, B) \colon (A, w) \in R, |w|_B \geq 1\}$ is acyclic.

For a singleton grammar $G = (N, \Sigma, R, S)$, let $\mathsf{D}_G \colon (N \cup \Sigma) \to (N \cup \Sigma)^+$ be defined by $\mathsf{D}_G(A) = R(A)$, $A \in N$, and $\mathsf{D}_G(a) = a$, $a \in \Sigma$. We extend D_G to a morphism $(N \cup \Sigma)^+ \to (N \cup \Sigma)^+$ by setting $\mathsf{D}_G(\alpha_1\alpha_2 \dots \alpha_n) = \mathsf{D}_G(\alpha_1)\,\mathsf{D}_G(\alpha_2) \dots \mathsf{D}_G(\alpha_n)$, for $\alpha_i \in (N \cup \Sigma)$, $1 \leq i \leq n$. Furthermore, for every $\alpha \in (N \cup \Sigma)^+$, we set $\mathsf{D}^1_G(\alpha) = \mathsf{D}_G(\alpha)$, $\mathsf{D}^k_G(\alpha) = \mathsf{D}(\mathsf{D}^{k-1}_G(\alpha))$, for every $k \geq 2$, and $\mathfrak{D}_G(\alpha) = \lim_{k\to\infty} \mathsf{D}^k_G(\alpha)$ is the *derivative* of α. By definition, $\mathfrak{D}_G(\alpha)$ exists for every $\alpha \in (N \cup \Sigma)^+$ and is an element from Σ^+. The *size* of the singleton grammar G is defined by $|G| = \sum_{A\in N} |\mathsf{D}_G(A)|$ and the *rule-size* of G is defined by $|G|_{\mathsf{r}} = |G| + |N|$ or, equivalently, $|G|_{\mathsf{r}} = \sum_{A\in N}(|\mathsf{D}_G(A)| + 1)$. Our main size measure will be $|\cdot|$. The rule-size $|\cdot|_{\mathsf{r}}$ will play a role in Section 3.3 and will be discussed in more detail there.

Remark 1 The class of singleton grammars exactly coincides with the class of context-free grammars that do not have unreachable rules (i. e., rules that cannot occur in any derivation) and that can derive exactly one word. As mentioned before, such grammars are also called *straight-line programs* in the literature. A context-free grammar that can derive only a single word and is *not* a singleton grammar must contain some rules that are not reachable. Since unreachable rules can easily be discovered and removed, we directly add this restriction to the concept of singleton grammars.

The *derivation tree* of G is a ranked ordered tree with node-labels from $\Sigma \cup N$, inductively defined as follows. The root is labelled by S and every node labelled by $A \in N$ with $\mathsf{D}(A) = \alpha_1\alpha_2 \dots \alpha_n$ has n children labelled by $\alpha_1, \alpha_2, \dots, \alpha_n$ in exactly this order; note that this means that all leaves are from Σ.

From now on, we simply use the term *grammar* instead of singleton grammar and if the grammar under consideration is clear from the context, we also drop the subscript G. We set $\mathfrak{D}(G) = \mathfrak{D}(S)$ and say that G *is a grammar for* $\mathfrak{D}(G)$. Since for singleton grammars, the start symbol is somewhat superfluous, we will ignore it and denote grammars $G = (N, \Sigma, R, S)$ in the form $G = (N, \Sigma, R, \mathsf{ax})$ instead, where $\mathsf{ax} = R(S)$ is called the *axiom* (*of* G). In particular, we interpret derivations to start directly with the axiom and, correspondingly, we also sometimes ignore the root of derivation trees. However, this does not change the size measures $|\cdot|$ and $|\cdot|_{\mathsf{r}}$, which, when ignoring the start symbol, can also be defined as $|G| = (\sum_{A\in N} |\mathsf{D}_G(A)|) + |\mathsf{ax}|$ and $|G|_{\mathsf{r}} = (\sum_{A\in N}(|\mathsf{D}_G(A)| + 1)) + |\mathsf{ax}| + 1$.

The number of *levels* of a grammar $G = (N, \Sigma, R, \mathsf{ax})$ is $\min\{k \colon \mathsf{D}^k_G(\mathsf{ax}) = \mathfrak{D}_G(\mathsf{ax})\}$, and a grammar with d levels is a *d-level grammar*. Intuitively speak-

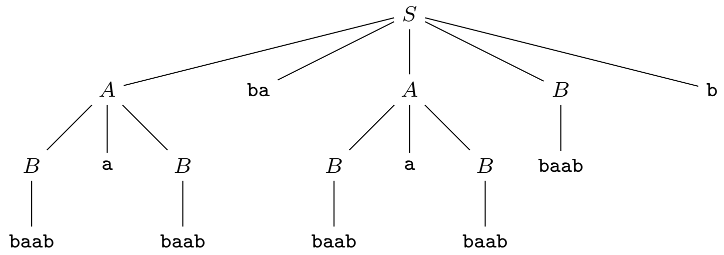

Fig. 1 Derivation tree for the grammar G from Example 1 (for the sake of convenience, neighbouring leaves are merged).

ing, a grammar G is a d-level grammar if we need exactly d derivation steps in order to derive $\mathfrak{D}(G)$ from the axiom; thus, the number of levels measures what we called in the introduction the "hierarchical depth" of a grammar. Note that for a d-level grammar, the derivation tree has a maximum depth of $d+1$ and $d+2$ levels (when counting the root as well). With this definition, the grammars that are the most restricted with respect to their hierarchical depth and that are still reasonable, are 1-level grammars (i. e., an axiom that derives a word in one step).

Let $G = (N, \Sigma, R, \mathsf{ax})$ be a 1-level grammar. The *profit* of a rule $(A, \alpha) \in R$ is defined by $\mathsf{p}(A) = |\mathsf{ax}|_A(|\alpha| - 1) - |\alpha|$. Intuitively speaking, if all occurrences of A in ax are replaced by α and the rule $A \to \alpha$ is deleted, then the size of the grammar increases by exactly $\mathsf{p}(A)$. Consequently, $|G| = |\mathfrak{D}(G)| - \sum_{A \in N} \mathsf{p}(A)$.

Example 1 The grammar $G = (N, \Sigma, R, \mathsf{ax})$ with $N = \{A, B\}$, $\Sigma = \{\mathtt{a}, \mathtt{b}\}$, $\mathsf{ax} = A\mathtt{ba}AB\mathtt{b}$ and

$$R = \{A \to B\mathtt{a}B, B \to \mathtt{baab}\}$$

is a 2-level grammar of size 13 (and rule-size 16) with axiom $A\mathtt{ba}AB\mathtt{b}$. Furthermore, $\mathfrak{D}(B) = \mathtt{baab}$, $\mathfrak{D}(A) = \mathfrak{D}(B)\mathtt{a}\,\mathfrak{D}(B) = \mathtt{baababaab}$ and

$$\mathfrak{D}(G) = \mathfrak{D}(S) = \underbrace{\mathtt{baababaab}}_{\mathfrak{D}(A)}\mathtt{ba}\underbrace{\mathtt{baababaab}}_{\mathfrak{D}(A)}\underbrace{\mathtt{baab}}_{\mathfrak{D}(B)}\mathtt{b}\,.$$

Consequently, G is a size 13 representation of a word of length 25. A derivation tree of G can be seen in Figure 1.

Replacing the axiom by $R(A)\mathtt{ba}R(A)B\mathtt{b} = B\mathtt{a}B\mathtt{ba}B\mathtt{a}BB\mathtt{b}$ and deleting rule $A \to B\mathtt{a}B$ turns G into a 1-level grammar G' with $\mathfrak{D}(G') = \mathfrak{D}(G)$. Moreover, $\mathsf{p}(B) = |\mathsf{ax}|_B(|R(B)| - 1) - |R(B)| = 5(4-1) - 4 = 11$ and $|G'| = |\mathfrak{D}(G')| - \mathsf{p}(B) = 25 - 11 = 14$.

A *smallest* grammar for a word w is any grammar G with $\mathfrak{D}(G) = w$ and $|G| \leq |G'|$ for every grammar G' with $\mathfrak{D}(G') = w$; generally, a grammar G is smallest if it is a smallest grammar for $\mathfrak{D}(G)$ (grammars that are smallest with respect to the rule-size measure will be called r-*smallest* grammars). The decision problem variant of computing smallest grammars is defined as follows:

Smallest Grammar Problem (SGP)
Instance: A word w and a $k \in \mathbb{N}$.
Question: Does there exist a grammar G with $\mathfrak{D}(G) = w$ and $|G| \leq k$?

The Smallest 1-Level Grammar Problem (1-SGP) is defined analogously, with the only difference that we ask for a 1-level grammar of size at most k. By SGP_{r} and $1\text{-SGP}_{\mathsf{r}}$, we denote the problem variants, where we consider the rule-size instead of the size, i. e., we require $|G|_{\mathsf{r}} \leq k$.

The optimisation variant of SGP, i. e., the task of actually producing a smallest grammar for a given word w, shall be denoted by $\text{SGP}_{\mathsf{opt}}$ (and $\text{SGP}_{\mathsf{r,opt}}$ if we are concerned with the rule-size). More precisely, according to the definitions given in Section 2.1, $\text{SGP}_{\mathsf{opt}} = (I, S, m)$, where $I = \Sigma^*$, $S(w) = \{G \colon \mathfrak{D}(G) = w\}$ and $m(w, G) = |G|$ (or $m(w, G) = |G|_{\mathsf{r}}$ for $\text{SGP}_{\mathsf{r,opt}}$).

2.3 Examples

While the following examples illustrate the smallest grammar problem in general, they are particularly tailored to the technicalities to be encountered in Section 3, i. e., they shall point out the difficulties arising in predicting how factors in a larger word are compressed by a smallest grammar, which is crucial in the design of gadgets for a hardness reduction.

Let $w = \prod_{i=1}^{n} 10^i$ be a word over the binary alphabet $\Sigma = \{0, 1\}$, where $n = 2^k$, $k \in \mathbb{N}$. This word has a very simple structure and can be interpreted as a list of a (potentially unbounded) number of integers. This is crucial, since if we want to encode objects (e. g., graphs), the size of which is not bounded in terms of the alphabet size, then structures of this form will inevitably appear.

One way of compressing w that comes to mind is by the use of rules $A_1 \to 10$, $A_i \to A_{i-1}0$, $2 \leq i \leq n-1$, and an axiom $A_1 A_2 \dots A_{n-1} A_{n-1} 0$, which leads to the grammar $G_1 = (N, \Sigma, R, \mathsf{ax})$, with:

$$\begin{aligned} N &= \{A_i \colon 1 \leq i \leq n-1\}\,, \\ R &= \{A_1 \to 10\} \cup \{A_i \to A_{i-1}0 \colon 2 \leq i \leq n-1\}\,, \\ \mathsf{ax} &= A_1 A_2 \dots A_{n-1} A_{n-1} 0\,. \end{aligned}$$

This grammar has an overall size given by $|G_1| = \underbrace{n+1}_{\mathsf{ax}} + \underbrace{2(n-1)}_{\text{rules}} = 3n - 1$.

However, it is also possible to construct the factors 0^i, $1 \leq i \leq n$, "from the middle" by rules $A_1 \to 010$, $A_i \to 0A_{i-1}0$, $2 \leq i \leq \frac{n}{2} - 1$, and an axiom $1(A_1)^2(A_2)^2 \dots$ By using these ideas, we can construct the smaller grammar $G_2 = (N, \Sigma, R, \mathsf{ax})$, where

$$\begin{aligned} N &= \{A_i \colon 1 \leq i \leq \tfrac{n}{2} - 1\} \cup \{B_i \colon 1 \leq i \leq k-2\}\,, \\ R &= \{A_1 \to 010, B_1 \to 00\} \cup \{A_i \to 0A_{i-1}0 \colon 2 \leq i \leq \tfrac{n}{2} - 1\} \cup \\ &\quad \{B_i \to B_{i-1}B_{i-1} \colon 2 \leq i \leq k-2\}\,, \\ \mathsf{ax} &= 1(A_1)^2(A_2)^2 \dots (A_{\frac{n}{2}-1})^2 0 A_{\frac{n}{2}-1} 0 B_{k-2} B_{k-2}\,. \end{aligned}$$

We have $|G_2| = \underbrace{n+4}_{\mathsf{ax}} + \underbrace{3(\frac{n}{2}-1) + 2(k-2)}_{\text{rules}} = \frac{5n}{2} + 2k - 3$.

Both of these grammars achieve an asymptotic compression rate of order $\mathcal{O}(\sqrt{|w|})$, but, generally, grammars are capable of exponential compression rates (see [14,39]). Aiming for such exponential compression, it seems worthwhile to represent every unary factor $0^{2^{\ell}}$, $1 \leq \ell \leq k$, by a nonterminal B_{ℓ} (obviously, this requires only k rules of size 2) and then represent all unary factors by sums of these powers (e. g., 0^{74} is compressed by $B_1B_3B_6$). Formally, consider $G_3 = (N, \Sigma, R, \mathsf{ax})$, where

$$\begin{aligned} N &= \{B_i \colon 1 \leq i \leq k-1\}\,, \\ R &= \{B_1 \to 00\} \cup \{B_i \to B_{i-1}B_{i-1} \colon 2 \leq i \leq k-1\}\,, \\ \mathsf{ax} &= \left(\prod_{i=1}^{n-1} 1\alpha_i\right)(B_{k-1})^2\,, \end{aligned}$$

where $\alpha_i = x_0x_1 \ldots x_{k-1}$ and, for every j, $1 \leq j \leq k-1$, $x_j = B_j$ if the j^{th} bit (i. e., the one representing 2^j) of the binary representation of i is 1 and $x_j = \varepsilon$ otherwise. However, this yields a grammar of size

$$|G_3| = \underbrace{\tfrac{1}{2}(n-1)k}_{\mathsf{ax}} + \underbrace{2(k-1)}_{\text{rules}} = \frac{k(n+3)}{2} - 2\,,$$

which, if k is sufficiently large, is worse than the previous grammars.

A grammar that is even smaller than G_2 can be obtained by combining the idea of G_2 with that of representing factors $0^{2^{\ell}}$ by nonterminals B_{ℓ}. More precisely, for every ℓ, $1 \leq i \leq k-2$, we represent $0^{2^{\ell}}$ by an individual nonterminal B_{ℓ} and, in addition, we use rules $A_1 \to 010$, $A_i \to 0A_{i-1}0$, $2 \leq i \leq \frac{n}{4}$. Then the left and right half of w can be compressed in the way of G_2, with the only difference that in the right part, for every unary factor, we also need an occurrence of B_{k-1}, i. e., consider $G_4 = (N, \Sigma, R, \mathsf{ax})$ with:

$$\begin{aligned} N &= \{A_i \colon 1 \leq i \leq \tfrac{n}{4}\} \cup \{B_i \colon 1 \leq i \leq k-1\}\,, \\ R &= \{A_1 \to 010, B_1 \to 00\} \cup \{A_i \to 0A_{i-1}0 \colon 2 \leq i \leq \tfrac{n}{4}\} \cup \\ &\quad \{B_i \to B_{i-1}B_{i-1} \colon 2 \leq i \leq k-1\}\,, \\ \mathsf{ax} &= 1(A_1)^2(A_2)^2 \ldots (A_{\frac{n}{4}})^2 B_{k-2} \\ &\quad (A_1B_{k-1})^2(A_2B_{k-1})^2 \ldots (A_{\frac{n}{4}-1}B_{k-1})^2 A_{\frac{n}{4}} B_{k-1} B_{k-2}\,. \end{aligned}$$

This grammar yields a size of $|G_4| = \underbrace{\frac{3n}{2} + 1}_{\mathsf{ax}} + \underbrace{\frac{3n}{4} + 2(k-1)}_{\text{rules}} = \frac{9n}{4} + 2k - 1$. Note that again the asymptotic compression rate is of order $\mathcal{O}(\sqrt{|w|})$.

These considerations point out that even for simply structured words like w, it is very difficult to determine the structure of a smallest grammar or its size. However, for reducing an NP-hard problem, we need to know, to at least some extent, how smallest grammars compress the constructed strings in order to relate the reduced instances to the original instances. Consequently, the above examples point out the challenges that arise in this regard.

We conclude this list of examples, by pointing out that giving a smallest grammar for our toy-example $w = \prod_{i=1}^{n} 10^i$ in dependency of n, is essentially an open problem. A respective asymptotic bound of $\Omega(\sqrt{|w|})$ is a reasonable assumption, but we have no proof for this claim.

2.4 Storer and Szymanski's External Pointer Macro Scheme and Grammar-Based Compression

Storer and Szymanski [58] introduce a very general form of a compression scheme that covers a large variety of different compression strategies, in particular also grammar-based compression. On the one hand, we cite their work as the first that, in a sense, considered grammar-based compression, but in the context of our paper, it is also of greater importance for the following reasons. The technical report [57][8] provides a comprehensive complexity analysis of many different variants of Storer and Szymanski's compression scheme with many NP-hardness reductions. Some of the considered variants also concern the case of fixed alphabets, which has led to the misunderstanding that the hardness of the smallest grammar problem for fixed alphabets is provided by [57], leading to the misconception that also in practical scenarios – i. e., for fixed alphabets – grammar-based compression is known to be intractable. Since closing this gap by providing the assumed hardness result is one of the main objectives of this paper, we shall discuss in some more detail why it *cannot* already be found among the many hardness results of [57].

First, we recall the definitions of Storer and Szymanski [58] that are relevant here. For a word $w \in \Sigma^+$ and a pointer size $p \in \mathbb{N}$, a *compressed form of w for pointer size p using the external pointer macro*, EPM for short, is any word $s_0\#s_1$ with $s_0, s_1 \in (\Sigma \cup \{1, 2, \dots, |s_0|\}^2)^+$, $\# \notin \Sigma$, and w can be obtained from $s_0\#s_1$ by repeating the following two steps:

- Replace every symbol (i, j) in s_1 by $s_0[i..j]$,
- repeat the first step until s_1 equals w.

The size of an EPM $s_0\#s_1$ is defined by $\sum_{i=1}^{|s_0 s_1|} \ell_i$, where $\ell_i = 1$, if $s_0 s_1[i] \in \Sigma$ and $\ell_i = p$, otherwise (i. e., each occurrence of a symbol from $\{1, 2, \dots, |s_0|\}^2$ (the actual *pointers*) contribute the pointer size p to the overall size of the EPM).

A grammar for a word w easily translates into an EPM for w. For example, the grammar $G = (N, \Sigma, R, \mathsf{ax})$ with $N = \{A, B\}$, $\Sigma = \{a, b, c\}$, $R = \{A \to BcB, B \to ba\}$ and $\mathsf{ax} = AabBBAc$ translates into the external pointer macro $ba(1, 2)c(1, 2)\#(3, 5)ab(1, 2)(1, 2)(3, 5)c$. More precisely, the prefix ab is the right side of the rule for B, $(1, 2)c(1, 2)$ corresponds to the right side of the rule for A, where the occurrences of B are represented by pointers $(1, 2)$ to

[8] The report can be downloaded at `http://www.informatik.uni-trier.de/~fernau/Sto77.pdf`.

the prefix $s_0[1..2] = ab$, $(3,5)ab(1,2)(1,2)(3,5)c$ corresponds to the axiom, where occurrences of A and B are represented by pointers $(3,5)$ and $(1,2)$, respectively. If the pointer size is 1, then the EPM has the same size as the grammar.

If an EPM $s_0\#s_1$ is *non-overlapping*, i. e., it is never the case that for two pointers (i,j) and (k,ℓ) we have $i \leq k \leq j$ or $k \leq i \leq \ell$, then it also translates into a grammar by transforming each pointer (i,j) into a nonterminal $A_{(i,j)}$ with a rule $A_{(i,j)} \to s_0[i..j]$. In this regard, it is important to note that the property of an EPM that s_1 can be turned into w by repeated replacement of the pointers ensures that the derivation function of the grammar constructed in this way is acyclic.

We conclude that the concept of singleton grammars and the concept of EPMs with pointer size 1 and without overlapping are more or less identical, i. e., they just differ syntactically. Consequently, the problem of grammar-based compression and the problem of computing smallest EPMs with pointer size 1 and without overlapping are identical problems.

However, a closer look at Storer [57] shows that in this paper the variant of computing EPMs with pointer size 1 is not considered. Instead, the focus is on EPMs (and other kind of compression schemes), for which the pointer size is not even constant, but a function of the length of the word that is compressed, typically logarithmic in the size $|w|$. Note that this avoids the main difficulties encountered when designing a reduction for grammar-based compression with fixed alphabets (see Section 3): the factors that encode vertices of a graph must have unbounded length, which makes it rather difficult to control how the grammar compresses these codewords. On the other hand, if the pointers (which correspond to nonterminals in the grammar) have size $\log(|w|)$, then it does not make sense to compress factors that are smaller than this size (since we gain nothing by replacing them by pointers). It is straightforward to represent a graph as a word of length linear in the size of the graph, where the length of the factors (i. e., the codewords) that represent single vertices are logarithmic in the size of the graph (this is the case in all reductions of [14, 39, 58]). The property mentioned above, i. e., that factors of logarithmic size are not compressed, then simply means that we can assume that the codewords for vertices are not compressed in the string that describes the graph, which makes is rather simple to devise a hardness reduction (in fact, controlling the possible compression of codewords is the main technical challenge in our reductions).

3 NP-Hardness of Computing Smallest Grammars for Fixed Alphabets

In their basic structure, the hardness reductions to be presented next are similar to the one from [14, 39], which shows NP-hardness of SGP for unbounded alphabets by a reduction from the vertex cover problem. All the effort of this section will consist in the extension of the general idea to the case of a fixed alphabet. In order to facilitate the accessibility of our technical proofs, we shall sketch this reduction from [14,39].

Let $\mathcal{G} = (V, E)$ be a graph with

$$V = \{v_1, \ldots, v_n\} \text{ and } E = \{(v_{j_{2i-1}}, v_{j_{2i}}) \colon 1 \leq i \leq m\}\,.$$

We define the following word over the alphabet $V \cup \{\diamond_i \colon 1 \leq i \leq 5n+m\} \cup \{\#\}$ (for the sake of simplicity, every individual occurrence of $\diamond$ in the word stands for a distinct symbol of $\{\diamond_i \colon 1 \leq i \leq 5n+m\}$):

$$w_{\mathcal{G}} = \prod_{i=1}^{n} (\# v_i \diamond v_i \# \diamond)^2 \prod_{i=1}^{n} (\# v_i \# \diamond) \prod_{i=1}^{m} (\# v_{j_{2i-1}} \# v_{j_{2i}} \# \diamond)\,.$$

Let $G = (N, \Sigma, R, S)$ be a smallest grammar for $w_{\mathcal{G}}$, then we can observe the following:

- For every $A \in N$, $\mathfrak{D}(A) \in \{\# v_i, v_i \#, \# v_i \# \colon 1 \leq i \leq n\}$. This is due to the fact that the only factors of $w_{\mathcal{G}}$ with repetitions are of the form $\# v_i$, $v_i \#$ or $\# v_i \#$.
- We can assume that, for every i, $1 \leq i \leq n$, there are rules $A_i \to \# v_i$ and $B_i \to v_i \#$, since if some of these rules are missing, then adding them and compressing the respective factors does not increase the size of the grammar.
- Let $\mathfrak{I} \subseteq \{1, 2, \ldots, n\}$ contain exactly the indices i such that a rule with derivative $\# v_i \#$ exists; moreover, we can assume that all these rules have the form $C_i \to A_i \#$.
- Let $\Gamma = \{v_i \colon i \in \mathfrak{I}\}$. If an edge $(v_{j_{2i-1}}, v_{j_{2i}})$ is not covered by Γ, then adding a rule $C_{j_{2i-1}} \to A_{j_{2i-1}} \#$ or $C_{j_{2i}} \to A_{j_{2i}} \#$ does not increase the size of the grammar. So we can assume that Γ is a vertex cover.

These observations show that there exists a grammar G for $w_{\mathcal{G}}$ with $|G| \leq 15n + 3m + k$ if and only if there is a vertex cover for $\mathcal{G}$ of size at most k (for a formal proof, we refer to [14,39]).

A simple modification of this reduction yields the following.

Theorem 1 1-SGP *is* NP*-complete.*

Proof We slightly change the reduction from [14,39] as follows:

$$w_{\mathcal{G}} = \prod_{i=1}^{n} (\# v_i \diamond v_i \# \diamond)^2 \prod_{i=1}^{n} (\# v_i \# \diamond)^2 \prod_{i=1}^{m} (\# v_{j_{2i-1}} \# v_{j_{2i}} \# \diamond)\,.$$

The only difference from the original reduction is that the size of the rules with derivative $\# v_i \#$ has increased by 1, i. e., they now have the form $C_i \to \# v_i \#$, so by repeating the factors $\# v_i \# \diamond$, we make sure that adding such a rule whenever an edge is not covered does not increase the size of the grammar. □

In these reductions, we encode the different vertices of a graph by single symbols and also use individual separator symbols (i. e., symbols with only one occurrence in the word to be compressed). This makes it particularly easy to devise suitable gadgets, but, on the other hand, it assumes that we have an arbitrarily large alphabet at our disposal. In the remainder of this section, we shall extend these hardness results to the more realistic case of fixed alphabets. The general structure of our reductions is similar to the ones of [14,39,57] sketched above, but, due to the constraint of having a fixed alphabet, they substantially differ on a more detailed level. More precisely, since fixed alphabets make it impossible to use single symbols (or even words of constant size) as separators or as representatives for vertices, we need to use special encodings for which we are able to determine how a smallest grammar will compress them (in this regard, recall our examples from section 2.3 demonstrating how difficult it can be to determine a smallest grammar even for a single simply structured word). This constitutes a substantial technical challenge, which complicates our reductions considerably.

In the following, we prove that 1-SGP and SGP are NP-hard, even for constant alphabet of size 5 and 24, respectively. The stronger result claimed in the abstract and introduction, i. e., the hardness of SGP for alphabets of size 17, is presented later as an improvement (see Section 3.4, Corollary 1).

3.1 The 1-Level Case

As a tool for proving the hardness of 1-SGP, but also as a result in its own right, we first show that the compression of any 1-level grammar is at best quadratic (in contrast to general grammars, which can achieve exponential compression). Note that the bound of Lemma 1 is tight, e. g., consider a^{n^2} and a grammar with rules $S \to A^n$ and $A \to \mathsf{a}^n$.

Lemma 1 *Let G be a 1-level grammar. Then $|G| \geq 2\sqrt{|\mathfrak{D}(G)|}$.*

Proof Let $n = |\mathfrak{D}(G)|$, let ax be the axiom and let $A \to u$ be a rule with a right side of maximum length. Obviously, $|\mathsf{ax}||u| \geq n$, and, since $x+y \geq 2\sqrt{xy}$ holds for all $x, y \geq 0$, also $|\mathsf{ax}| + |u| \geq 2\sqrt{|\mathsf{ax}||u|}$. Consequently,

$$|G| \geq |\mathsf{ax}| + |u| \geq 2\sqrt{|\mathsf{ax}||u|} \geq 2\sqrt{n}\,.$$

□

In order to prove the NP-hardness of 1-SGP for constant alphabets, we also devise a reduction from the vertex cover problem. To this end, let $\mathcal{G} = (V, E)$ be the graph defined above and, without loss of generality, we assume $n \geq 40$. We define $\Sigma = \{\mathsf{a}, \mathsf{b}, \diamond, \star, \#\}$ and $[\diamond] = \diamond^{n^3}$. For each i, $1 \leq i \leq n$, we encode v_i by a word $\overline{v_i} \in \{\mathsf{a}, \mathsf{b}\}^{\lceil \log(n) \rceil}$ such that $\overline{v_i} \neq \overline{v_j}$ if and only if $i \neq j$ (e. g., by taking $\overline{v_i}$ to be the binary representation of i over symbols a and b with $\lceil \log(n) \rceil$ many digits). We now define the following word over Σ:

$$\begin{aligned} w = &\prod_{i=1}^{n}(\#\overline{v_i}[\diamond]\overline{v_i}\#[\diamond])^{2\lceil\log(n)\rceil+3}\prod_{i=1}^{n}(\#\overline{v_i}\#[\diamond])^{\lceil\log(n)\rceil+1}\\ &\prod_{i=1}^{m}(\#\overline{v_{j_{2i-1}}}\#\overline{v_{j_{2i}}}\#[\diamond])^2\star[\diamond]^{n^3}\,. \end{aligned}$$

First, we show how a vertex cover for $\mathcal{G}$ translates into a grammar for w:

Lemma 2 *If there exists a size k vertex cover of $\mathcal{G}$, then there exists a* 1*-level grammar G with $\mathfrak{D}(G) = w$ and $|G| = 13n\lceil\log(n)\rceil + 17n + k + 6m + 1 + 2n^3$.*

Proof Let $\Gamma \subseteq V$ be a size-k vertex cover of $\mathcal{G}$. We define a grammar $G = (N, \Sigma, R, \mathsf{ax})$ with

$$\begin{aligned} N &= \{D, \overleftarrow{V_i}, \overrightarrow{V_i}, \overleftrightarrow{V_j} : 1 \le i \le n, v_j \in \Gamma\}\,,\\ R &= \{S \to u, D \to [\diamond]\} \cup \{\overleftarrow{V_i} \to \#\overline{v_i}, \overrightarrow{V_i} \to \overline{v_i}\# : 1 \le i \le n\} \cup\\ &\quad \{\overleftrightarrow{V_j} \to \#\overline{v_j}\# : v_j \in \Gamma\}\,,\\ \mathsf{ax} &= \prod_{i=1}^{n}(\overleftarrow{V_i}\,D\,\overrightarrow{V_i}\,D)^{2\lceil\log(n)\rceil+3}\prod_{i=1}^{n}(y_i\,D)^{\lceil\log(n)\rceil+1}\prod_{i=1}^{m}(z_i\,D)^2\ \star\ D^{n^3}\,, \end{aligned}$$

where, for every i, $1 \le i \le n$, $y_i = \overleftrightarrow{V_i}$ if $v_i \in \Gamma$ and $y_i = \overleftarrow{V_i}\#$ otherwise, and, for every i, $1 \le i \le m$, $z_i = \overleftrightarrow{V}_{j_{2i-1}}\overrightarrow{V}_{j_{2i}}$ if $v_{j_{2i-1}} \in \Gamma$ and $z_i = \overleftarrow{V}_{j_{2i-1}}\overleftrightarrow{V}_{j_{2i}}$ if $v_{j_{2i-1}} \notin \Gamma$ (note that in this case $v_{j_{2i}} \in \Gamma$).

Obviously, G is a 1-level grammar and it can be easily verified that $\mathfrak{D}(G) = w$. It remains to determine the size of G. To this end, we first observe that each rule $\overleftarrow{V_i} \to \#\overline{v_i}$ and $\overrightarrow{V_i} \to \overline{v_i}\#$, $1 \le i \le n$, has size of $\lceil\log(n)\rceil + 1$, each rule $\overleftrightarrow{V_j} \to \#\overline{v_j}\#$, $v_j \in \Gamma$, has size of $\lceil\log(n)\rceil + 2$, and the rule $D \to [\diamond]$ has size of n^3. Hence, the size contributed by these rules is

$$2n\lceil\log(n)\rceil + 2n + k\lceil\log(n)\rceil + 2k + n^3\,.$$

The axiom has size of

$$\begin{aligned} &4n(2\lceil\log(n)\rceil + 3) + (3n - k)(\lceil\log(n)\rceil + 1) + 6m + 1 + n^3\\ = {}&11n\lceil\log(n)\rceil - k\lceil\log(n)\rceil + 15n - k + 6m + 1 + n^3\,. \end{aligned}$$

So the total size is

$$13n\lceil\log(n)\rceil + 17n + k + 6m + 1 + 2n^3\,.$$

□

Next, we take care of the opposite direction, i. e., we show how a vertex cover can be extracted from a grammar for w:

Lemma 3 *If there exists a* 1*-level grammar* G *with* $\mathfrak{D}(G) = w$ *and* $|G| \leq 13n\lceil\log(n)\rceil + 17n + k + 6m + 1 + 2n^3$*, then there exists a size* k *vertex cover of* $\mathcal{G}$*.*

Proof Let $G = (N, \Sigma, R, \mathsf{ax})$ be a smallest 1-level grammar with

$$|G| \leq 13n\lceil\log(n)\rceil + 17n + k + 6m + 1 + 2n^3$$

and $\mathfrak{D}(G) = w$. We first observe that, since $n \geq 40$,

$$13n\lceil\log(n)\rceil + 17n + k + 6m + 1 < 19n^2 + 18n < 20n^2 = \frac{40}{2}n^2 \leq \frac{n}{2}n^2 = \frac{n^3}{2}\,.$$

Thus, $|G| < \frac{n^3}{2} + 2n^3 = \frac{5n^3}{2}$. Due to the separator symbol $\star$ with only one occurrence in w, we know that the axiom of G has the form $u \star u'$. Hence, we can consider all the nonterminals (and their rules) that occur in u' as an individual 1-level grammar G' for the word $\mathfrak{D}(u') = [\diamond]^{n^3}$ of size n^6. By Lemma 1, we can conclude that $|G'| \geq 2n^3$; thus, $2n^3 \leq |G| < \frac{5n^3}{2}$.

Claim 1: There is a $D \in N$ with $D \to [\diamond]$ and, for every other rule $A \to x$ in R, $|x|_\diamond = 0$.

Proof of Claim 1: First, we assume that there is a rule $A \to \diamond^\ell$ with $\ell > n^3$. This rule can only be used in order to compress the suffix $[\diamond]^{n^3}$ of w, since the other part of w has no occurrence of a factor $\diamond^\ell$. Hence, we can replace $A \to \diamond^\ell$ by the rule $A \to \diamond^{n^3}$ and change the axiom to $u \star A^{n^3}$. By Lemma 1, the rule $A \to \diamond^{n^3}$ with axiom A^{n^3} compresses the subword $[\diamond]^{n^3}$ optimally which means that this operation does not increase the size of G. Therefore, we conclude that G does not contain a rule $A \to \diamond^\ell$ with $\ell > n^3$.

Since w contains at least n^3 non-overlapping occurrences of the factor $[\diamond]$ and since $|G| < 3n^3$, at least one of these factors must be produced by at most 2 nonterminals. This implies that there is a rule $B \to v$ with $|v| \geq \frac{|[\diamond]|}{2} = \frac{n^3}{2}$. If v contains a symbol from $\Sigma \setminus \{\diamond\}$, then $B \to v$ is not a rule of G'; thus, by Lemma 1, it follows that $|G| \geq |G'| + \frac{n^3}{2} \geq 2n^3 + \frac{n^3}{2} = \frac{5n^3}{2}$, which is a contradiction. Hence, we can conclude that $v \in \{\diamond\}^*$ and we further assume that, among all rules with a right side in $\{\diamond\}^*$ of size at least $\frac{n^3}{2}$, $B \to v$ is such that $|v|$ is maximal. Moreover, let $|v| = n^3 - t$, for a $t \in \mathbb{N}$.

We note that, due to the maximality of $B \to v$ and the fact that all rules in G' have a right side in $\{\diamond\}^*$, a rule of maximum size in G' has size at most $n^3 - t$. In particular, this implies

$$|u'| \geq \frac{n^6}{n^3 - t} > \frac{n^6 - t^2}{n^3 - t} = \frac{(n^3 + t)(n^3 - t)}{n^3 - t} = n^3 + t\,,$$

where u' is the right side of the axiom as defined above.

We now remove rule $B \to v$, add the rule $D \to [\diamond]$ and replace part u' of the axiom by D^{n^3}. Since $|[\diamond]| = |v| + t$ and $|u'| \geq n^3 + t = |D^{n^3}| + t$, this does not increase the size of the grammar. However, the rule $B \to v$ might have been used in order to produce some of the factors $[\diamond]$ in the left part u

of the axiom of G; thus, since we removed the rule $B \to v$, we have to repair G accordingly.

To this end, we first note that every occurrence of $[\diamond]$ to the left of $\star$ in w is compressed by a sequence $E_1 C_1 C_2 \ldots C_p E_2$ of terminals or nonterminals, such that $\mathfrak{D}(E_1 C_1 C_2 \ldots C_p E_2) = x[\diamond]y$, where $E_1 \to x\diamond^q$, $q \geq 1$, or $E_1 = \varepsilon$, and $E_2 \to \diamond^r y$, $r \geq 1$, or $E_2 = \varepsilon$. For every such occurrence of $[\diamond]$ to the left of $\star$ in w, we exchange $E_1 C_1 C_2 \ldots C_p E_2$ by $E'_1 D E'_2$, where $E'_1 = \varepsilon$, if $E_1 = \varepsilon$ and $E'_1 = x$ if $E_1 \to x\diamond^q$, $q \geq 1$, and $E'_2 = \varepsilon$, if $E_2 = \varepsilon$ and $E'_2 = y$ if $E_2 \to \diamond^r y$, $r \geq 1$. This construction removes rules or shortens them; thus, in order to conclude that the overall size of the grammar does not increase, we only have to observe that the size of the axiom is not increased. To this end, we first observe that if $p = 0$, then E_1 or E_2 must have a right side of length at least $\frac{n^3}{2}$ that contains a symbol from $\Sigma \setminus \{\diamond\}$, but, as shown above, such rules do not exist. Hence, we can assume that $p \geq 1$. Furthermore, since $E_1 = \varepsilon$ implies $E'_1 = \varepsilon$ and $E_2 = \varepsilon$ implies $E'_2 = \varepsilon$, $|E_1 C_1 C_2 \ldots C_p E_2| \geq |E'_1 D E'_2|$ follows.

We conclude that the overall size of the grammar did not increase due to these modifications. Moreover, G now contains a rule $D \to [\diamond]$ and, since all occurrences of $\diamond$ in w are produced by this rule, we can safely remove all other rules that produce an occurrence of $\diamond$ from the grammar. (*Claim* 1) □

The statement of the previous claim particularly implies that the axiom of G has the form

$$\mathsf{ax} = \prod_{i=1}^{n} (\alpha_i \, D \, \alpha'_i \, D)^{2\lceil \log(n) \rceil + 3} \prod_{i=1}^{n} (\beta_i \, D)^{\lceil \log(n) \rceil + 1} \prod_{i=1}^{m} (\gamma_i \, D)^2 \, \star \, D^{n^3} \,,$$

where $\alpha_i, \alpha'_i, \beta_i, \gamma_j \in (N \cup \Sigma)^*$, $1 \leq i \leq n$, $1 \leq j \leq m$.

Claim 2: For every i, $1 \leq i \leq n$, $\alpha_i = \overleftarrow{V_i}$, $\alpha'_i = \overrightarrow{V_i}$, where $\overleftarrow{V_i}, \overrightarrow{V_i}$ are nonterminals with rules $\overleftarrow{V_i} \to \#\overline{v_i}$ and $\overrightarrow{V_i} \to \overline{v_i}\#$.

Proof of Claim 2: Obviously, for every i, $1 \leq i \leq n$, $\mathfrak{D}(\alpha_i) = \#\overline{v_i}$, which means that $|\alpha_i| = 1$ implies that α_i is a nonterminal with derivative $\#\overline{v_i}$. We now assume that $|\alpha_i| \geq 2$ for some i, $1 \leq i \leq n$. If we substitute α_i, by a new nonterminal $\overleftarrow{V_i}$ with a rule $\overleftarrow{V_i} \to \#\overline{v_i}$, then we shorten the axiom by at least $2\lceil \log(n) \rceil + 3$ and the size of the new rule is $|\#\overline{v_i}| = \lceil \log(n) \rceil + 1$; thus, the overall size of the grammar does not increase. An analogous argument applies if $|\alpha'_i| \geq 2$ for some i, $1 \leq i \leq n$. Consequently, we can assume that we have $\overleftarrow{V_i}, \overrightarrow{V_i} \in N$ with rules $\overleftarrow{V_i} \to \#\overline{v_i}$ and $\overrightarrow{V_i} \to \overline{v_i}\#$, and $\alpha_i = \overleftarrow{V_i}$, $\alpha'_i = \overrightarrow{V_i}$, $1 \leq i \leq n$. (*Claim* 2) □

We recall that, for every i, $1 \leq i \leq n$, $\mathfrak{D}(\beta_i) = \#\overline{v_i}\#$. Hence, if, for some i, $1 \leq i \leq n$, $|\beta_i| \geq 2$, then we can as well replace β_i by $\overleftarrow{V_i}\#$ without increasing the size of the grammar. This implies that, for every i, $1 \leq i \leq n$, $\beta_i = \overleftarrow{V_i}\#$ or $\beta_i = \overleftrightarrow{V_i}$ with $\overleftrightarrow{V_i} \to \#\overline{v_i}\#$.

Next, recall that, for every j, $1 \leq j \leq m$, $\mathfrak{D}(\gamma_i) = \# \, \overline{v_{j_{2i-1}}} \, \# \, \overline{v_{j_{2i}}} \, \#$. If, for some i, $1 \leq i \leq n$, $|\gamma_i| \geq 3$, then we can as well replace γ_i by $\overleftarrow{V}_{j_{2i-1}} \overleftarrow{V}_{j_{2i}} \#$

without increasing the size of the grammar. If $|\gamma_i| = 1$, then there is a rule $E \to \#\overline{v_{j_{2i-1}}}\#\overline{v_{j_{2i}}}\#$ of size $2\lceil\log(n)\rceil+3$. If we now replace γ_i by $\overleftarrow{V}_{j_{2i-1}}\overleftarrow{V}_{j_{2i}}\#$, then we increase the size of the axiom (and therefore of the grammar) by 4. However, since there are no other occurrences of $\#\overline{v_{j_{2i-1}}}\#\overline{v_{j_{2i}}}\#$ in w, there are no other occurrences of E in the axiom; thus, we can remove the rule $E \to \#\overline{v_{j_{2i-1}}}\#\overline{v_{j_{2i}}}\#$, which decreases the size of the grammar by $2\lceil\log(n)\rceil+3 \geq 4$. Hence, the overall size of the grammar does not increase. If $|\gamma_i| = 2$, then $\gamma_i = E_1E_2$ with $E_1 \to \#\overline{v_{j_{2i-1}}}\#x$ or $E_2 \to x\#\overline{v_{j_{2i}}}\#$. Let us assume that there is a rule $E_1 \to \#\overline{v_{j_{2i-1}}}\#x$ (the case $E_2 \to x\#\overline{v_{j_{2i}}}\#$ is analogous). If we now change this rule to $E_1 \to \#\overline{v_{j_{2i-1}}}\#$ and substitute every E_2 by $\overrightarrow{V}_{j_{2i}}$, then the size of the grammar does not increase (note that the nonterminals E_1 and E_2 can only occur in some γ_j, which has been replaced in this way).

These considerations demonstrate that we can assume that, in addition to the rule $D \to [\diamond]$, the rules of G are $\overleftarrow{V_i} \to \#\overline{v_i}$, $\overrightarrow{V_i} \to \overline{v_i}\#$, $1 \leq i \leq n$, and rules $\overleftrightarrow{V_i} \to \#\overline{v_i}\#$ with $i \in \mathfrak{I}$, for some $\mathfrak{I} \subseteq \{1, 2, \ldots, n\}$. We now define $\ell = |\mathfrak{I}|$ and the vertex set $\mathcal{V} = \{v_i \colon i \in \mathfrak{I}\}$; furthermore, let t be the number of edges from $\mathcal{G}$ that are covered by some vertex of $\mathcal{V}$. The axiom has the following form:

$$\mathsf{ax} = \prod_{i=1}^{n}(\overleftarrow{V_i}\, D\, \overrightarrow{V_i}\, D)^{2\lceil\log(n)\rceil+3} \prod_{i=1}^{n}(y_i\, D)^{\lceil\log(n)\rceil+1} \prod_{i=1}^{m}(z_i\, D)^2 \,\star\, D^{n^3}\,,$$

where, for every i, $1 \leq i \leq n$, $y_i = \overleftrightarrow{V_i}$ if $v_i \in \mathcal{V}$ and $y_i = \overleftarrow{V_i}\#$ otherwise, and, for every i, $1 \leq i \leq m$, $z_i = \overleftarrow{V}_{j_{2i-1}}\overleftarrow{V}_{j_{2i}}\#$, if the edge $(v_{j_{2i-1}}, v_{j_{2i}})$ is not covered by $\mathcal{V}$, $z_i = \overleftrightarrow{V}_{j_{2i-1}}\overrightarrow{V}_{j_{2i}}$ or $z_i = \overleftarrow{V}_{j_{2i-1}}\overleftrightarrow{V}_{j_{2i}}$, if $v_{j_{2i-1}} \in \mathcal{V}$ or $v_{j_{2i}} \in \mathcal{V}$, respectively.

The total size of the rules is

$$2n\lceil\log(n)\rceil + 2n + \ell\lceil\log(n)\rceil + 2\ell + n^3\,.$$

Moreover,

$$\begin{aligned}|\mathsf{ax}| &= 4n(2\lceil\log(n)\rceil + 3) + (\lceil\log(n)\rceil + 1)(3n - \ell)) + 6t + 8(m - t) + 1 + n^3\\ &= 11n\lceil\log(n)\rceil + 15n - \ell\lceil\log(n)\rceil - \ell + 8m - 2t + 1 + n^3\,.\end{aligned}$$

Consequently, $|G| = 13n\lceil\log(n)\rceil + 17n + \ell + 8m - 2t + 1 + 2n^3$. Since, by assumption, $|G| \leq 13n\lceil\log(n)\rceil + 17n + k + 6m + 1 + 2n^3$, we conclude that $\ell + 8m - 2t \leq k + 6m$. From this inequality, since $t \leq m$, we can deduce $\ell \leq k$ on the one hand and also $m - \frac{k-\ell}{2} \leq t$ on the other.

Consequently, the vertex set $\mathcal{V}$ covers already $m - \frac{k-\ell}{2}$ edges of $\mathcal{G}$. This implies that we can extend $\mathcal{V}$ to a vertex cover $\mathcal{V}'$ for $\mathcal{G}$ by adding q vertices, where $q \leq \frac{k-\ell}{2} \leq k - \ell$. Since $|\mathcal{V}| = \ell$, $|\mathcal{V}'| \leq |\mathcal{V}| + q \leq \ell + k - \ell = k$. □

From Lemmas 2 and 3, we can directly conclude the following theorem:

Theorem 2 1-SGP *is* NP-*complete, even for* $|\Sigma| = 5$.

3.2 The Multi-Level Case

In the above reduction for the 1-level case, the main difficulty is the use of unary factors as separators. However, once those separators are in place, we know the factors of w that are produced by nonterminals and, for a smallest 1-level grammar, this already fully determines the axiom and therefore also the grammar itself. For the multi-level case, the situation is much more complicated. Even if we manage to force the axiom to factorise w into parts that are either separators or codewords of vertices, this only determines the top-most level of the grammar and we do not necessarily know how these single factors are further hierarchically compressed and, more importantly, the dependencies between these compressions (i. e., how they share the same rules).

To deal with these issues, we rely on a larger alphabet Σ and we use palindromic codewords $u\star u^R$, where $\star \in \Sigma$ and u is a word over an alphabet of size 7 representing a 7-ary number. The purpose of the palindromic structure is twofold. Firstly, it implies that codewords always start and end with the same symbol, which, in the construction of w, makes it easier to avoid the situation that an overlapping between neighbouring codewords is repeated elsewhere in w (see Lemma 4). Secondly, if all codewords are produced by individual nonterminals, then we can show that they are produced best "from the middle", similar to the rules of the example grammar G_2 from Section 2.3. In addition to this, we also need a vertex colouring and an edge colouring of certain variants of the graph to be encoded.

In order to formally define the reduction, we first give some preparatory definitions. Let

$$\Sigma = \{x_1, \dots, x_7, d_1, \dots, d_7, \star, \#, ¢_1, ¢_2, \$_1, \dots, \$_6\}$$

be an alphabet of size 24. The function $M \colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ is defined by

$$M(q, k) := \min\{r > 0 \colon \exists\, t \in \mathbb{N} \colon q = tk + r\}$$

(note that M is the positive modulo-function, i. e., $M(q, k) = q\%k$, if $q\%k \neq 0$ and $M(q, k) = k$, otherwise). Let the functions $f \colon \mathbb{N} \to \{x_1, \dots, x_7\}^+$ and $g \colon \mathbb{N} \to \{d_1, \dots, d_7\}^+$ be defined by

$$\begin{aligned} f(q) :=& x_{a_0} x_{a_1} \dots x_{a_k} \text{ and} \\ g(q) :=& d_{a_0} d_{a_1} \dots d_{a_k}\,, \end{aligned}$$

for every $q \in \mathbb{N}$, where $k \in \mathbb{N} \cup \{0\}$ and $a_i \in \{1, 2, \dots, 7\}$, $0 \leq i \leq k$, such that $q = \sum_{i=0}^{k} a_i 7^i$ is satisfied. Note that since, for every $q \in \mathbb{N}$, there are unique $k \in \mathbb{N}$ and $a_i \in \{1, 2, \dots, 7\}$, $1 \leq i \leq k$, such that $q = \sum_{i \geq 0}^{k} a_i 7^i$, the functions f and g are well-defined.

For every $i \in \mathbb{N}$, let $\langle i \rangle_v := f(i) \star f(i)^R$ and $\langle i \rangle_\diamond := g(i) \star g(i)^R$. The factors $\langle i \rangle_v$ and $\langle i \rangle_\diamond$ are called *codewords*; $\langle i \rangle_v$ represents a vertex v_i, while the $\langle i \rangle_\diamond$ are used as separators.

Observation 1 *The functions f and g are bijections and they are 7-ary representations of the integers $n > 0$ (least significant digit first). Thus, for any $n \in \mathbb{N} \cup \{0\}$, $g(7n+i)[1] = d_i$ and $f(7n+i)[1] = x_i$, $1 \leq i \leq 7$. In particular, this means that $\{g(n+i)[1] \colon 0 \leq i \leq 6\} = \{d_1, \ldots, d_7\}$ and $\{f(n+i)[1] \colon 0 \leq i \leq 6\} = \{x_1, \ldots, x_7\}$, for every $n \in \mathbb{N}$. Consequently, for every $n, n' \in \mathbb{N}$ with $M(n,7) \neq M(n',7)$, the factors $\langle n \rangle_v$ and $\langle n' \rangle_v$ do not share any prefixes or suffixes (and the same holds for the words $\langle n \rangle_\diamond$).*

Let $\mathcal{G} = (V, E)$ be a subcubic graph (i. e., a graph with maximum degree 3) with $V = \{v_1, \ldots, v_n\}$ and $E = \{\{v_{j_{2i-1}}, v_{j_{2i}}\} \colon 1 \leq i \leq m\}$ (note that the vertex cover problem remains NP-hard if restricted to subcubic graphs (see [26])). Let $\mathcal{G}' = (V, E')$ be the multi-graph defined by

$$E' := \big\{\{v_{j_{2i}}, v_{j_{2i+1}}\} \colon 1 \leq i \leq m-1\big\} \ .$$

By [55], it is possible to compute in polynomial time a proper edge-colouring (meaning a colouring such that no two edges which share one or two vertices have the same colour) for a multi-graph with at most $\lfloor \frac{3}{2} m \rfloor$ colours, where m is the maximum degree of the multi-graph. Since the graph $\mathcal{G}$ is subcubic, the maximum degree of $\mathcal{G}'$ is three and we can compute a proper edge-colouring $C_e \colon E' \to \{1,2,3,4\}$ for $\mathcal{G}'$ with colours {1,2,3,4}. Let $\mathcal{G}^2 = (V, E'')$ be the graph defined by

$$E'' = \{\{u, v\} \colon \ \{u, w\}, \{w, v\} \in E \text{ for some } w \in V \backslash \{u, v\}, u \neq v\} \ .$$

Since $\mathcal{G}$ is subcubic, $\mathcal{G}^2$ has maximum degree at most six. Let $C_v \colon \{1, \ldots, n\} \to \{1,2,3,4,5,6,7\}$ be a proper vertex-colouring (defined over the vertex-indices of $V = \{v_1, \ldots, v_n\}$) for $\mathcal{G}^2$ with colours $\{1,2,3,4,5,6,7\}$. Such a colouring can be computed by an algorithmic version of Brook's theorem [56].

Let $w_{\mathcal{G}} = uvw$ be the word representing $\mathcal{G}$, where $u, v, w \in \Sigma^+$ are defined as follows (note that $m \leq \frac{3n}{2}$, so $7m < 14n$ in the word w).

$$u = \prod_{j=0}^{6} \left(\prod_{i=1}^{14n} (\langle i \rangle_\diamond \ \langle M(i+j, 14n) \rangle_v) \right) \ \$_1$$

$$\begin{aligned} v = & \prod_{i=1}^{n} (\# \ \langle 7i + C_v(i) \rangle_v \ ¢_1 \ \langle 7i-1 \rangle_\diamond) \ \$_2 \ \prod_{i=1}^{n} (\# \ \langle 7i + C_v(i) \rangle_v \ ¢_2 \ \langle 7i-2 \rangle_\diamond) \ \$_3 \\ & \prod_{i=1}^{n} (\langle 7i + C_v(i) \rangle_v \ \# \ \langle 7i-2 \rangle_\diamond \ ¢_1) \ \$_4 \ \prod_{i=1}^{n} (\langle 7i + C_v(i) \rangle_v \ \# \ \langle 7i-1 \rangle_\diamond \ ¢_2) \ \$_5 \\ & \prod_{i=1}^{n} (\# \ \langle 7i + C_v(i) \rangle_v \ \# \ \langle 7i \rangle_\diamond) \ \$_6 \end{aligned}$$

$$w = \prod_{i=1}^{m-1} (\# \, \langle 7j_{2i-1} + C_v(j_{2i-1})\rangle_v \, \# \, \langle 7j_{2i} + C_v(j_{2i})\rangle_v \, \# \, \langle 7i + C_e(v_{j_{2i}}, v_{j_{2i+1}})\rangle_\diamond)$$
$$\# \, \langle 7j_{2m-1} + C_v(j_{2m-1})\rangle_v \, \# \, \langle 7j_{2m} + C_v(j_{2m})\rangle_v \, \#$$

This concludes the definition of the reduction. Since the following proof of correctness is very complicated, we first present a corresponding "road-map", to make it more accessible:

- First, and completely independent from the question of how a grammar could compress $w_{\mathcal{G}}$, we take a closer look at the structure of this word. More precisely, in Propositions 1 and 2, we show that if a factor of $w_{\mathcal{G}}$ spans over the symbol $\star$ of some codeword $\langle i\rangle_v$ or $\langle i\rangle_\diamond$ and also reaches over the boundaries of this codeword into some other factor, then it is not repeated in $w_{\mathcal{G}}$. This property is the main reason for the complicated structure of $w_{\mathcal{G}}$ (especially the factor v).
- An immediate consequence of the property described in the previous point, is that in a smallest grammar, any nonterminal that derives a factor with an occurrence of $\star$ necessarily derives a factor that is completely contained in some codeword $\langle i\rangle_\diamond$ or in some codeword $\langle i\rangle_v$ delimited by two occurrences of the symbol # (see Lemma 4).
- Next, we show that we can assume that in a smallest grammar, there are nonterminals that have exactly our codewords as derivatives (see Lemma 5).
- The next result (Lemma 6) states that we can also assume that in a smallest grammar there are nonterminals with derivative $\#\langle 7i + C_v(i)\rangle_v$ and nonterminals with derivative $\langle 7i + C_v(i)\rangle_v\#$.
- Finally, we are able to fix the structure of a smallest grammar (Lemma 7) and we can show that, just like in the reduction from [14,39] (see Page 16), the set of rules that derive factors of the form $\#\langle 7i + C_v(i)\rangle_v\#$ can be transformed into a vertex cover (see Lemma 8).

The following simple, but crucial observation shall be helpful throughout the proof of correctness:

Observation 2 *The word $w_{\mathcal{G}}$ contains each of the symbols $\$_1, \ldots, \$_6$ exactly once, which implies that any smallest grammar for $w_{\mathcal{G}}$ has an axiom of the form $\prod_{i=1}^{6}(\beta_i \$_i)\beta_7$, $\beta_i \in ((V \cup \Sigma) \setminus \{\$_1, \ldots, \$_6\})^+$, $1 \leq i \leq 7$.*

We now prove the two propositions that establish the property with respect to the repetitions of factors containing $\star$.

Proposition 1 *For every i, $1 \leq i \leq 14n$, and j, $1 \leq j \leq 7$, the word $w_{\mathcal{G}}$ contains at most one occurrence of a factor of the form*

$$\star\, f(i)^R d_j, \qquad d_j f(i)\star, \qquad \star\, g(i)^R x_j, \qquad x_j g(i)\star\ .$$

Furthermore, if such a factor occurs in $w_{\mathcal{G}}$, then the occurrence is in u.

Proof We first note that factors of the form stated in the lemma can only occur in factors of the form $\langle i\rangle_v \, \langle i'\rangle_\diamond$ or $\langle i\rangle_\diamond \, \langle i'\rangle_v$. Since such factors only occur in u, the second statement of the proposition holds.

We first take care of factors of the form $\langle i\rangle_v d_{j'}$, $1 \leq i \leq 14n$, $1 \leq j' \leq 7$. These factors are subwords of $\langle M(x+j, 14n)\rangle_v \, \langle x+1\rangle_\diamond$ for some $j \in \{0, \dots, 6\}$ and x such that $i = M(x+j, 14n)$, which for each choice of pair (j, x) occur at most once in u. For every i, $6 < i \leq 14n$, this gives the seven choices $(j, i-j)$ with $0 \leq j \leq 6$; note that $i = M(x+j, 14n)$ implies $x = i-j$. This shows that the word u contains the subword $\langle i\rangle_v g(x+1)[1] = \langle i\rangle_v g(i-j+1)[1]$ once for each j, $0 \leq j \leq 6$, and these are the only occurrences of a subword of the form $\langle i\rangle_v d_{j'}$ for some $j' \in \{1, \dots, 7\}$ in u. Since $\{g(i-j+1)[1] \colon 0 \leq j \leq 6\} = \{d_1, \dots, d_7\}$ by Observation 1, it follows that no subword of the form $\langle i\rangle_v d_{j'}$ with $j' \in \{1, \dots, 7\}$ appears in u more than once. For every i, $1 \leq i \leq 6$, the choices of pairs (j, x) shift x by taking the modulo and are $(j, i-j)$ for $0 \leq j < i$ and $(j, 14n-j+i)$ for $i \leq j \leq 6$. The word u hence contains the subword $\langle i\rangle_v g(i-j+1)[1]$ once for each j, $0 \leq j < i$, the subword $\langle i\rangle_v g(14n-j+i+1)[1]$ once for each j, $i \leq j \leq 6$, and these are the only occurrences of a subword of the form $\langle i\rangle_v d_{j'}$ for some $j' \in \{1, \dots, 7\}$ in u. By reducing the $14n$ modulo 7 to zero, shifting by $+7$ and substituting j by $7-r$ we get that $\{g(14n-j+i+1)[1] \colon i \leq j \leq 6\} = \{g(i+1+r)[1] \colon 1 \leq r \leq 7-i\}$ and $\{g(i-j+1)[1] \colon 0 \leq j < i\} = \{g(i+1+r)[1] \colon 7-i < r \leq 7\}$. By Observation 1 we can hence conclude that each subword of the form $\langle i\rangle_v d_{j'}$ with $j' \in \{1, \dots, 7\}$ appears in u mat most once. Note that for $i = 6$, the factor $\langle i\rangle_v g(14n-j+i+1)[1]$ for the only choice $j = 6$ does not show up, as in this case u ends and $\langle 6\rangle_v$ is followed by $\$_1$. Consequently, for every i, $1 \leq i \leq 14n$, every factor $\star \, f(i)^R \, d_j$, $1 \leq j \leq 7$, has at most one occurrence in u.

Analogously, we can show that, for every i, $1 \leq i \leq 14n$, every factor $d_j \, f(i) \star$, $1 \leq j \leq 7$, has at most one occurrence in u. More precisely, it is sufficient to observe that, for every $6 < i \leq 14n$, the word u contains the subword $g(i-j)[1]\langle i\rangle_v$ once for each j, $0 \leq j \leq 6$; for every $1 \leq i \leq 6$, the subword $g(i-j)[1]\langle i\rangle_v$ once for each j, $0 \leq j \leq i-1$, and the subword $g(14n-j)[1]\langle i\rangle_v$ once for each j, $0 \leq j \leq 6-i$. As before, these are the only occurrences of a subword of the form $d_{j'}\langle i\rangle_v$ for some $j' \in \{1, \dots, 7\}$ in u.

For every i, $1 \leq i \leq 14n$, there are exactly 7 factors of the form $\star \, g(i)^R \, x_j$, for some j, $1 \leq j \leq 7$. Let $\star \, g(i) \, x_{j_\ell}$, $1 \leq \ell \leq 7$, be these 7 factors. By the structure of u, we observe that $\{j_\ell \colon 1 \leq \ell \leq 7\} = \{x_1, x_2, \dots, x_7\}$, which directly implies that, for every i, $1 \leq i \leq 14n$, every factor $\star g(i)^R x_{j_\ell}$, $1 \leq \ell \leq 7$, has at most one occurrence in u. Analogously, we can show that, for every i, $1 < i \leq 14n$, every factor of the form $x_j \, g(i) \star$, $1 \leq j \leq 7$, has at most one occurrence in u. Finally, there are exactly 6 factors of the form $x_j \, g(1) \star$, $1 \leq j \leq 7$, namely the factors $f(14n)[1] \, g(1) \star$ and $f(j)[1] \, g(1) \star$, $1 \leq j \leq 5$. Since $\{f(14n)[1], f(j)[1] \colon 1 \leq j \leq 5\} = \{x_7, x_1, x_2, \dots, x_5\}$, it follows that every factor of the form $x_j \, g(1) \star$, $1 \leq j \leq 7$, has at most one occurrence in u. □

Proposition 2 *For every i, $1 \leq i \leq 14n$, and j, $1 \leq j \leq 7$, the word $w_{\mathcal{G}}$ contains at most one occurrence of a factor of the form*

$$\begin{array}{llll} \star\, g(i)^R y, & y g(i)\star, & \star\, f(i)^R z, & z f(i)\star, \\ d_j \# f(i)\star, & \star\, f(i)^R \# d_j, & \star\, f(i)^R \# x_j, & x_j \# f(i)\star, \end{array}$$

where $y \in \Sigma \setminus \{d_1, \ldots, d_7\}$ and $z \in \Sigma \setminus \{x_1, \ldots, x_7, \#\}$.

Proof We first consider the factors $\star g(i)^R y$ with $y \in \Sigma \setminus \{d_1, \ldots, d_7\}$. In the case $y \in \{x_1, \ldots, x_7\}$, Proposition 1 shows that such factors have at most one occurrence in $w_{\mathcal{G}}$. For $y \in \{\star, \#, ¢_1, ¢_2, \$_1, \ldots, \$_6\}$, there are occurrences of factors of the form $\star g(i)^R y$ in v and in w, but not in u. We note that each two occurrences of factors $\star g(i)^R y$ and $\star g(i')^R y'$ in w satisfy $i \neq i'$ and are therefore different. Moreover, all factors $\star g(i)^R y$ in w satisfy $g(i)[1] \in \{1, 2, 3, 4\}$ (this is due to the colouring C_e). We next observe that all factors $\star g(i)^R y$ in v satisfy $i \in \{7i', 7i' - 1, 7i' - 2 \colon i' \in \mathbb{N}\}$, which implies that for these factors, we have $g(i)[1] \in \{5, 6, 7\}$; thus, they all differ from the factors $\star g(i)^R y$ in w. Consequently, if a factor of the form $\star g(i)^R y$ repeats, then there must be individual occurrences of factors $\langle i \rangle_\diamond y$ and $\langle i \rangle_\diamond y'$ in v. This is only the case for $i = 7i' - 1$, but then there are exactly two such factors and with $y \in \{\#, \$_2\}$, $y' = ¢_2$, or for $i = 7i' - 2$, but then there are exactly two such factors and with $y \in \{\#, \$_3\}$, $y' = ¢_1$. This shows that each factor $\star g(i)^R y$ with $y \in \Sigma \setminus \{d_1, \ldots, d_7\}$ has at most one occurrence in $w_{\mathcal{G}}$. For the factors $y g(i)\star$ the argument is the same up to the point where we consider individual occurrences of factors $y\langle i \rangle_\diamond$ and $y'\langle i \rangle_\diamond$ in v. Again, this is only possible for $i = 7i' - 1$ or $i = 7i' - 2$, but in the first case, we have $y = ¢_1$, $y' = \#$, while in the second case, we have $y = ¢_2$, $y' = \#$.

We next turn to the factors $\star f(i)^R z$ with $z \in \Sigma \setminus \{x_1, \ldots, x_7, \#\}$. Again, Proposition 1 shows that for $y \in \{d_1, \ldots, d_7\}$ such factors have at most one occurrence in $w_{\mathcal{G}}$; thus, we consider the case $y \in \{\star, ¢_1, ¢_2, \$_1, \ldots, \$_6\}$. We first note that such factors have no occurrence in u. Moreover, for every i, $1 \leq i \leq 14n$, any factor of the form $\langle i \rangle_v y$ with $y \notin \{d_1, \ldots, d_7, x_1, \ldots, x_7\}$ has either no occurrence in vw, or exactly 5 occurrences in v and at most 3 occurrences in w (this is due to the fact that $\mathcal{G}$ is subcubic). However, y is equal to $\#$ for all but two of those occurrences, where one occurrence is with $y = ¢_1$ and the other with $y = ¢_2$. Consequently, each factor $\star f(i)^R z$ with $z \in \Sigma \setminus \{x_1, \ldots, x_7, \#\}$ has at most one occurrence in $w_{\mathcal{G}}$. The argument for the factors $z f(i)\star$ with $z \in \Sigma \setminus \{x_1, \ldots, x_7, \#\}$ is analogous, with the difference that the only two occurrences of a factor $y\langle i \rangle_v$ in v with $y \notin \{d_1, \ldots, d_7, x_1, \ldots, x_7, \#\}$ are once with $y \in \{\$_3, ¢_1\}$ and once with $y \in \{\$_4, ¢_2\}$.

We next consider the factors $d_j \,\#\, f(i) \star$ and first note that such a factor only occurs in a factor $\#\, \langle i \rangle_v$ that is preceded by a factor $\langle i' \rangle_\diamond$, for some i', $1 \leq i' \leq 14n$, and that such factors only occur in v or w. In v, there are either no or exactly 3 occurrences of $\#\, \langle i \rangle_v$. The first one is either a prefix of v or preceded by $\langle 7\ell - 1 \rangle_\diamond$, $1 \leq \ell \leq n$, the second is preceded by either $\$_2$ or $\langle 7\ell - 2 \rangle_\diamond$, $1 \leq \ell \leq n$, and the third one is preceded by either $\$_5$ or $\langle 7\ell \rangle_\diamond$, $1 \leq \ell \leq n$. Hence, these three occurrences are preceded by symbols

d_6, d_5 and d_7, respectively (or by symbols not in $\{d_1, \ldots, d_7\}$). Consequently, the factor $d_j \mathbin{\#} f(i) \star$ is not repeated in v and if it occurs, $j \in \{5, 6, 7\}$ holds. Next, we note that every $\# \langle i \rangle_v$ in w that is preceded by a $\langle i' \rangle_\diamond$, satisfies $i' = 7\ell + C_e(v_{j_{2\ell}}, v_{j_{2\ell+1}})$, and since the range of C_e is $\{1, 2, 3, 4\}$, this occurrence of $\# \langle i \rangle_v$ is preceded by symbol d_1, d_2, d_3 or d_4. Finally, we have to show that no $d_j \mathbin{\#} \langle i \rangle_v$ is repeated in w. To this end, we assume that $d_j \mathbin{\#} \langle i \rangle_v$ with $j \in \{1, 2, 3, 4\}$ is repeated. This implies that there are k, k', $1 \leq k < k' \leq m-1$, with $j_{2k-1} = j_{2k'-1} = i$, and, furthermore, $\langle 7(k-1) + C_e(v_{j_{2(k-1)}}, v_{j_{2(k-1)+1}}) \rangle_\diamond$ and $\langle 7(k'-1) + C_e(v_{j_{2(k'-1)}}, v_{j_{2(k'-1)+1}}) \rangle_\diamond$ both end with symbol d_j. Thus, $C_e(v_{j_{2(k-1)}}, v_{j_{2(k-1)+1}}) = C_e(v_{j_{2(k'-1)}}, v_{j_{2(k'-1)+1}}) = j$, which is a contradiction, since the edges $(v_{j_{2(k-1)}}, v_{j_{2(k-1)+1}})$ and $(v_{j_{2(k'-1)}}, v_{j_{2(k'-1)+1}})$ of $\mathcal{G}'$ are incident with the same vertex $v_{j_{2k-1}} = v_{j_{2k'-1}} = v_i$ and C_e is a proper edge colouring for $\mathcal{G}'$. Consequently, no $d_j \mathbin{\#} \langle i \rangle_v$ is repeated in w; thus, the word $w_{\mathcal{G}}$ contains at most one occurrence of a factor of the form $d_j \mathbin{\#} f(i) \star$.

In an analogous way, we can show that every factor of form $\star f(i)^R \mathbin{\#} d_j$ in v satisfies $j \in \{5, 6, 7\}$ and in w it satisfies $j \in \{1, 2, 3, 4\}$. That these factors do not repeat follows from the fact that $\star f(i)^R \#$ occurs at most 3 times in v (followed by the different symbols d_5, d_6 and d_7) and the repetitions of $\star f(i)^R \#$ in w are followed by distinct symbols from $\{d_1, d_2, d_3, d_4\}$ due to the proper edge colouring C_e of $\mathcal{G}'$. Thus, the word $w_{\mathcal{G}}$ contains at most one occurrence of a factor of the form $\star f(i)^R \mathbin{\#} d_j$.

For any i, $1 \leq i \leq 14n$, and j, $1 \leq j \leq 7$, the factor $\star f(i)^R \mathbin{\#} x_j$ only occurs in w and only in a factor of the form $\langle 7\ell + C_v(\ell) \rangle_v \# \langle 7\ell' + C_v(\ell') \rangle_v$, $1 \leq \ell, \ell' \leq n$, with $i = 7\ell + C_v(\ell)$ and $f(7\ell' + C_v(\ell'))[1] = x_j$. Hence, if $\star f(i)^R \mathbin{\#} x_j$ has two occurrences, then there are ℓ', ℓ'', $1 \leq \ell', \ell'' \leq n$, such that the vertices $v_{\ell'}$ and $v_{\ell''}$ are neighbours of v_ℓ (in $\mathcal{G}$), and $f(7\ell' + C_v(\ell'))[1] = f(7\ell'' + C_v(\ell''))[1] = x_j$, which implies $C_v(\ell') = C_v(\ell'') = j$. This is a contradiction to the fact that C_v is a proper vertex colouring for the graph $\mathcal{G}^2$. In an analogous way, it follows that the factor $x_j \mathbin{\#} f(i) \star$ is not repeated. □

Since a smallest grammar does not contain rules which produce a factor which is not repeated, Propositions 1 and 2 yield the following:

Lemma 4 *For every smallest grammar $G = (N, \Sigma, R, \mathsf{ax})$ for $w_{\mathcal{G}}$, $|\mathfrak{D}(A)|_\star \geq 1$ for some $A \in N$ implies that $\mathfrak{D}(A)$ is a factor of some $\# \langle 7i + C_v(i) \rangle_v \#$, $1 \leq i \leq n$, or a factor of some $\langle j \rangle_v$, $1 \leq j \leq 14n$, or a factor of some $\langle j \rangle_\diamond$, $1 \leq j \leq 14n$.*

The main consequence of Lemma 4 is that, in a smallest grammar, the axiom has a length of at least the number of occurrences of $\star$ in $w_{\mathcal{G}}$. This allows us to show that, without increasing the size of the grammar, the axiom can be restructured, such that each individual codeword is produced by its own nonterminal.

Lemma 5 *There is a smallest grammar G for $w_{\mathcal{G}}$ such that, for every i, $1 \leq i \leq 14n$, there is a nonterminal with derivative $\langle i \rangle_\diamond$ and a nonterminal with derivative $\langle i \rangle_v$.*

Proof Let $G = (N, \Sigma, R, \mathsf{ax})$ be a smallest grammar with $\mathfrak{D}(G) = w_{\mathcal{G}}$. We shall first show how G can be modified in such a way that, for every i, $1 \leq i \leq 14n$, there is a nonterminal with derivative $\langle i \rangle_\diamond$. To this end, we assume that for some $\mathfrak{I}_\diamond \subseteq \{1, 2, \ldots, 14n\}$ and every i, $1 \leq i \leq 14n$, there currently is a nonterminal in G with derivative $\langle i \rangle_\diamond$ if and only if $i \in \mathfrak{I}_\diamond$; furthermore, let $\overline{\mathfrak{I}_\diamond} = \{1, 2, \ldots, 14n\} \setminus \mathfrak{I}_\diamond$. For the sake of concreteness, for every $i \in \mathfrak{I}_\diamond$, let $\widehat{D}_i$ be the nonterminal with $\mathfrak{D}(\widehat{D}_i) = \langle i \rangle_\diamond$.

We now recursively define a set of rules $R_\diamond := \{r_{\diamond,i} \colon 1 \leq i \leq 14n\}$ for nonterminals D_i, $1 \leq i \leq 14n$, by $r_{\diamond,i} := D_i \rightarrow d_i \star d_i$, $1 \leq i \leq 7$, and $r_{\diamond,i} := D_i \rightarrow g(i)[1]\ D_{h(i)}\ g(i)[1]$, $8 \leq i \leq 14n$, where $h(i) := \frac{i-M(i,7)}{7}$. Obviously, $\mathfrak{D}(D_i) = \langle i \rangle_\diamond$, $1 \leq i \leq 14n$. We modify G by the following algorithm. For every $i = 1, 2, \ldots, 14n$, if $i \in \overline{\mathfrak{I}_\diamond}$, then we add the rule D_i from $R_\diamond$ to G, and if $i \in \mathfrak{I}_\diamond$, then we replace the rule $\widehat{D}_i \rightarrow \alpha$ by $D_i \rightarrow \alpha$. Furthermore, we can carry out an analogous modification with respect to derivatives $\langle i \rangle_v$. More precisely, we define $\mathfrak{I}_v \subseteq \{1, 2, \ldots, 14n\}$ to be such that, for exactly the $i \in \mathfrak{I}_v$, there is a nonterminal with derivative $\langle i \rangle_v$. Then, in the same way as above, we can add rules from the set $R_v := \{r_{v,i} \colon 1 \leq i \leq 14n\}$, where $r_{v,i} := V_i \rightarrow x_i \star x_i$, $1 \leq i \leq 7$, and $r_{v,i} := V_i \rightarrow f(i)[1]\ V_{h(i)}\ f(i)[1]$, $8 \leq i \leq 14n$, where $h(i) := \frac{i-M(i,7)}{7}$.

We denote this modified grammar by G' and note that, by the considerations from above, for every i, $1 \leq i \leq 14n$, G' contains nonterminals D_i and V_i with

$$\mathfrak{D}(D_i) = \langle i \rangle_\diamond \text{ and } \mathfrak{D}(V_i) = \langle i \rangle_v, 1 \leq i \leq 14n\,.$$

Moreover, since every rule from $R_\diamond$ and R_v has size 3, $|G'| = |G| + 3(|\overline{\mathfrak{I}_\diamond}| + |\overline{\mathfrak{I}_v}|)$. In the remainder of this proof, we show that this size increase can be compensated by using the new rules in order to significantly shorten the axiom. Hence, we obtain a smallest grammar, with the properties claimed in the lemma. To this end, we first measure the size of the axiom of the original grammar G.

Claim 1: $\mathsf{ax} = \prod_{i=1}^{6}(\beta_i \$_i)\beta_7$, where $\beta_i \in ((N \cup \Sigma) \setminus \{\$_1, \ldots, \$_6\})^+$, $1 \leq i \leq 7$, and β_1 contains at least $196n$ occurrences of symbols (terminal or nonterminal) that each produces exactly one occurrence of $\star$.

Proof of Claim 1: From Observation 2, it follows that $\mathsf{ax} = \prod_{i=1}^{6}(\beta_i \$_i)\beta_7$, $\beta_i \in ((N \cup \Sigma) \setminus \{\$_1, \ldots, \$_6\})^+$, $1 \leq i \leq 7$. Furthermore, β_1 contains at least $|u|_\star$ symbols (terminal or nonterminal), since otherwise at least two occurrences of $\star$ of u are produced by the same nonterminal, which is a contradiction to Lemma 4. Hence, β_1 contains at least $196n$ occurrences of symbols that each produces exactly one occurrence of $\star$. (*Claim* 1) □

Claim 2: There are at least $7\lceil \frac{|\overline{\mathfrak{I}_\diamond}|+|\overline{\mathfrak{I}_v}|}{2} \rceil$ occurrences of symbols in β_1 (terminal or nonterminal), each of which has a derivative without any occurrence of $\star$.

Proof of Claim 2: Let $i \in \overline{\mathfrak{I}_\diamond}$, i. e., there is no nonterminal with derivative $\langle i \rangle_\diamond$. Furthermore, a derivative that properly contains $\langle i \rangle_\diamond$ (and the corresponding nonterminal which occurs in β_1) contains an occurrence of $\star$ and occurrences of symbols from both sets $\{d_1, \dots, d_7\}$ and $\{x_1, \dots, x_7\}$, which contradicts Lemma 4. Consequently, each of the 7 occurrences of $\langle i \rangle_\diamond$ are produced by at least two symbols. Hence, for each of these 7 occurrences, there is one symbol producing a factor of $\langle i \rangle_\diamond$ containing the symbol $\star$ and a second symbol, which produces a factor of $\langle i \rangle_\diamond$ that contains symbols from $\{d_1, \dots, d_7\}$. Due to Lemma 4, this second symbol cannot also produce the next or preceding occurrence of $\star$. This means that for each $i \in \overline{\mathfrak{I}_\diamond}$, there exist 7 symbols that do not produce a symbol $\star$. In the same way, we can also conclude that for each $i \in \overline{\mathfrak{I}_v}$, there exist 7 symbols that do not produce a symbol $\star$. However, it is possible that these symbols in β_1 which do not produce a $\star$ coincide, i. e., such a symbol can produce parts of some $\langle i \rangle_\diamond$ with $i \in \overline{\mathfrak{I}_\diamond}$ and $\langle i' \rangle_v$, with $i' \in \overline{\mathfrak{I}_v}$. So we can only conclude that there are at least $7\lceil \frac{|\overline{\mathfrak{I}_\diamond}|+|\overline{\mathfrak{I}_v}|}{2} \rceil$ occurrences of symbols in β_1 that do not produce an occurrence of $\star$. (*Claim* 2) □

From these two claims, it follows that the axiom of G (and therefore the whole grammar G) has size of at least $196n+7\lceil \frac{|\overline{\mathfrak{I}_\diamond}|+|\overline{\mathfrak{I}_v}|}{2} \rceil$. We now change G' a second time (into G''), as follows. We replace β_1 in the axiom $\mathsf{ax}' = \prod_{i=1}^{6} (\beta_i \$_i)\beta_7$ of G' (note that Observation 2 implies that ax' must have this structure) by $\beta_1' = \prod_{j=0}^{6} \prod_{i=1}^{14n} D_i V_{M(i+j,14n)}$. We note that $|\beta_1| \geq 196n + 7\lceil \frac{|\overline{\mathfrak{I}_\diamond}|+|\overline{\mathfrak{I}_v}|}{2} \rceil$, whereas $|\beta_1'| = 196n$. Consequently,

$$\begin{aligned}
|G''| &= \underbrace{|G| + 3(|\overline{\mathfrak{I}_\diamond}| + |\overline{\mathfrak{I}_v}|)}_{|G'|} + |\beta_1'| - |\beta_1| \\
&\leq |G| + 3(|\overline{\mathfrak{I}_\diamond}| + |\overline{\mathfrak{I}_v}|) + 196n - \left(196n + 7\left\lceil \frac{|\overline{\mathfrak{I}_\diamond}| + |\overline{\mathfrak{I}_v}|}{2} \right\rceil\right) \\
&= |G| + 3(|\overline{\mathfrak{I}_\diamond}| + |\overline{\mathfrak{I}_v}|) - 7\left\lceil \frac{|\overline{\mathfrak{I}_\diamond}| + |\overline{\mathfrak{I}_v}|}{2} \right\rceil \\
&\leq |G| \,.
\end{aligned}$$

□

In the hardness proof from [14,39] for the case of unbounded alphabets (see Page 16), one simple, but crucial fact was that for every i, $1 \leq i \leq n$, we can assume that nonterminals for each factor $\# v_i$ and $v_i \#$ exist. By using the previously mentioned lemmas, we now show a similar statement for our reduction:

Lemma 6 *There is a smallest grammar G for $w_{\mathcal{G}}$ such that, for every i, $1 \leq i \leq n$, there is a nonterminal with derivative $\#\langle 7i+C_v(i)\rangle_v$ and a nonterminal with derivative $\langle 7i + C_v(i)\rangle_v\#$.*

Proof Let $G = (N, \Sigma, R, \mathsf{ax})$ be a smallest grammar for $w_{\mathcal{G}}$. By Lemma 5, we can assume that, for every i, $1 \leq i \leq 14n$, there is a nonterminal D_i with derivative $\langle i \rangle_\diamond$ and a nonterminal V_i with derivative $\langle i \rangle_v$.

Let ℓ be the total number of occurrences of symbols from $\{\star$, ¢$_1$, ¢$_2$, $\#$, $\$_1$, …, $\$_6\}$ in $w_{\mathcal{G}}$. We can conclude that $|\mathsf{ax}| \leq \ell$, since an axiom of length ℓ

can be obtained from $w_{\mathcal{G}}$ (without introducing any new rules) by replacing all occurrences of $\langle i\rangle_{\diamond}$ and $\langle i\rangle_{v}$ by D_i and V_i, respectively.

Let $N_{\mathsf{ax}} = \{A \colon A \in N, |\mathsf{ax}|_A \geq 1, |\mathfrak{D}(A)|_{\star} \geq 1\}$ and let $\Gamma = \{\star, ¢_1, ¢_2, \#\}$. Furthermore, for every i, $1 \leq i \leq 3$, $N_{\mathsf{ax},i} = \{A \colon A \in N_{\mathsf{ax}}, \sum_{x \in \Gamma} |\mathfrak{D}(A)|_x = i\}$. Since, for every $A \in N_{\mathsf{ax}}$, $\sum_{x \in \Gamma} |\mathfrak{D}(A)|_x > 3$ is a contradiction to Lemma 4, we can conclude that $\{N_{\mathsf{ax},1}, N_{\mathsf{ax},2}, N_{\mathsf{ax},3}\}$ is a partition of N_{ax}. Consequently, we can use this partition in order to estimate the length of the axiom in the following way: $|\mathsf{ax}| \geq \ell - \sum_{A \in N_{\mathsf{ax},2}} |\mathsf{ax}|_A - 2\sum_{A \in N_{\mathsf{ax},3}} |\mathsf{ax}|_A$ (note that each occurrence of some $A \in N_{\mathsf{ax},j}$, $j \in \{2,3\}$, is responsible for $|\mathsf{ax}|_A$ units of the size $|\mathsf{ax}|$, but also for exactly $j|\mathsf{ax}|_A$ occurrences of the total amount ℓ of symbols from $\{\star, ¢_1, ¢_2, \#, \$_1, \ldots, \$_6\}$). Moreover, also due to Lemma 4, for every $A \in N_{\mathsf{ax},2}$, $\mathfrak{D}(A) = \# f(7i + C_v(i)) \star r_i$ or $\mathfrak{D}(A) = r_i \star f(7i + C_v(i))^R \#$ with $|r_i| \leq |f(7i + C_v(i))|$ and, for every $A \in N_{\mathsf{ax},3}$, $\mathfrak{D}(A) = \#\langle 7i + C_v(i)\rangle_v \#$.

We now add to G, for every i, $1 \leq i \leq n$, the rules $\overleftarrow{V}_i \to \# V_{7i+C_v(i)}$ and $\overrightarrow{V}_i \to V_{7i+C_v(i)}\#$, and, for every $A \in N_{\mathsf{ax},3}$, we add the rule $\overleftrightarrow{V}_i \to \overleftarrow{V}_i\#$, where $\mathfrak{D}(A) = \#\langle 7i+C_v(i)\rangle_v\#$. Then, we replace ax by a new axiom ax' that is obtained from $w_{\mathcal{G}}$ in the following way. Every factor $\langle i\rangle_{\diamond}$ is replaced by D_i. For every occurrence of $\star$ in $w_{\mathcal{G}}$, if this occurrence of $\star$ is produced (according to ax) by a nonterminal $A \in N_{\mathsf{ax},3}$, which, since $\mathfrak{D}(A) = \#\langle 7i+C_v(i)\rangle_v\#$, implies that it is inside a factor $\#\langle 7i+C_v(i)\rangle_v\#$, then we replace $\#\langle 7i+C_v(i)\rangle_v\#$ by $\overleftrightarrow{V}_i$. All remaining factors of the form $\#\langle 7i + C_v(i)\rangle_v\#$ are replaced by $\overleftarrow{V}_i\#$. Then, all remaining factors $\#\langle 7i+C_v(i)\rangle_v$ and $\langle 7i+C_v(i)\rangle_v\#$ are replaced by $\overleftarrow{V}_i$ and $\overrightarrow{V}_i$, respectively (note that since there are no factors of the form $\#\langle 7i + C_v(i)\rangle_v\#$ left, this is unambiguous). We note that $|\mathsf{ax}'| = \ell - \sum_{i=1}^{n}(|\mathsf{ax}'|_{\overleftarrow{V}_i} + |\mathsf{ax}'|_{\overrightarrow{V}_i}) - 2\sum_{i=1}^{n}|\mathsf{ax}'|_{\overleftrightarrow{V}_i}$.

Next, we show that all the rules for the nonterminals of $N_{\mathsf{ax},2} \cup N_{\mathsf{ax},3}$ can be removed from the grammar. To this end, let $A \in N_{\mathsf{ax},2} \cup N_{\mathsf{ax},3}$, which means that $|\mathfrak{D}(A)|_{\#} \geq 1$. However, every occurrence of $\#$ of $w_{\mathcal{G}}$ that is produced by a rule (and is not already present in the new axiom ax'), is directly produced by $\overleftarrow{V}_i$, $\overrightarrow{V}_i$ or $\overleftrightarrow{V}_i$, i. e., it occurs on the right side of these rules and is not produced by means of any other nonterminal. Consequently, in the derivation of $w_{\mathcal{G}}$, the nonterminal A is not used and, therefore, its rule can be erased.

It only remains to show that the modified grammar is not larger than the original one, i. e., we have to compare $|\mathsf{ax}'|$ to $|\mathsf{ax}|$ show that the size increase of 2 caused by each added rule is compensated. For every new rule $\overleftrightarrow{V}_i \to \overleftarrow{V}_i\#$ (of cost 2), there is an $A \in N_{\mathsf{ax},3}$ with $\mathfrak{D}(A) = \#\langle 7i + C_v(i)\rangle_v\#$ (of cost at least 2), for which the rule is erased and all all occurrences of A in ax correspond to occurrences of some $\overleftrightarrow{V}_i$ in ax', hence $\sum_{i=1}^{n}|\mathsf{ax}'|_{\overleftrightarrow{V}_i} = \sum_{A \in N_{\mathsf{ax},3}} |\mathsf{ax}|_A$. For every new rule $\overleftarrow{V}_i \to \# V_{7i+C_v(i)}$ consider $\overleftarrow{I} := \{i \colon \mathfrak{D}(A) = \# f(7i + C_v(i)) \star r_i$ for some $A \in N_{\mathsf{ax},2}\}$. If $i \in \overleftarrow{I}$ we have removed at least one rule $A \to \alpha$ with $\mathfrak{D}(A) = \# f(7i + C_v(i)) \star r_i$ with $|\alpha| \geq 2$, so the cost for all rules $\overleftarrow{V}_i \to \# V_{7i+C_v(i)}$ with $i \in \overleftarrow{I}$ is compensated. Further, every occurrence of this A in ax yields an occurrence of $\overleftarrow{V_i}$ in ax'. If $i \notin \overleftarrow{I}$, then both occurrences of $\#\langle 7i + C_v(i)\rangle_v$ in the factor v of $w_{\mathcal{G}}$ are produced in ax by at least two nonterminals each. An analogous argument applies to the new rules $\overrightarrow{V_i} \to V_{7i+C_v(i)}\#$ with $\overrightarrow{I} := \{i\colon \mathfrak{D}(A) = r_i \star f(7i + C_v(i))^R\#$ for some $A \in N_{\mathsf{ax},2}\}$. This yields $\sum_{i=1}^{n}(|\mathsf{ax}'|_{\overleftarrow{V_i}} + |\mathsf{ax}'|_{\overrightarrow{V_i}}) \geq \sum_{A\in N_{\mathsf{ax},2}}|\mathsf{ax}|_A + 2(n - |\overleftarrow{I}|) + 2(n - |\overrightarrow{I}|)$. Together with $\sum_{i=1}^{n}|\mathsf{ax}'|_{\overleftrightarrow{V_i}} = \sum_{A\in N_{\mathsf{ax},3}}|\mathsf{ax}|_A$ we can conclude:

$$\begin{aligned}
|\mathsf{ax}'| &= \ell - \sum_{i=1}^{n}(|\mathsf{ax}'|_{\overleftarrow{V_i}} + |\mathsf{ax}'|_{\overrightarrow{V_i}}) - 2\sum_{i=1}^{n}|\mathsf{ax}'|_{\overleftrightarrow{V_i}}\\
&\leq \ell - \sum_{A\in N_{\mathsf{ax},2}}|\mathsf{ax}|_A - 2(n - |\overleftarrow{I}|) - 2(n - |\overrightarrow{I}|) - 2\sum_{A\in N_{\mathsf{ax},3}}|\mathsf{ax}|_A\\
&\leq |\mathsf{ax}| - 2(n - |\overleftarrow{I}|) - 2(n - |\overrightarrow{I}|)
\end{aligned}$$

Since every new rule for $\overleftarrow{V_i}$ or $\overrightarrow{V_i}$ is added at a cost of two, the difference between $|\mathsf{ax}'|$ and $|\mathsf{ax}|$ compensates for the additional rules $\overleftarrow{V_i} \to \#V_{7i+C_v(i)}$ with $i \notin \overleftarrow{I}$ and $\overrightarrow{V_i} \to V_{7i+C_v(i)}\#$ with $i \notin \overrightarrow{I}$. Recall further that the cost for the rules for $\overleftrightarrow{V_i}$ are compensated by deleting the rules in $N_{\mathsf{ax},3}$. Overall, the modified grammar is not larger than the original grammar. Furthermore, the new grammar has now the form stated in the lemma. □

Now, by the lemmas presented above, we are able to sufficiently pin down the structure of a smallest grammar for $w_{\mathcal{G}}$:

Lemma 7 *There is a smallest grammar G for $w_{\mathcal{G}}$ that contains all the rules*

- $R_\diamond := \{r_{\diamond,i}\colon 1 \leq i \leq 14n\}$, *with* $r_{\diamond,i} := D_i \to d_i \star d_i$, $1 \leq i \leq 7$, *and* $r_{\diamond,i} := D_i \to g(i)[1]\ D_{h(i)}\ g(i)[1]$, $8 \leq i \leq 14n$, *where* $h(i) := \frac{i-M(i,7)}{7}$,
- $R_v := \{r_{v,i}\colon 1 \leq i \leq 14n\}$, *with* $r_{v,i} := V_i \to x_i \star x_i$, $1 \leq i \leq 7$, *and* $r_{v,i} := V_i \to f(i)[1]\ V_{h(i)}\ f(i)[1]$, $8 \leq i \leq 14n$, *where* $h(i) := \frac{i-M(i,7)}{7}$,
- $\overleftarrow{V} := \{\overleftarrow{V_i} \to \#V_{7i+C_v(i)}\colon 1 \leq i \leq n\}$,
- $\overrightarrow{V} := \{\overrightarrow{V_i} \to V_{7i+C_v(i)}\#\colon 1 \leq i \leq n\}$,
- $\overleftrightarrow{V} := \{\overleftrightarrow{V_i} \to \#\overrightarrow{V_i}\colon i \in \mathfrak{I}\}$, *for some* $\mathfrak{I} \subseteq \{1, 2, \ldots, n\}$.

and an axiom $\mathsf{ax} = \prod_{i=1}^{6}(\beta_i \$_i)\beta_7$ *with*

$$\beta_1 = \prod_{j=0}^{6}\left(\prod_{i=1}^{14n}(D_i\ \ V_{M(i+j,14n)})\right), \qquad \beta_2 = \prod_{i=1}^{n}\left(\overleftarrow{V_i}\ \ \text{¢}_1\ \ D_{7i-1}\right),$$

$$\beta_3 = \prod_{i=1}^{n}\left(\overleftarrow{V_i}\ \ \text{¢}_2\ \ D_{7i-2}\right), \qquad \beta_4 = \prod_{i=1}^{n}\left(\overrightarrow{V_i}\ \ D_{7i-2}\ \ \text{¢}_1\right),$$

$$\beta_5 = \prod_{i=1}^{n}\left(\overrightarrow{V_i}\ \ D_{7i-1}\ \ \text{¢}_2\right),$$

$$\beta_6 = \prod_{i=1}^{n} (y_i \;\, D_{7i})\,, \text{ where for every } i,\, 1 \le i \le n,\, y_i = \begin{cases} \overleftrightarrow{V_i} & \text{if } i \in \mathfrak{I}, \\ \overleftarrow{V_i}\# & \text{otherwise,} \end{cases}$$

$$\beta_7 = \prod_{i=1}^{m-1} (y_i D_{7i+C_e(v_{j_{2i}}, v_{j_{2i+1}})}) y_m, \text{ where for every } i,\, 1 \le i \le m,$$

$$y_i \in \{\overleftrightarrow{V}_{j_{2i-1}} \overrightarrow{V}_{j_{2i}}, \overleftarrow{V}_{j_{2i-1}} \overleftrightarrow{V}_{j_{2i}}\} \text{ if } \{j_{2i-1}, j_{2i}\} \cap \mathfrak{I} \neq \emptyset,$$

$$y_i = \overleftarrow{V}_{j_{2i-1}} \overleftarrow{V}_{j_{2i}} \# \text{ otherwise.}$$

Proof Let G be a smallest grammar for $w_{\mathcal{G}}$. By Lemma 5, we can assume that, for every i, $1 \le i \le 14n$, there is a nonterminal D_i with derivative $\langle i \rangle_\diamond$ and a nonterminal V_i with derivative $\langle i \rangle_v$, and, by Lemma 6, we can assume that, for every i, $1 \le i \le n$, there is a nonterminal $\overleftarrow{V_i}$ with derivative $\#\, \langle 7i + C_v(i) \rangle_v$ and a nonterminal $\overrightarrow{V_i}$ with derivative $\langle 7i + C_v(i) \rangle_v \#$. Obviously, for every i, $1 \le i \le n$, we can substitute the rule for $\overleftarrow{V_i}$ by $\overleftarrow{V_i} \to \#\, V_i$ and the rule for $\overrightarrow{V_i}$ by $\overrightarrow{V_i} \to V_i\, \#$, without increasing the size of G.

Next, for every $V_j \to \alpha_j$ with $|\alpha_j| \ge 3$, we can replace $V_j \to \alpha_j$ by $V_j \to x_j \star x_j$, if $j \le 7$, and by $V_j \to f(j)[1]\, V_{h(j)}\, f(j)[1]$, if $8 \le j$, where $h(j) := \frac{j - M(j,7)}{7}$. This does not increase the size of G, since the size of the modified rules can only decrease and no new rules need to be added. Now let $j = \max\{i \colon 1 \le i \le 14n, V_i \to \alpha_i, |\alpha_i| = 2\}$. We can now again replace $V_j \to \alpha_j$ by $V_j \to x_j \,\star\, x_j$, if $j \le 7$, and by $V_j \to f(j)[1]\; V_{h(j)}\; f(j)[1]$, if $8 \le j$, where $h(j) := \frac{j - M(j,7)}{7}$, but now this operation increases the size of the grammar by 1, which, as shall be shown next, is compensated by removing a rule from the grammar. To this end, we note that $\alpha_j = A_j B_j$ and $\mathfrak{D}(A_j) = f(j) \,\star\, t_j$ or $\mathfrak{D}(B_j) = t_j \,\star\, f(j)^R$ for some $t_j \in \{x_1, \ldots, x_7\}^*$. Let us assume that $\mathfrak{D}(A_j) = f(j) \,\star\, t_j$ (the case $\mathfrak{D}(B_j) = t_j \,\star\, f(j)^R$ can be handled analogously); note that this particularly implies that $A_j \notin \{V_i \colon 1 \le i \le 14n\}$, since its derivative is not of the form $\langle i \rangle_v$. Since $f(j) \star t_j$ does not occur in any $\langle j' \rangle_v$ with $j' < j$, A_j is not involved in a production of any $\langle j' \rangle_v$ with $j' < j$. Moreover, A_j cannot occur on the right side of the rule for a $V_{j'}$ with $j < j'$, since, due to the maximality of j and the modifications from above, those only have nonterminals of the form V_i on the right side. Thus, A_j has no occurrence in any of the rules for the nonterminals V_i, $1 \le i \le 14n$. This means that A_j can only occur on the right side of some nonterminal with a derivative that is not a factor of some $\langle i \rangle_v$ and, since $|\mathfrak{D}(A_j)|_\star \ge 1$, with Lemma 4, we can further conclude that A_j can only occur on the right side of some nonterminal with a derivative $\#\langle i \rangle_v$, $\langle i \rangle_v\#$ or $\#\langle i \rangle_v\#$. The rules $\overleftarrow{V_i} \to \#\, V_i$ and $\overrightarrow{V_i} \to V_i\, \#$ have the derivatives $\#\langle i \rangle_v$ and $\langle i \rangle_v\#$, respectively, and their right sides do not contain A_j. Furthermore, if the right side of a nonterminal with derivative $\#\langle i \rangle_v\#$ contains A_j, we can replace it by $\overleftarrow{V_i}\#$ without increasing the size of the grammar. Consequently, we can assume that the nonterminal A_j is never used and therefore its rule can be removed. By repeating this argument, it follows that G contains all the rules R_v.

In a similar way, we can show that G contains all the rules $R_\diamond$ (in fact, the argument is simpler, since in this case, Lemma 4 together with the fact that A_j can only occur on the right side of some nonterminal with a derivative that is not a factor of some $\langle i\rangle_\diamond$ immediately implies that A_j does not occur on any right side).

We now assume that $\mathsf{ax} = \prod_{i=1}^{6}(\beta_i\$_i)\beta_7$ is the axiom of G. In the same way as in the proofs of Lemmas 5 and 6, we can conclude that $|\beta_1| \geq 196n$, $|\beta_\ell| \geq 3n$, $1 \leq \ell \leq 5$. Hence, replacing ax by $\mathsf{ax}' = \prod_{i=1}^{6}(\beta'_i\$_i)\beta'_7$ with

$$\beta'_1 = \prod_{j=0}^{6}\left(\prod_{i=1}^{14n}(D_i \;\; V_{M(i+j,14n)})\right), \qquad \beta'_2 = \prod_{i=1}^{n}\left(\overleftarrow{V}_i \;\; \text{¢}_1 \;\; D_{7i-1}\right),$$

$$\beta'_3 = \prod_{i=1}^{n}\left(\overleftarrow{V}_i \;\; \text{¢}_2 \;\; D_{7i-2}\right), \qquad \beta'_4 = \prod_{i=1}^{n}\left(\overrightarrow{V}_i \;\; D_{7i-2} \;\; \text{¢}_1\right),$$

$$\beta'_5 = \prod_{i=1}^{n}\left(\overrightarrow{V}_i \;\; D_{7i-1} \;\; \text{¢}_2\right),$$

does not increase the size of the grammar. We now consider β_6, which produces the word $v_6 = \prod_{i=1}^{n}(\# \; \langle 7i + C_v(i)\rangle_v \; \# \; \langle 7i\rangle_\diamond)$. We can conclude the following from Lemma 4. No two occurrences of $\star$ in v_6 can be produced by the same nonterminal; thus, $|\beta_6| \geq 2n$. Furthermore, the only factors that are repeated in $w_\mathcal{G}$ and that contain an occurrence of both $\star$ and $\#$ are factors of $\#\langle 7i + C_v(i)\rangle_v\#$. Hence, for every i, $1 \leq i \leq n$, if the factor $\#\langle 7i + C_v(i)\rangle_v\#$ in $\# \; \langle 7i + C_v(i)\rangle_v \; \# \; \langle 7i\rangle_\diamond$ is not produced by a single nonterminal, then there is an additional nonterminal in β_6 (i. e., in addition to the two nonterminals producing the two occurrences of $\star$ in $\# \; \langle 7i + C_v(i)\rangle_v \; \# \; \langle 7i\rangle_\diamond$). This implies that $|\beta_6| \geq 3n - p$, where p is the number of nonterminals with a derivative of $\#\langle 7i + C_v(i)\rangle_v\#$. This means that we can replace every such nonterminal and its rule by $\overleftrightarrow{V}_i \to \#\overrightarrow{V}_i$ without increasing the size of the grammar. Furthermore, again without increasing the size of the grammar, we can replace β_6 by $\prod_{i=1}^{n}(y_i \;\; D_{7i})$, where, for every i, $1 \leq i \leq n$, $y_i = \overleftrightarrow{V}_i$ if this nonterminal exists and $y_i = \overleftarrow{V}_i\#$ otherwise.

Next, we consider β_7, which produces the word

$$\begin{aligned} v_7 = \prod_{i=1}^{m-1} & (\#\langle 7j_{2i-1} + C_v(j_{2i-1})\rangle_v\#\langle 7j_{2i} + C_v(j_{2i})\rangle_v\#\langle 7i + C_e(v_{j_{2i}}, v_{j_{2i+1}})\rangle_\diamond) \\ & \#\langle 7j_{2m-1} + C_v(j_{2m-1})\rangle_v\#\langle 7j_{2m} + C_v(j_{2m})\rangle_v\# \,. \end{aligned}$$

Similar as for the word v_6, every occurrence of $\star$ in v_7 requires a distinct nonterminal and, in addition to that, also a distinct nonterminal for each factor $\#\langle 7i + C_v(i)\rangle_v\#$ that is not completely produced by a single nonterminal. Hence, $|\beta_7| \geq 4m - 1 - q$, where q is the number of nonterminals $\overleftrightarrow{V}_i$ used in β_7. Consequently, we can also replace β_7 by $v_7 = \prod_{i=1}^{m-1}(y_i D_{7i+C_e(v_{j_{2i}}, v_{j_{2i+1}})})y_m$, where, for every i, $1 \leq i \leq m$, $y_i \in \{\overleftrightarrow{V}_{j_{2i-1}}\overrightarrow{V}_{j_{2i}}, \overleftarrow{V}_{j_{2i-1}}\overleftrightarrow{V}_{j_{2i}}\}$, if $\overleftrightarrow{V}_{j_{2i-1}}$ or $\overleftrightarrow{V}_{j_{2i}}$

exist, and $y_i = \overleftarrow{V}_{j_{2i-1}}\overleftarrow{V}_{j_{2i}}\#$, otherwise. We note that this does not increase the size of the grammar.

The grammar has now the form claimed in the statement of the lemma (note that all other rules not mentioned in the statement of the lemma can be ignored, since they are not used anymore). □

Finally, we are able to conclude the proof of correctness by establishing the connection between the size of a smallest grammar for $w_{\mathcal{G}}$ and the size of a vertex cover for $\mathcal{G}$.

Lemma 8 *The graph $\mathcal{G}$ has a vertex cover of size k if and only if $w_{\mathcal{G}}$ has a grammar of size $299n + k + 3m + 5$.*

Proof Let Γ be a size-k vertex cover of $\mathcal{G}$. We construct the grammar described in Lemma 7 with respect to $\mathfrak{I} = \{i \colon v_i \in \Gamma\}$. Since Γ is a vertex cover, in the definition of β_7, we have $y_i \in \{\overleftrightarrow{V}_{j_{2i-1}}\overrightarrow{V}_{j_{2i}}, \overleftarrow{V}_{j_{2i-1}}\overleftrightarrow{V}_{j_{2i}}\}$, for every $1 \leq i \leq m$. Consequently, by simply counting the symbols on the right sides of the rules, we conclude $|G| = 299n + |\mathfrak{I}| + 3m + 5 = 299n + k + 3m + 5$.

On the other hand, if there is a grammar of size $299n + k + 3m + 5$ for $w_{\mathcal{G}}$, then, by Lemma 7, we can also assume that there exists a grammar G for $w_{\mathcal{G}}$ with $|G| = 299n + |\mathfrak{I}| + 3m + 5 \leq 299n + k + 3m + 5$ that has the form described in Lemma 7, with respect to some $\mathfrak{I} \subseteq \{1, 2, \ldots, n\}$. If, for some edge (v_i, v_j), $\{v_i, v_j\} \cap \mathfrak{I} = \emptyset$, then adding i to $\mathfrak{I}$ (and therefore the rule $\overleftrightarrow{V_i} \to \#\overrightarrow{V_i}$ to the grammar) does not increase the size of the grammar. This is due to the fact that the additional cost of 2 for introducing the rule is compensated by using $\overleftrightarrow{V_i}$ once in β_6 and once in β_7. Consequently, we can assume that $\Gamma = \{v_i \colon i \in \mathfrak{I}\}$ is a vertex cover. Since $|G| = 299n + |\mathfrak{I}| + 3m + 5 \leq 299n + k + 3m + 5$, this means that Γ is a vertex cover for $\mathcal{G}$ of size at most $|\mathfrak{I}| = k$. □

From Lemma 8, we directly conclude our main result:

Theorem 3 SGP *is* NP*-complete, even for alphabets of size* 24*.*

Obviously, Theorem 3 leaves some room for improvement with respect to smaller alphabet sizes. In our reduction, we did use terminal symbols economically, but, for reasons explained next, this was not our main concern. While we generally believe that the alphabet size can be slightly reduced in our reduction, we consider it very unlikely that its current structure allows a substantial improvement in this regard (e. g., an alphabet size below 10). Thus, we did not further pursue this point, which we expect to lead to an even more involved reduction while at the same time only insignificantly decreases the alphabet size. Consequently, the NP-hardness of the smallest grammar problem for small alphabets (with the most interesting candidates being 2 (i. e., binary strings) and 4 (due to the fact that DNA-sequences use a 4-letter alphabet)) remains open. Furthermore, we expect that completely new techniques are required for respective hardness reductions. In this regard, note that for alphabets of size 1, the smallest grammar problem is strongly connected to the problem of computing the smallest *addition chain* for a single integer; a problem that is neither known to be in P nor to be NP-hard (see [39] or Section 6 for details).

3.3 Extensions of the Reductions

In this section, we conclude several important hardness results by slight modifications of the reduction presented in Section 3.2. First, we show that the optimisation variant of the smallest grammar problem (over fixed alphabets) is APX-hard and therefore it does not allow for a polynomial-time approximation scheme, unless $\mathsf{P} = \mathsf{NP}$. Just like Theorem 3 lifts the known NP-hardness of the smallest grammar problem for unbounded alphabets to the practically relevant case of fixed alphabets, this APX-hardness result lifts the inapproximability result for unbounded alphabets of [14,39] to the fixed alphabet case. There is one caveat, though, which is that the corresponding constant lower bound on the approximation ratio is much lower than the already low 1.0001 achieved for unbounded alphabets; thus, we do not bother to actually compute it and we consider the value of the APX-hardness result that the existence of a PTAS is ruled out.

Theorem 4 $\mathrm{SGP}_{\mathsf{opt}}$ *is* APX*-hard, even for alphabets of size* 24.

Proof The reduction used for Theorem 3 can also be seen as an L-reduction from the optimisation variant of the minimum vertex cover problem restricted to cubic graphs (each vertex has degree 3), which remains APX-hard (see [2]). More precisely, this problem is denoted by $(I_{\mathrm{VC}}, S_{\mathrm{VC}}, m_{\mathrm{VC}})$, where I_{VC} is the set of undirected cubic graphs, $S_{\mathrm{VC}}(\mathcal{G}) = \{C \colon C \text{ is a vertex cover for } \mathcal{G}\}$ and $m_{\mathrm{VC}}(\mathcal{G}, C) = |C|$; we denote $\mathrm{SGP}_{\mathsf{opt}}$ by $(I_{\mathrm{SGP}}, S_{\mathrm{SGP}}, m_{\mathrm{SGP}})$.

Next, we describe an L-reduction from the problem $(I_{\mathrm{VC}}, S_{\mathrm{VC}}, m_{\mathrm{VC}})$ to the problem $(I_{\mathrm{SGP}}, S_{\mathrm{SGP}}, m_{\mathrm{SGP}})$. The above described translation of a graph $\mathcal{G}$ to the word $w_{\mathcal{G}}$ (i. e., the one defined in Section 3.2 in order to prove Theorem 3) gives the function f for the L-reduction. The function g, that maps $\mathcal{G} \in I_{\mathrm{VC}}$ and a grammar $G \in S_{\mathrm{SGP}}(f(\mathcal{G}))$ to a vertex cover $C \in S_{\mathrm{VC}}(\mathcal{G})$ works as follows. We first build a grammar G' with $|G'| \leq |G|$ which is of the form described in Lemma 7; observe that all transformations that are necessary to reach this kind of normal form are constructive and computable in polynomial time. Then $g(\mathcal{G}, G) = \{v_i \colon i \in \mathfrak{I}\}$, which is a vertex cover for $\mathcal{G}$ by Lemma 8 (note that the set $\mathfrak{I}$ is ensured by Lemma 7). Finally, we show that choosing $\beta = 613$ and $\gamma = 1$ satisfies the inequalities. To this end, we first note that, for any cubic graph $\mathcal{G}$ with n vertices and m edges, we have $m = \frac{3}{2}n$ (since each vertex has degree 3) and $m^*_{\mathrm{VC}}(\mathcal{G}) \geq \frac{n}{2}$ (since each vertex can cover at most three edges), and $m^*_{\mathrm{VC}}(\mathcal{G}) \geq 1$.

$$\begin{aligned}
m^*_{\mathrm{SGP}}(w_{\mathcal{G}}) &= 299n + 3m + 5 + m^*_{\mathrm{VC}}(\mathcal{G}) \\
&= 607 \cdot \frac{n}{2} + 5 + m^*_{\mathrm{VC}}(\mathcal{G}) \\
&\leq 607 \cdot m^*_{\mathrm{VC}}(\mathcal{G}) + 5 + m^*_{\mathrm{VC}}(\mathcal{G}) \\
&\leq 613 \cdot m^*_{\mathrm{VC}}(\mathcal{G}) = \beta \cdot m^*_{\mathrm{VC}}(\mathcal{G}) \, ,
\end{aligned}$$

for any $\mathcal{G} \in I_{\mathrm{VC}}$. Furthermore,

$$\begin{aligned} m_{\mathrm{VC}}(\mathcal{G}, g(\mathcal{G}, G)) - m^*_{\mathrm{VC}}(\mathcal{G}) &= (299n + 3m + 5 + m_{\mathrm{VC}}(\mathcal{G}, g(\mathcal{G}, G))) - \\ &\quad (299n + 3m + 5 + m^*_{\mathrm{VC}}(\mathcal{G})) \\ &= 1 \cdot (m_{\mathrm{SGP}}(w_{\mathcal{G}}, G) - m^*_{\mathrm{SGP}}(w_{\mathcal{G}}))\,, \end{aligned}$$

for any $\mathcal{G} \in I_{\mathrm{VC}}$ and $G \in S_{\mathrm{SGP}}(w_{\mathcal{G}})$. □

Next, we take a closer look at the rule-size measure of grammars, i. e., at the problems SGP_r and 1-SGP_r. As defined in Section 2.2, the rule-size also takes the number of rules into account. In fact, the literature on grammar-based compression is inconsistent with respect to which kind of size is used, e. g., in [5,14,39,39,35,62,41], the size of a grammar coincides with our definition $|\cdot|$, while in [51,7,23,13], the rule-size is used. The rule-size seems to be mainly motivated by the question of how a grammar is encoded as a single string, which, in any reasonable way, requires an additional symbol per rule.[9] In many contexts, the difference between size and rule-size of grammars seems negligible, but, formally, the problems SGP and SGP_r (as well as 1-SGP and 1-SGP_r) are different decision problems and hardness results do not automatically carry over from one to the other. Since the existing literature suggests that the rule-size is of interest as well, we consider it a worthwhile task to extend our hardness results accordingly.

It seems intuitively clear that the size increase caused by measuring with the rule-size does not have an impact on the complexity of the smallest grammar problem. In fact, the arguments in the proof for Theorem 2 for the 1-level case also apply for the rule-size, but with an addition of $2n + k + 2$ (i. e., the number of rules) to the size of an r-smallest grammar. This is due to the fact that the rules that are introduced in the proof of Lemma 3 also shorten the grammar with respect to the rule-size measure.

Theorem 5 1-SGP_r *is* NP*-complete, even for even for alphabets of size* 5.

In the multi-level case, however, the situation is not so simple. In particular, in the proof of Theorem 3, there are some arguments, which do not apply for the rule-size. For example, a rule which only compresses a factor of length two is only profitable (with respect to the rule-size) if it can be used at least three times, which is problematic, since the rules which correspond to the vertex cover have length two and, in case the vertex only covers one edge, compress factors which only occur twice. Beside these problems, already in Lemma 5, we can see that it is hard to prove that the rule-size of the desired grammar G'' is smaller than $|G|_{\mathsf{r}}$ as we now have to pay a cost of 4 for each rule V_i (or D_i) with $i \notin \mathfrak{I}_v$ (or $i \notin \mathfrak{I}_\diamond$) which cannot be compensated by shortening the axiom for u only by $7\lceil \frac{|\mathfrak{I}_\diamond| + |\mathfrak{I}_v|}{2} \rceil$.

[9] For example, a grammar can be formed into a single string by using an order on the rules and then listing the right sides with separators in between, or by listing the rules with the corresponding nonterminals.

With a larger alphabet and certain repetitions of subwords of $w_{\mathcal{G}}$, we can modify the reduction to accommodate the rule-size, such that the arguments used for Theorem 3 still hold for this measure. To this end, we now encode $\langle i \rangle_v$ and $\langle i \rangle_\diamond$ over 8-ary instead of 7-ary alphabets $\{x_1, \ldots, x_8\}$ and $\{d_1, \ldots, d_8\}$, respectively, with analogous functions f and g. Let v' and w' be defined as v and w on page 23, but with respect to the new 8-ary codewords which only means that each occurrence of '7' in the definition of v and w is replaced by '8'. Moreover, let u' be defined as u on page 23, but with the '6' of the first product replaced by '7' and the '$14n$' of the second product replaced by '$24n + 4$' (the latter is necessary, since we need more separators of the form $\langle i \rangle_\diamond$). The colourings C_v and C_e remain unchanged.

In order to adapt the reduction to the rule-size measure, we have to repeat each factor $\#\langle 8i + C_v(i) \rangle_v$ and each factor $\langle 8i + C_v(i) \rangle_v \#$ once more, but in such a way that Proposition 2 still holds, which is done by using three new symbols $\$_7, \$_8$ and $¢_3$, and to add the following to v':

$$\begin{aligned} v'' = v' &\prod_{i=1}^{n} (\langle 8i + C_v(i) \rangle_v \; \# \; \langle 8i - 3 \rangle_\diamond \; ¢_3) \; \$_7 \\ &\prod_{i=1}^{n} (\# \; \langle 8i + C_v(i) \rangle_v \; ¢_3 \; \langle 8i - 3 \rangle_\diamond) \; \$_8 \, . \end{aligned}$$

In order to also repeat once more the factors $\#\langle 8j_{2i} + C_v(j_{2i}) \rangle_v \#$ to make covering edges profitable with respect to the rule-size, we repeat the complete list of edges, but every edge $(v_{j_{2i-1}}, v_{j_{2i}})$ is represented in reverse order as $\#\langle 8j_{2i} + C_v(j_{2i}) \rangle_v \# \langle 8j_{2i-1} + C_v(j_{2i-1}) \rangle_v \#$ to make sure that no subword of the form $\langle i \rangle_v \# x_j$ or $x_j \# \langle i \rangle_v$ is repeated. We further choose a new, previously not used set of separators $\langle i \rangle_\diamond$ (actually the $2m + 4$ more for which we created codewords with u) to make sure that each factor of the form $\langle i \rangle_\diamond \#$ or $\#\langle i \rangle_\diamond$ occurs at most once. We still chose the separators according to the edge-colouring to make sure that no factors of the form $\langle i \rangle_v \# d_j$ or $d_j \# \langle i \rangle_v$ are repeated; observe that by repeating the edges in reverse order, a factor of the form $\langle i \rangle_v \# d_j$ in w' becomes a factor of the form $d_j \# \langle i \rangle_v$ in the reverse listing. Formally, we define:

$$w'' = w' \; \tilde{w} \; \#\langle 8j_2 + C_v(j_2) \rangle_v \; \# \; \langle 8j_1 + C_v(j_1) \rangle_v \# \, ,$$

where

$$\begin{aligned} \tilde{w} = \prod_{i=m}^{2} (\; &\#\langle 8j_{2i} + C_v(j_{2i}) \rangle_v \# \langle 8j_{2i-1} + C_v(j_{2i-1}) \rangle_v \\ &\#\langle 8(i + m) + C_e(v_{j_{2(i-1)}}, v_{j_{2i-1}}) \rangle_\diamond) \, . \end{aligned}$$

Finally, we set $w'_{\mathcal{G}} = u'v''w''$.

It can be easily verified that Lemma 4 remains true for the new construction; observe that appending the new part of w'' yields the only occurrence of the factor ## (note that w' ends with #) which implies that the old and the new part are separated in the axiom of any r-smallest grammar for $w'_{\mathcal{G}}$. The equivalent to Lemma 5 also holds, since the part of the axiom for u' now has a length of at least $384n + 64 + 8\lceil \frac{|\overline{\mathfrak{I}_\diamond}|+|\overline{\mathfrak{I}_v}|}{2} \rceil + 1$ and the set of new rules, which now costs $4(|\overline{\mathfrak{I}_\diamond}| + |\overline{\mathfrak{I}_v}|)$, shortens this to $384n + 65$ (i. e., the number of occurrences of $\star$ in u' plus 1 for $\$_1$). Lemma 6 follows with the same arguments as before, just with 3 occurrences for each $\#\langle i \rangle_v$ and $\langle i \rangle_v\#$, which makes the rules for these subwords profitable even with respect to the rule-size. An analogue of Lemma 7 then follows exactly as before (the only addition is that the new parts of v'' and w'' are compressed in the obvious way by the existing rules). The following observation shall be helpful.

Observation 3 *If $\mathfrak{I} \subseteq \{1, 2, \ldots, n\}$ is such that $\{v_i : i \in \mathfrak{I}\}$ is a vertex cover, then the grammar for $w'_{\mathcal{G}}$ according to the adapted version of Lemma 7 with respect to $\mathfrak{I}$ (see the proof of Lemma 8) satisfies $|G| = 553n + |\mathfrak{I}| + 6m + 94$ and $|G|_{\mathsf{r}} = 603n + 2|\mathfrak{I}| + 6m + 103$ (note that for the rule-size, we also have to count the start rule, so the sizes differ by the number of rules which is $50n + k + 9$).*

An analogous statement of Lemma 8 can now be concluded as follows. For a size-k vertex cover Γ of $\mathcal{G}$, we set $\mathfrak{I} = \{i : v_i \in \Gamma\}$ and then construct a grammar G for $w'_{\mathcal{G}}$ according to the adapted version of Lemma 7 with respect to $\mathfrak{I}$ with $|G|_{\mathsf{r}} = 603n + 2|\mathfrak{I}| + 6m + 103$ (see Observation 3). On the other hand, if there is a grammar for $w'_{\mathcal{G}}$ of rule-size $603n + 2k + 6m + 103$, then, by the adapted version of Lemma 7, there is a grammar G for $w'_{\mathcal{G}}$ with $|G|_{\mathsf{r}} = 603n + 2|\mathfrak{I}| + 6m + 103 \leq 603n + 2k + 6m + 103$ that has the form given by the adapted version of Lemma 7, with respect to some $\mathfrak{I} \subseteq \{1, 2, \ldots, n\}$. If, for some edge (v_i, v_j), $\{v_i, v_j\} \cap \mathfrak{I} = \emptyset$, then the factors $\#\langle 8i + C_v(i) \rangle_v \# \langle 8j + C_v(j) \rangle_v \#$ and $\#\langle 8j + C_v(j) \rangle_v \# \langle 8i + C_v(i) \rangle_v \#$ in w'' each correspond to three symbols in the axiom, and the factor $\#\langle 8i + C_v(i) \rangle_v \#$ in v'' corresponds to two symbols in the axiom. Hence, introducing the rule $\overleftrightarrow{V_i} \to \#\overrightarrow{V_i}$ has a cost of three with respect to the rule-size and shortens the axiom by at least three. Consequently, as in the proof of Lemma 8, we can assume that $\Gamma = \{v_i : i \in \mathfrak{I}\}$ is a vertex cover. Since $|G|_{\mathsf{r}} = 603n + 2|\mathfrak{I}| + 6m + 103 \leq 603n + 2k + 6m + 103$, this means that Γ is a vertex cover for $\mathcal{G}$ of size at most k. Thus, we conclude that the graph $\mathcal{G}$ has a vertex cover of size k if and only if there exists a grammar of rule-size $603n + 2k + 6m + 103$ for $w'_{\mathcal{G}}$, which yields the following:

Theorem 6 SGP_r *is* NP*-complete, even for alphabets of size* 29*.*

Similar to Theorem 4, the above reduction can also be seen as an L-reduction (with the only change of setting $\beta = 1329$), which shows that the optimisation variant of the smallest grammar problem remains APX-hard under the rule-size measure.

Theorem 7 $\mathrm{SGP}_{\mathsf{r,opt}}$ *is* APX*-hard, even for alphabets of size* 29*.*

We conclude that if we change from the normal size measure to the rule-size measure, NP- and APX-hardness of the smallest grammar problem over fixed alphabets remains, although the smallest alphabet size in our constructions is slightly larger. We conclude this section by another interesting observation that follows from the rule-size variant of our reduction.

Obviously, the modified reduction to SGP_r can also be interpreted as a reduction to SGP. While, on first glance, this only seems to yield a weaker hardness result compared to the one of Theorem 3, it has a nice feature that entails an interesting result in its own right. More precisely, with respect to the modified reduction and the normal size measure, every rule from Lemma 7 has a positive profit (i.e., replacing all occurrences of the nonterminal by the right side of the rule would increase the overall size) and, furthermore, every rule added in the proofs of Lemmas 5 and 6 yields a strictly smaller grammar (note that this directly follows from the correctness of the construction for the rule-size measure). Moreover, there are no repeated substrings in the grammar with this set of rules which means that no additional rules with nonnegative profit can be added. Consequently, we have not only determined the size of a smallest (with respect to $|\cdot|$) grammar G for $w'_{\mathcal{G}}$ to be $553n + k + 6m + 94$, where k is the size of a smallest vertex cover for $\mathcal{G}$ (see Observation 3), but also that G requires exactly $|G|_{\mathsf{r}} - |G| = 50n + k + 9$ rules (or nonterminals). Hence, the modified reduction also serves as a reduction from the vertex cover problem to the following (weaker) variant of the smallest grammar problem:

Rule Number-SGP (RN-SGP)
Instance: A word w and a $k \in \mathbb{N}$.
Question: Does there exist a smallest grammar $G = (N, \Sigma, R, S)$ for w with $|N| \leq k$?

Theorem 8 RN-SGP *is* NP*-hard, even for alphabets of size* 29*.*

For the 1-level case, the original reduction already provides the analogous result (here, 1-RN-SGP denotes the variant of RN-SGP, where we ask whether there is a smallest 1-level grammar with $|N| \leq k$):

Theorem 9 1-RN-SGP *is* NP*-hard, even for alphabets of size* 5*.*

While the problems RN-SGP and 1-RN-SGP naturally arise in the context of grammar-based compression, they are particularly interesting in the light of the results presented in Section 4.1 and their relevance shall be discussed there in more detail.

3.4 (Limits of) Alphabet Reduction

As shall be discussed in this section, we can achieve a slight reduction of the alphabet size in Theorem 3. However, it seems rather unlikely that a substantial decrease is possible with our current general approach. In particular, it is suggested that a different approach is needed to prove the hardness of SGP for small, e. g., binary, alphabets.

We first note that we already saved one further unique separator of the form $\$_i$ in the construction for the rule-size by using ## instead, simply exploiting the fact that this substring of length two is not repeated anywhere else, which makes a rule containing it impossible in a smallest grammar. We can actually also shrink our alphabet in the construction used to prove Theorem 3 by saving separator symbols, more precisely, by only using one symbol \$ instead of $\$_1, \dots, \$_6$. Recall that $\$_1, \dots, \$_6$ only had the purpose to cut the grammar at these symbols as described in Observation 2 and hence avoid unwanted repetitions.

As a first observation, it is not hard to see that $\$_2, \$_4, \$_5$ can be removed from the $w_\mathcal{G}$, without creating unwanted repetitions. Removing $\$_2$ only creates the two unwanted (in the sense that those should not repeat by Propositions 1 and 2) substrings $\star g(7n-1)\#$ and $d_1\#f(C_v(1))\star$, which do not occur elsewhere in $w_\mathcal{G}$ (more precisely for the second substring: $y\#f(C_v(1))\star$ with $y \notin \{x_1, \dots, x_7\}$ occurs only two other times once with $y = \$_1$ and, after removal of $\$_5$, once with $y = ¢_2$). Similar arguments hold for removing $\$_4$ and $\$_5$. The remaining $\$_i$ occur in the subwords: $x_6\$_1\#x_{C_v(1)}$, $d_5\$_3x_{C_v(1)}$, $d_7\$_6\#x_{C_v(j_1)}$. Now consider replacing $\$_1, \$_3, \$_6$ each by the same symbol \$. If we make sure to list the edges in an order such that $C_v(1) \neq C_v(j_1)$, the only repeating factor of length more than one containing this new symbol \$ is \$#. As this subword of length two only occurs twice, it is not profitable for a smallest grammar to compress it with a rule. So with the little adjustments of deleting $\$_2, \$_4, \$_5$, possible picking another order to list the edges and replacing $\$_1, \$_3, \$_6$ by \$, we need five symbols less for our reduction.

Further reduction of the alphabet size requires much more effort. Our main kind of argument is that certain rules cannot exist, simply because their derivative does not occur more than once in $w_\mathcal{G}$. There are cases, where it is possible to show that certain rules *with* a repeated derivative do not occur, but the respective argument cannot be local and would rather depend on the structure of the whole grammar. On the other hand, rules that we want fixed in a smallest grammar have to be provably profitable. With these properties in mind, it is quite obvious that there is not much room to reduce the alphabet size further.

The symbols $\star, \#, ¢_1, ¢_2$ and, after applying the replacement above, \$ each have a very specific purpose. It seems very difficult to reduce the alphabet by replacing one of those characters by another or some codeword.

For the symbols $x_1, \dots, x_7, d_1, \dots, d_7$, we see that in Lemma 5, which fixes the codewords for vertices and separators built from these symbols, we require at least six repetitions of each desired codeword. Doing this without repeating unwanted subwords, means that, at least with the idea we used to repeat these codewords in the alternating fashion given by the subword u, we need at least six different symbols in each encoding. For the separators $\langle j \rangle_\diamond$, our construction requires the seven different symbols $d_1, \dots, d_7$, to have unique separators between the repetitions of the subwords $\#\langle i \rangle_v$, $\langle i \rangle_v\#$ and $\#\langle i \rangle_v\#$

in v and between the edges in the listing in w, for which we need four different kinds of separators, one for each colour of the edge-colouring C_e. For the vertex codewords $\langle i \rangle_v$, we also need seven different symbols to represent the vertex colouring C_v. So, first of all, the only way to save symbols among $x_1, \ldots, x_7, d_1, \ldots, d_7$ seems to modify the input graph in such a way that the colourings C_e and C_v require less colours. It is possible to do this with the adjustments described in the following.

Given a subcubic graph $\mathcal{G} = (V, E)$, we first build the graph $\overline{\mathcal{G}}$ from $\mathcal{G}$ by subdividing each edge twice, i.e., we replace each edge $(u, v) \in E$ by three edges $(u, u_v), (u_v, v_u)$ and (v_u, v), where u_v and v_u are two new vertices which are not adjacent to further edges. We now construct the word for SGP to represent the graph $\overline{\mathcal{G}}$. This shift to the graph $\overline{\mathcal{G}}$ can be used to decrease the number of colours we require both for C_v and C_e. First observe that the graph $\overline{\mathcal{G}}^2$ (i. e., the graph obtained from $\overline{\mathcal{G}}$ by the same operation used to obtain $\mathcal{G}^2$ from $\mathcal{G}$ in the original reduction; see page 23) has maximum degree three, as a vertex $v \in V$ is adjacent to the at most three vertices in $\{u_v \colon (u, v) \in E\}$, and a vertex v_u, added by the subdivision process for an edge (u, v), is adjacent to u and possible the at most two vertices in $\{v_x \colon (v, x) \in E, x \neq u\}$. The vertex colouring C_v hence only needs four different colours to properly colour $\overline{\mathcal{G}}^2$.

Next, we choose a specific listing of the edges of $\overline{\mathcal{G}}$ such that the three edges of $\overline{\mathcal{G}}$ corresponding to an edge (u, v) of $\mathcal{G}$ are consecutively listed as $(u_v, u), (v_u, u_v), (v, v_u)$ (and the relative order of such triples is arbitrary). In this way, the multi-graph $\overline{\mathcal{G}}'$ (i. e., the graph obtained from $\overline{\mathcal{G}}$ by the same operation used to obtain $\mathcal{G}'$ from $\mathcal{G}$ in the original reduction; see page 23) contains the edges $\{(u, v_u), (u_v, v) \colon (u, v) \in E\}$ for vertices from V and, in addition, we have at most one edge of the form $(u_v, u'_{v'})$ for each new vertex added by the subdivision. This means that in $\overline{\mathcal{G}}'$, a vertex $v \in V$ is only adjacent to the at most three vertices in $\{u_v \colon (u, v) \in E\}$, and a vertex u_v added by the subdivision process for the edge (u, v) is adjacent to one edge connected to v and to at most one other edge connected to a vertex added by the subdivision process different from u_v. Consequently, $\overline{\mathcal{G}}'$ is a simple graph and of maximum degree three. Further, observe that the vertices of degree three in $\overline{\mathcal{G}}'$ (which are a subset of the vertices in V) form an independent set in $\overline{\mathcal{G}}'$. By a theorem of Fournier [22], an edge-colouring for a graph with these properties, only requires three colours and can be computed in polynomial time with Vizings algorithm [60]. With the same arguments used to prove Theorem 3, it follows that a smallest grammar encodes a minimum vertex cover for $\overline{\mathcal{G}}$. It remains to observe that the size of a minimum vertex cover for the original input graph $\mathcal{G}$ can be derived from a minimum vertex cover for $\overline{\mathcal{G}}$. If $\mathcal{G}$ has a vertex cover of size k, then this can be extended to a vertex cover of size $k + |E|$ for $\overline{\mathcal{G}}$ by adding exactly one of u_v and v_u for each edge (u, v) of $\mathcal{G}$. On the other hand, it can be easily seen that, without loss of generality, a minimum vertex cover for $\overline{\mathcal{G}}$ contains exactly one of u_v and v_u for each edge (u, v) of $\mathcal{G}$, and, moreover, the remaining k vertices in the vertex cover for $\overline{\mathcal{G}}$ must be a vertex cover for the graph $\mathcal{G}$.

Overall, the adjustments described so far lead to a hardness reduction which only uses an alphabet with 17 symbols, as we now only require a 6-ary encoding for vertices and separators. Observe that, although the colouring C_v only requires four colours now, we cannot reduce the alphabet for the vertices to be less than six, as we need six different symbols for the repetitions in u.

Corollary 1 SGP *is* NP-*complete, even for alphabets of size* 17.

The reduction sketched above can still be seen as an L-reduction from the optimisation version of vertex cover to $\text{SGP}_{\mathsf{opt}}$. Too see this, observe that the adjustments made to reduce the alphabet only cause an addition of $\mathcal{O}(m)$ to the size of a smallest grammar for the word constructed for the input graph $\mathcal{G}$. As $\mathcal{O}(m) \subseteq \mathcal{O}(m^*_{VC}(\mathcal{G}))$ (recall that $\mathcal{G}$ is cubic), the size of the smallest grammar can be linearly bounded by $m^*_{VC}(\mathcal{G})$ in a similar way as shown in the proof of Theorem 4.

Corollary 2 $\text{SGP}_{\mathsf{opt}}$ *is* APX-*hard, even for alphabets of size* 17.

The only way to further reduce the alphabet would be to not just use the repetitions in u to prove Lemma 5 but the repetitions in the whole word. This however is very difficult, as including the rules we want to fix can no longer easily be shown to shorten the axiom. If there is no nonterminal V_i which derives $\langle i \rangle_v$ for some index i, the larger substring $\#\langle i \rangle_v ¢_1$ in v, for example, might still only require three symbols in the axiom by compressing parts of $\langle i \rangle_v$ with $\#$ or $¢_1$. Similarly for all occurrences of the substring $\langle i \rangle_v$ in v or w. This problem is actually the reason, why we need the nonterminals V_i and D_i fixed for Lemma 6, to make our desired rules to derive $\langle i \rangle_v\#$ and $\#\langle i \rangle_v$ in the cheapest possible way to enable the argument that other unwanted rules in N_{ax} cannot be more profitable. Consequently, an alphabet of size 17 seems to be necessary to cleanly prove Theorem 3 with our construction.

Similar ideas and limits for alphabet reduction hold for the rule-size measure. A reduction that only uses \$ instead of $\$_1, \dots, \$_8$ works analogously. The symbols $\$_i$ with $i \in \{2, 4, 5, 6, 7\}$ can be deleted without creating repetitions of unwanted subwords. Replacing the remaining $\$_i$, $i \in \{1, 4, 8\}$ by \$ and again reordering the edges in the listing given in w'' such that $x_{C_v(1)} \neq x_{C_v(j_1)}$ makes sure that the only repeating factor of length more than one containing the new symbol \$ is $\$\#$. This factor occurs exactly twice and is hence not compressed by a rule in a smallest grammar (observe that with the rule-size as measure, such a rule is not just unprofitable but even makes the grammar larger). As we here require eight repetitions to show the equivalent of Lemma 5 for the rule-size, saving symbols among $x_1, \dots, x_8, d_1, \dots, d_8$ is not possible. Consequently, Theorems 6 and 7 can be improved to require only an alphabet of size 22 but a reduction with a smaller alphabet will be very difficult with our construction.

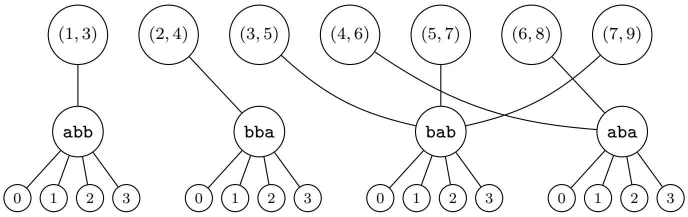

Fig. 2 The third layer of $\Phi_1(\texttt{abbababab})$ (edges from E_1 are omitted). The uppermost vertices $(1,3),(2,4),\ldots$ are from V_1, the ones in the middle labelled by $\texttt{abb},\texttt{bba},\ldots$ are the ones from V_2 and, finally, the lower vertices are from V_3 (for the sake of convenience, these are labelled by i instead of (u,i)).

4 Smallest Grammars with a Bounded Number of Nonterminals

A natural follow-up question to the hardness for fixed alphabets is whether polynomial-time solvability is possible if instead the cardinality of the nonterminal alphabet N (or, equivalently, the number of rules) is bounded. In this section, we answer this question in the affirmative by representing words $w \in \Sigma^*$ as graphs $\Phi_{\mathrm{m}}(w)$ and $\Phi_1(w)$, such that smallest independent dominating sets of these graphs correspond to smallest grammars and smallest 1-level grammars, respectively, for w.

It will be more convenient to first take care of the simpler 1-level case and to treat then the multi-level case as an extension of it, i. e., we first define $\Phi_1(w)$ and then derive $\Phi_{\mathrm{m}}(w)$ from $\Phi_1(w)$. Recall that, as defined in Section 2, $\mathsf{F}_{\geq 2}(w)$ is the set of factors of w with size at least 2. Let $\Phi_1(w) = (V, E)$ be defined by $V = V_1 \cup V_2 \cup V_3$ and $E = E_1 \cup E_2 \cup E_3$, where:

$$\begin{aligned}
V_1 &= \{(i,j) \colon 1 \leq i \leq j \leq |w|\}\,, & E_1 &= \{\{(i_1,j_1),(i_2,j_2)\} \colon i_1 \leq i_2 \leq j_1\}\,,\\
V_2 &= \mathsf{F}_{\geq 2}(w)\,, & E_2 &= \{\{w[i..j],(i,j)\} \colon 1 \leq i < j \leq |w|\}\,,\\
V_3 &= \{(u,i) \colon u \in V_2, 0 \leq i \leq |u|\}\,, & E_3 &= \{\{u,(u,i)\} \colon u \in V_2, 0 \leq i \leq |u|\}\,.
\end{aligned}$$

Intuitively speaking, the vertices of V_1 represent every factor by its start and end position, whereas V_2 contains exactly one vertex per factor of length at least 2. Every $u \in V_2$ is connected to (i,j), if and only if $w[i..j] = u$. Vertices (i,j), (i',j') are connected if they refer to overlapping factors. For every $u \in V_2$, there are $|u| + 1$ special vertices in V_3 that are only adjacent with u. Consequently, we can view $\Phi_1(w)$ as consisting of $|w|$ layers, where the i^{th} layer contains the vertices $(j, j + (i-1)) \in V_1$, $1 \leq j \leq |w| - (i-1)$, the vertices $\{u \in V_2 \colon |u| = i\}$ and the vertices $\{(u,j) \in V_3 \colon |u| = i, 0 \leq j \leq |u|\}$ (see Figure 2 for an illustration).

Next, we show that 1-level grammars for w correspond to independent dominating sets for $\Phi_1(w)$. Intuitively speaking, the vertices in an independent dominating set from V_1 induce a factorisation of w, which, in turn, induces the axiom of a 1-level grammar in the natural way (i. e., every factor of size at least 2 is represented by a rule). If $(i,j) \in V_1$ is in the independent dominating set, then $w[i..j] \in V_2$ is not; thus, due to the domination-property, all $(w[i..j], \ell) \in V_3$, $0 \leq \ell \leq j - i + 1$, are in the independent dominating set, which represents the size of the rule.

Lemma 9 *Let $w \in \Sigma^*$, $k \geq 1$. There exists an independent dominating set D of cardinality at most k for $\Phi_1(w)$ if and only if there exists a 1-level grammar G for w with $|G| \leq k - |\mathsf{F}_{\geq 2}(w)|$.*

Proof We start with the *if* direction. If $G = (N, \Sigma, R, \mathsf{ax})$ is a 1-level grammar for w with size $k - |\mathsf{F}_{\geq 2}(w)|$, then we can construct an independent dominating set D for $\Phi_1(w)$ of size k as follows. Let $\mathsf{ax} = A_1 A_2 \ldots A_n$, $A_i \in N \cup \Sigma$, $1 \leq i \leq n$, and let $F = \{\mathfrak{D}(A) \colon A \in N\}$. For every i, $1 \leq i \leq n$, we add $(|\mathfrak{D}(A_1 \ldots A_{i-1})| + 1, |\mathfrak{D}(A_1 \ldots A_i)|) \in V_1$ to D and, if $A_i \in N$, then we also add all $\{(\mathfrak{D}(A_i), j) \colon 0 \leq j \leq |\mathfrak{D}(A_i)|\}$ to D. Furthermore, we add all $V_2 \setminus F$ to D. It can be easily verified that D is an independent dominating set. Moreover, $|D| = |\mathsf{ax}| + \sum_{v \in F}(|v| + 1) + |V_2 \setminus F| = |\mathsf{ax}| + \sum_{v \in F}|v| + |V_2| = |\mathsf{ax}| + \sum_{A \in N}|\mathfrak{D}(A)| + |V_2| = |G| + |\mathsf{F}_{\geq 2}(w)|$. Since $|G| = k - |\mathsf{F}_{\geq 2}(w)|$, we conclude that $|D| = k$.

Next, we prove the *only if* direction. Let D be an independent dominating set for $\Phi_1(w)$. We first note that, for every $u \in V_2 \setminus D$, $\{(u, j) \colon 0 \leq j \leq |u|\} \subseteq D$, which implies that

$$\begin{aligned}
|D| &= |D \cap V_1| + |D \cap V_2| + |D \cap V_3| \\
&\geq |D \cap V_1| + |D \cap V_2| + \sum_{u \in (V_2 \setminus D)} \{(u, j) \colon 0 \leq j \leq |u|\} \\
&= |D \cap V_1| + |D \cap V_2| + \sum_{u \in (V_2 \setminus D)} (|u| + 1) \\
&= |D \cap V_1| + |V_2| + \sum_{u \in (V_2 \setminus D)} |u| \,.
\end{aligned}$$

For every i, $1 \leq i \leq |w|$, we say that i is covered by $(j, j') \in V_1$ if $(j, j') \in D$ and $j \leq i \leq j'$ (recall that any vertex (i, i) can only be dominated by some vertex (j, j') with $j \leq i \leq j'$, since vertex (i, i) has no neighbours in V_2). If some i, $1 \leq i \leq |w|$, is not covered by any $(j, j') \in V_1$, then (i, i) is not dominated by D and if i is covered by two different elements from V_1, then there is an edge (from E_1) between them, so that D is not an independent set. Thus, every i, $1 \leq i \leq |w|$, is covered by exactly one element $(j, j') \in V_1$. This directly implies that $D \cap V_1 = \{(\ell_1, r_1), (\ell_2, r_2), \ldots, (\ell_m, r_m)\}$, such that $(u_1, u_2, \ldots, u_m)$ is a factorisation of w, where $u_j = w[\ell_j..r_j]$, $1 \leq j \leq m$. Due to the edges in E_2, we know that, for every j, $1 \leq j \leq m$, with $\ell_j < r_j$, there is an edge $(u_j, (\ell_j, r_j))$; thus, $u_j \in (V_2 \setminus D)$. Next, we define $N = \{A_u \colon u \in (V_2 \setminus D)\}$ and $R = \{A_u \to u \colon u \in (V_2 \setminus D)\}$. Since now for each j, $1 \leq j \leq m$, either $u_j \in \Sigma$ or there exists a non-terminal A_{u_j} which derives u_j, we can define an axiom of length m by $\mathsf{ax} = C_{u_1} C_{u_2} \ldots C_{u_m}$ with $C_{u_j} = A_{u_j}$ for all j with $|u_j| > 1$ and $C_{u_j} = u_j$ otherwise, in order to obtain a 1-level grammar $G = (N, \Sigma, R, \mathsf{ax})$

with $\mathfrak{D}(G) = w$. Finally, we note that

$$\begin{aligned}|G| &= |\mathsf{ax}| + \sum_{u \in (V_2 \setminus D)} (|u|)\\ &= |D \cap V_1| + |V_2| + \left(\sum_{u \in (V_2 \setminus D)} |u|\right) - |V_2|\\ &\leq |D| - |\mathsf{F}_{\geq 2}(w)|\,.\end{aligned}$$

□

Since in the multi-level case the derivatives of the nonterminals that appear in the axiom are again compressed by a grammar, a first idea that comes to mind is to somehow represent the vertices $u \in V_2$ again by graph structures of the type $\Phi_1(u)$ and iterating this step. However, naively carrying out this idea would lead to redundancies (copies of the subgraph representing a factor u would appear inside subgraphs representing different superstrings $w_1 u w_2$ and $w'_1 u w'_2$) that even seem to cause an exponential size increase of the graph structure. Fortunately, it turns out that these redundancies can be avoided and a surprisingly simple modification of $\Phi_1(w)$ is sufficient.

For a word $w \in \Sigma^*$, let $\Phi_{\mathrm{m}}(w) = (V, E)$ be defined as follows. Let $V = V_1 \cup V_2 \cup V_3 \cup V_4$, where V_1 and V_2 are defined as for $\Phi_1(w)$, whereas

$$V_3 = \{(u, 0) \colon u \in V_2\} \text{ and}$$

$$V_4 = \bigcup_{u \in V_2} V_{4,u} \text{ with } V_{4,u} = \{(u, i, j) \colon 1 \leq i \leq j \leq |u|, u[i..j] \neq u\} \text{ for } u \in V_2\,.$$

Moreover, $E = E_1 \cup E_2 \cup E_3 \cup E_4 \cup E_5$, where E_1 and E_2 are defined as for $\Phi_1(w)$, while

$$\begin{aligned}E_3 =& \{\{u, (u, 0)\} \colon u \in V_2\} \cup \{\{u, (u, i, j)\} \colon u \in V_2, (u, i, j) \in V_{4,u}\}\,,\\ E_4 =& \bigcup_{u \in V_2} E_{4,u}, \text{ with } E_{4,u} = \{\{(u, i_1, j_1), (u, i_2, j_2)\} \subseteq V_{4,u} \colon i_1 \leq i_2 \leq j_1\},\\ & \text{for every } u \in V_2, \text{ and}\\ E_5 =& \{\{u, (v, i, j)\} \colon u, v \in V_2, v[i..j] = u, u \neq v\}\,.\end{aligned}$$

Intuitively speaking, $\Phi_{\mathrm{m}}(w)$ differs from $\Phi_1(w)$ in the following way. We add to every vertex $u \in V_2$ a subgraph $(V_{4,u}, E_{4,u})$, which is completely connected to u and which represents u in the same way as the subgraph (V_1, E_1) of $\Phi_1(w)$ represents w, i. e., factors $u[i..j]$ are represented by (u, i, j) and edges represent overlappings. Moreover, if a $u \in V_2$ is a factor of some $v \in V_2$, then there is an edge from u to all the vertices $(v, i, j) \in V_{4,v}$ that satisfy $v[i..j] = u$ (by these "crosslinks", we get rid of the redundancies mentioned above). Finally, every $u \in V_2$ is also connected with an otherwise isolated vertex $(u, 0) \in V_3$. See Figure 3 for a partial illustration of a $\Phi_{\mathrm{m}}(w)$.

Similar as for the 1-level case, we can show that (multi-level) grammars for w correspond to independent dominating sets for $\Phi_{\mathrm{m}}(w)$:

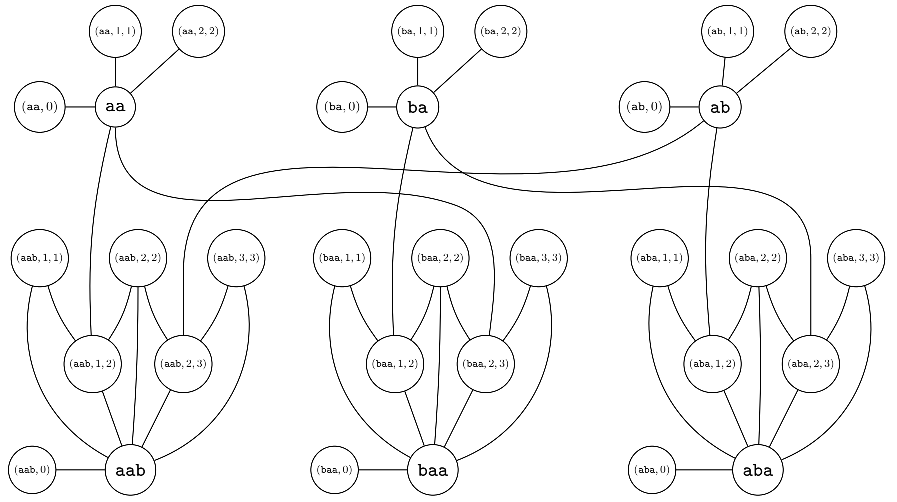

Fig. 3 Second and third layer of $\Phi_{\mathrm{m}}(\mathtt{abaabaa})$ (vertices from V_1 and edges from $E_1 \cup E_2$ omitted). For example, vertex $(\mathtt{aba}) \in V_2$ is connected to all the vertices $V_{4,\mathtt{aba}} = \{(\mathtt{aba}, i, j) \colon 1 \leq i \leq j \leq 3, j{-}i \leq 1\}$, and with $(\mathtt{aba}, 0) \in V_3$. Moreover, since $(\mathtt{aba})[1..2] = \mathtt{ab}$, there is an edge between $(\mathtt{aba}, 1, 2)$ and $(\mathtt{ab}) \in V_2$, and since $(\mathtt{aba})[2..3] = \mathtt{ba}$, there is an edge between $(\mathtt{aba}, 2, 3)$ and $(\mathtt{ba}) \in V_2$.

Lemma 10 *Let $w \in \Sigma^*$, $k \geq 1$. There is an independent dominating set D of cardinality k for $\Phi_{\mathrm{m}}(w)$ if and only if there is a grammar G for w with $|G| = k - |\mathsf{F}_{\geq 2}(w)|$.*

Proof Let D be an independent dominating set of cardinality k for $\Phi_{\mathrm{m}}(w)$. In the same way as in the proof of Lemma 9, it can be concluded that the set $V_1 \cap D = \{(\ell_1, r_1), (\ell_2, r_2), \ldots, (\ell_{m_w}, r_{m_w})\}$ corresponds to a factorisation $(w_1, w_2, \ldots, w_{m_w})$ of w, where $w_j = w[\ell_j..r_j]$, $1 \leq j \leq m_w$, and satisfies $\{w_1, w_2, \ldots, w_{m_w}\} \cap D = \emptyset$.

Next, for an arbitrary $u \in V_2$, we consider the subgraph with the vertices $N[u] \setminus V_1 = V_{4,u} \cup \{(v, i, j) \colon v[i..j] = u, u \neq v\} \cup \{u, (u, 0)\}$. If $u \in D$, then $N(u) \cap D = \emptyset$. On the other hand, if $u \notin D$, then $(u, 0) \in D$ and, analogously as for V_1, we can conclude that

$$V_{4,u} \cap D = \{(u, \ell_{u,1}, r_{u,1}), (u, \ell_{u,2}, r_{u,2}), \ldots, (u, \ell_{u,m_u}, r_{u,m_u})\},$$

such that $(u_1, u_2, \ldots, u_{m_u})$ is a factorisation of u (note that, in the same way as for V_1, if a position i of u is not covered in the sense that $(u, j, j') \in D$ with $j \leq i \leq j'$, then vertex (u, i, i) would neither be in D nor adjacent to a vertex in D), where $u_j = u[\ell_{u,j}..r_{u,j}]$, $1 \leq j \leq m_u$. Furthermore, for every j, $1 \leq j \leq m_u$, with $|u_j| \geq 2$, $\{u_j, (u, \ell_{u,j}, r_{u,j})\} \in E$; thus, $u_j \notin D$. Consequently, by induction, D induces a factorisation $(u_1, u_2, \ldots, u_{m_u})$ for every $u \in (V_2 \setminus D) \cup \{w\}$, such that, for every i, $1 \leq i \leq m_u$, $|u_j| \geq 2$ implies $u_j \in V_2 \setminus D$, which means that there is also a factorisation for u_j.

For every $u \in V_2 \setminus D$, we can now define a nonterminal A_u and a rule $A_u \rightarrow B_1 B_2 \dots B_{m_u}$, where, for every j, $1 \leq j \leq m_u$, $B_j = A_{u_j}$ if $|u_j| \geq 2$ and $B_j = u_j$ if $|u_j| = 1$. Obviously, these rules together with the axiom $\mathsf{ax} = C_1 C_2 \dots C_{m_w}$, where, for every j, $1 \leq j \leq m_w$, $C_j = A_{w_j}$ if $|w_j| \geq 2$ and $C_j = w_j$ if $|w_j| = 1$, defines a grammar G for w.

We note that $|\mathsf{ax}| = |V_1 \cap D|$ and, for every rule $A_u \rightarrow \alpha_u$, $|\alpha_u| = |V_{4,u} \cap D|$. Since

$$\begin{aligned}|D| &= |D \cap V_1| + |(D \cap (\bigcup_{u \in V_2} V_{4,u}))| + |D \cap (V_2 \cup V_3)|\,,\\ |V_2| &= |D \cap (V_2 \cup V_3)| \text{ and}\\ |G| &= |D \cap V_1| + |(D \cap (\bigcup_{u \in V_2} V_{4,u}))|\,,\end{aligned}$$

we conclude that $|G| = |D| - |V_2| = k - |\mathsf{F}_{\geq 2}(w)|$.

For a grammar G for w, we can select vertices from $\Phi_{\mathrm{m}}(w)$ according to the factorisations induced by the rules of G, which results in an independent dominating set D for $\Phi_{\mathrm{m}}(w)$ with $|D| = |G| + |V_2|$. □

For the algorithmic application of these graph encodings, it is important to note that the proofs of Lemmas 9 and 10 are constructive, i. e., they also show how an independent dominating set D of $\Phi_{\mathrm{m}}(w)$ or $\Phi_1(w)$ can be transformed into a grammar for w (a 1-level grammar for w, respectively) of size $|D| - |\mathsf{F}_{\geq 2}(w)|$, which, in the following, we will denote by $\mathsf{G}(D)$.

Thus, the smallest grammar problem can be solved by constructing $\Phi_{\mathrm{m}}(w)$ or $\Phi_1(w)$, then computing a smallest independent dominating set D for $\Phi_{\mathrm{m}}(w)$ (or $\Phi_1(w)$, respectively) and finally constructing $\mathsf{G}(D)$. Unfortunately, this does not lead to a polynomial-time algorithm, since computing a minimal independent dominating set is an NP-complete problem, even for quite restricted graph classes [47, Theorem 13].

In the following, we shall analyse the graph structures $\Phi_{\mathrm{m}}(w)$ and $\Phi_1(w)$ more thoroughly and we begin with their respective sizes:

Proposition 3 *Let $w \in \Sigma^*$. Then $\Phi_1(w)$ has $\mathcal{O}(|w|^3)$ vertices and $\mathcal{O}(|w|^4)$ edges; $\Phi_{\mathrm{m}}(w)$ has $\mathcal{O}(|w|^4)$ vertices and $\mathcal{O}(|w|^6)$ edges.*

Proof We first consider $\Phi_{\mathrm{m}}(w)$. The subgraph (V_1, E_1) has $\mathcal{O}(|w|^2)$ vertices and $\mathcal{O}(|w|^4)$ edges. Similarly, every induced subgraph on the set of vertices $V_{4,u} \cup \{u, (u, 0)\}$, $u \in V_2$ has $\mathcal{O}(|w|^2)$ vertices, $\mathcal{O}(|w|^4)$ edges and there are $\mathcal{O}(|w|^2)$ such subgraphs. In addition to this, there are $\mathcal{O}(|w|)$ edges connecting any $u \in V_2$ with vertices from V_1 and $\mathcal{O}(|w|^2)$ edges connecting any $u \in V_2$ with vertices from V_4. Finally, there are $\mathcal{O}(|w|^2)$ vertices in V_3 with one incident edge each. Consequently, $\Phi_{\mathrm{m}}(w)$ has $\mathcal{O}(|w|^4)$ vertices and $\mathcal{O}(|w|^6)$ edges.

For $\Phi_1(w)$, the situation is easier. The subgraph (V_1, E_1) has $\mathcal{O}(|w|^2)$ vertices and $\mathcal{O}(|w|^4)$ edges. There are $\mathcal{O}(|w|^2)$ vertices in V_2 and each $u \in V_2$ has $\mathcal{O}(|w|)$ edges. Finally, there are $\mathcal{O}(|w|^2)$ vertices in V_3 with one edge each. Consequently, $\Phi_1(w)$ has $\mathcal{O}(|w|^3)$ vertices and $\mathcal{O}(|w|^4)$ edges. □

Next, we investigate the interval-structure of $\Phi_{\mathrm{m}}(w)$ and $\Phi_1(w)$.

Proposition 4 *$\Phi_{\mathrm{m}}(w)$ and $\Phi_1(w)$ are 2-interval graphs.*

Proof In the following 2-interval representations, we denote by $I_1(v)$ the first and by $I_2(v)$ the second interval that represents a vertex v.

We first consider the graph $\Phi_1(w)$. For every $(i, j) \in V_1$, we set $I_1((i, j)) = [i, j]$; this already yields the subgraph (V_1, E_1). In addition, let $I_1(u)$, $u \in V_2$, be a sequence of pairwise disjoint intervals that are also disjoint with the intervals $I_1((i, j))$, $(i, j) \in V_1$. For every $(u, j) \in V_3$, let $I_1((u, j))$ be an interval that lies within $I_1(u)$ and is disjoint from every other interval. Now, it only remains to represent the edges from E_2, for which we simply let $I_2((i, j))$, $(i, j) \in V_1$, be an interval that lies within $I_1(w[i..j])$ and is disjoint from every other interval. Note that only the vertices from V_1 are represented by two intervals each.

For $\Phi_{\mathrm{m}}(w)$, we represent $V_1 \cup V_2$ and the edges $E_1 \cup E_2$ by intervals in the same way as for the graph $\Phi_1(w)$. Then, for every $u \in V_2$ and $(u, i, j) \in V_{4,u}$, we set $I_1((u, i, j)) = [i+k_u, j+k_u]$, where k_u is chosen such that all these intervals lie inside $I_1(u)$ without intersecting an interval $I_2((i, j))$ for some $(i, j) \in V_1$. In particular, this takes care of all the edges $E_{4,u}$ (due to the intersections between these intervals) and the edges between u and the vertices $V_{4,u}$ (due to the fact that these intervals lie inside $I_1(u)$). In order to take care of the edges from E_5, for every u and for every $(v, i, j) \in V_{4,v}$ with $v[i..j] = u$, we place a new interval $I_2((v, i, j))$ inside of $I_1(u)$ such that it does not intersect with any other interval inside of $I_1(u)$. This creates all the edges from E_5. Now it only remains to take care of vertices $(u, 0)$, $u \in V_2$, and their edges, which can be done by placing a new interval $I_1((u, 0))$ inside $I_1(u)$ such that it does not intersect with any other interval. □

Unfortunately, the independent dominating set problem for 2-interval graphs is still NP-complete (in [47], the hardness of the independent dominating set problem for subcubic graphs is shown and from [29], it follows that subcubic graphs are 2-interval graphs). Nevertheless, solving the smallest grammar problem by computing small independent dominating sets for $\Phi_{\mathrm{m}}(w)$ or $\Phi_1(w)$, as sketched before Proposition 3, might still be worthwhile, since computing small independent dominating sets is a well-researched problem, for which the literature provides fast and sophisticated algorithms (see [31,11]). In particular, the 2-interval structure suggests that we are dealing with simpler instances of the independent dominating set problem.

Our algorithmic application of the graph encodings, which leads to the polynomial-time solvability of the smallest grammar problem with a bounded number of nonterminals, can be sketched as follows. If we have fixed the set of factors $F \subseteq \mathsf{F}_{\geq 2}(w)$ to occur as derivatives of nonterminals in the grammar, i.e., $\{\mathfrak{D}(A) \colon A \in N\} = F$, then, for the corresponding independent dominating set D of $\Phi_{\mathrm{m}}(w)$ or $\Phi_1(w)$, we must have $(\mathsf{F}_{\geq 2}(w) \setminus F) \subseteq D$ and $F \cap D = \emptyset$. Thus, in order to find an independent dominating set that is minimal among all those that correspond to a grammar with $\{\mathfrak{D}(A) \colon A \in N\} = F$, it is sufficient to first select the vertices $(\mathsf{F}_{\geq 2}(w) \setminus F)$, deleting the neighbourhood of this vertex set and computing a smallest independent dominating set for what remains, which is the graph $\mathcal{H} = \Phi(w) \setminus (N[\mathsf{F}_{\geq 2}(w) \setminus F] \cup F)$.[10] However, $\mathcal{H}$ is an interval graph, so a smallest independent dominating set can be computed in linear time.

In order to carry out this approach, we first formally prove that $\mathcal{H}$ is an interval graph:

Proposition 5 *Let $w \in \Sigma^+$, $F \subseteq \mathsf{F}_{\geq 2}(w)$ and $\Phi(w) \in \{\Phi_{\mathrm{m}}(w), \Phi_1(w)\}$. Then $\mathcal{H} = \Phi(w) \setminus (N[\mathsf{F}_{\geq 2}(w) \setminus F] \cup F)$ is an interval graph.*

Proof We only prove the case $\Phi(w) = \Phi_{\mathrm{m}}(w)$, since the case $\Phi(w) = \Phi_1(w)$ can be handled analogously. First, we consider the 2-interval representation of $\Phi_{\mathrm{m}}(w)$ described in the proof of Proposition 4. We can now obtain a 1-interval representation of $\mathcal{H}$ from the 2-interval representation of $\Phi_{\mathrm{m}}(w)$ as follows. Since $\mathcal{H}$ does not contain any vertex from V_2, we first remove the corresponding intervals for vertices from V_2. The only vertices represented by more than one interval are the ones from V_1 and V_4. However, the second intervals of these only intersect intervals which represent vertices from V_2 in the 2-interval representation of $\Phi_{\mathrm{m}}(w)$, which means that they are now all isolated and can therefore be removed. Consequently, every vertex of $\mathcal{H}$ can be represented by one interval. □

Next, we show that independent dominating sets for $\mathcal{H}$ can be easily extended to independent dominating sets for $\Phi_{\mathrm{m}}(w)$ (or $\Phi_1(w)$).

Proposition 6 *Let $w \in \Sigma^+$, $F \subseteq \mathsf{F}_{\geq 2}(w)$, $\Phi(w) \in \{\Phi_{\mathrm{m}}(w), \Phi_1(w)\}$ and let $D_{\mathcal{H}}$ be an independent dominating set for $\mathcal{H} = \Phi(w) \setminus (N[\mathsf{F}_{\geq 2}(w) \setminus F] \cup F)$. Then $D_{\mathcal{H}} \cup (\mathsf{F}_{\geq 2}(w) \backslash F)$ is an independent dominating set for $\Phi(w)$.*

Proof We start with the multi-level case. Since $D_{\mathcal{H}}$ is an independent dominating set for $\mathcal{H}$, it is also an independent set for $\Phi_{\mathrm{m}}(w)$. The only vertices of $\Phi_{\mathrm{m}}(w)$ that are not necessarily dominated by $D_{\mathcal{H}}$ are from $N[\mathsf{F}_{\geq 2}(w) \setminus F]$ or F. Since $\mathsf{F}_{\geq 2}(w) \setminus F \subseteq D_{\mathcal{H}} \cup (\mathsf{F}_{\geq 2}(w) \backslash F)$, the vertices from $N[\mathsf{F}_{\geq 2}(w) \setminus F]$ are dominated by $D_{\mathcal{H}} \cup (\mathsf{F}_{\geq 2}(w) \backslash F)$. Regarding the vertices from F, we note that since $F \cap D_{\mathcal{H}} = \emptyset$, the vertices $\{(u, 0) \colon u \in F\}$ occur in $\mathcal{H}$ as isolated vertices and, thus, they must be included in $D_{\mathcal{H}}$, which means that the vertices F are dominated in $\Phi_{\mathrm{m}}(w)$ by $D_{\mathcal{H}} \cup (\mathsf{F}_{\geq 2}(w) \backslash F)$ as well. Now it only remains to observe that, by definition of $\Phi_{\mathrm{m}}(w)$, the vertices $(\mathsf{F}_{\geq 2}(w) \backslash F)$ are clearly independent and, since their neighbourhood is completely excluded from $\mathcal{H}$ and therefore also from $D_{\mathcal{H}}$, they are also independent from the vertices in $D_{\mathcal{H}}$. Consequently, $D_{\mathcal{H}} \cup (\mathsf{F}_{\geq 2}(w) \backslash F)$ is an independent dominating set for $\Phi_{\mathrm{m}}(w)$.

The argument for the 1-level case is very similar with the only difference that $\{(u, i) \colon u \in F, 0 \leq i \leq |u|\}$ are the vertices from $D_{\mathcal{H}} \cup (\mathsf{F}_{\geq 2}(w) \backslash F)$ that dominate the vertices F. □

[10] See page 8 for the definition of the closed neighbourhood.

For the sake of convenience, for any $F \subseteq \mathsf{F}_{\geq 2}(w)$, we denote a grammar $G = (N, \Sigma, R, \mathsf{ax})$ for w with $\{\mathfrak{D}(A) \colon A \in N\} = F$ by the term *F-grammar*, a smallest F-grammar for w is one that is minimal among all F-grammars for w.

Lemma 11 *Let $w \in \Sigma^+$ and $F \subseteq \mathsf{F}_{\geq 2}(w)$. A smallest F-grammar for w can be computed in time $\mathcal{O}(|w|^6)$ and a smallest 1-level F-grammar for w can be computed in time $\mathcal{O}(|w|^4)$.*

Proof Again, we only prove the multi-level case, since the 1-level case can be dealt with analogously. We compute a smallest F-grammar for w as follows. First, we construct $\Phi_{\mathrm{m}}(w)$ and then $\mathcal{H} = \Phi_{\mathrm{m}}(w) \setminus (N[\mathsf{F}_{\geq 2}(w) \setminus F] \cup F)$, which can be done in time $\mathcal{O}(|\Phi_{\mathrm{m}}(w)|) = |w|^6$ (see Proposition 3). Obviously, we could also construct $\mathcal{H}$ directly, which would not change the overall running-time. Next, we compute a minimal independent dominating set $D_{\mathcal{H}}$ for $\mathcal{H}$, which, since $\mathcal{H}$ is an interval graph (see Proposition 5), can be done in time $\mathcal{O}(|\mathcal{H}|) = \mathcal{O}(|w|^6)$ (see Section 2.1). Finally, we construct $G = \mathsf{G}(D_{\mathcal{H}} \cup (\mathsf{F}_{\geq 2}(w) \setminus F))$ (note that, by Proposition 6, $D_{\mathcal{H}} \cup (\mathsf{F}_{\geq 2}(w) \setminus F)$ is an independent dominating set for $\Phi_{\mathrm{m}}(w)$; thus, G is well-defined), which can be done in time $\mathcal{O}(|w|^6)$ as well.

It remains to prove that G is a smallest F-grammar. To this end, we assume that there exists an F-grammar G' for w and $|G'| < |G|$. Consequently, by Lemma 10, there is an independent dominating set D' for $\Phi_{\mathrm{m}}(w)$ with $|G'| = |D'| - |\mathsf{F}_{\geq 2}(w)|$. Since both G and G' are F-grammars, $\mathsf{F}_{\geq 2}(w) \setminus D = \mathsf{F}_{\geq 2}(w) \setminus D' = F$. This implies that $D'_{\mathcal{H}} = D' \setminus (\mathsf{F}_{\geq 2}(w) \setminus F)$ is an independent dominating set for $\mathcal{H}$. Since by Lemma 10, $|G| = |D| - |\mathsf{F}_{\geq 2}(w)|$ and, by assumption, $|D'| < |D|$, it follows that $|D'_{\mathcal{H}}| < |D_{\mathcal{H}}|$, which is a contradiction to the minimality of $D_{\mathcal{H}}$. Consequently, G is a smallest F-grammar for w. □

If instead of a set F of factors, we are only given an upper bound k on $|N|$, then we can compute a smallest grammar by enumerating all $F \subseteq \mathsf{F}_{\geq 2}(w)$ with $|F| \leq k$ and computing a smallest F-grammar. This shows that smallest grammars can be computed in polynomial time if the number of nonterminals is bounded.

Theorem 10 *Let $w \in \Sigma^*$ and $k \in \mathbb{N}$. A grammar (1-level grammar, resp.) for w with at most k rules that is smallest among all grammars (1-level grammars, resp.) for w with at most k rules can be computed in time $\mathcal{O}(|w|^{2k+6})$ ($\mathcal{O}(|w|^{2k+4})$, resp.).*

Proof Obviously, a grammar G for w with k rules and

$$|G| = \min\{|G'| \colon G' \text{ is smallest } F\text{-grammar, with } F \subseteq \mathsf{F}_{\geq 2}(w), |F| \leq k\}$$

is smallest among all grammars for w with at most k rules. In order to compute such a grammar, it is sufficient to compute, for every set $F \subseteq \mathsf{F}_{\geq 2}(w)$ with $|F| \leq k$, a smallest F-grammar, which requires time $\mathcal{O}(|w|^{2k} \cdot |w|^6) = \mathcal{O}(|w|^{2k+6})$.

Analogously, we can compute a 1-level grammar for w with at most k rules that is smallest among all 1-level grammars for w with at most k rules in time $\mathcal{O}(|w|^{2k+4})$. □

This result raises some related questions, which shall be discussed next.

4.1 Related Questions

In the literature on grammar-based compression, the size of a smallest grammar has been interpreted in terms of a computable upper bound of the Kolomogorov complexity and, thus, as some measure for entropy or information content of strings (see Section 1). Similarly, we could treat the minimal number of nonterminals (i. e., number of rules) that are needed for a smallest grammar as a general parameter of strings, which we call the *rule-number*. The main motivation for doing this is pointed out by Theorem 10, which shows that a smallest grammar for w can be computed in time that is exponential only in the rule-number of w (or, in parameterised complexity terms, the smallest grammar problem parameterised by $|N|$ is in XP). However, in order to apply the algorithm of Theorem 10 in this regard, we need to know the rule-number, which naturally leads to the question whether the rule-number of a given string can efficiently be computed. However, the hardness reductions for the rule-size variants of the smallest grammar problem (see Section 3.3) has already provided a negative answer to this question (see Theorems 8 and 9).

The XP-membership of the smallest grammar problem, provided by Theorem 10, shows that the parameter $|N|$ has a stronger impact on the complexity than $|\Sigma|$ and, furthermore, it gives reason to hope that bounding $|N|$ might also lead to practically relevant algorithms. In this regard, the algorithm of Theorem 10 with its running-time of the form $|w|^{\mathcal{O}(|N|)}$ is a bit dissapointing, since it cannot be considered practical for larger constant bounds on $|N|$. On the other hand, an algorithm with a running-time of $f(|N|) \cdot g(|w|)$, for a polynomial g, would be a huge improvement. In other words, the question is whether the smallest grammar problem is also fixed-parameter tractable with respect to the number of nonterminals. Unfortunately, this seems unlikely, since, as stated by the next result, these parameterisations of 1-SGP and SGP are W[1]-hard. To prove this, we devise a parameterised reduction from the independent set problem parameterised by the size of the independent set, which is known to be W[1]-hard (see [16]).

Let $\mathcal{G} = (V, E)$ be a graph with $V = \{v_1, v_2, \ldots, v_n\}$, $|E| = m$, and let $k \in \mathbb{N}$. We define the alphabet $\Sigma = V \cup \{\#\} \cup \{\diamond_i \colon 1 \leq i \leq m + \sum_{i=1}^{n} n - |N(v_i)|\}$ and the following word over Σ

$$w = \prod_{\{v_i, v_j\} \in E} (\# v_i \# v_j \# \diamond) \prod_{i=1}^{n} (\# v_i \# \diamond)^{n - |N(v_i)|} .$$

As already done in Section 3, every occurrence of $\diamond$ in the word stands for a distinct symbol of $\{\diamond_i \colon 1 \leq i \leq m + \sum_{i=1}^{n} n - |N(v_i)|\})$. Note that $|w| = 6m + 4(n^2 - 2m) = 4n^2 - 2m$.

Lemma 12 *The following statements are equivalent for each $k \leq n$:*

- *$\mathcal{G}$ has an independent set I with $|I| = k$.*
- *There is a grammar G for w with at most k nonterminals and $|G| \leq 4n^2 - 2m + 3k - 2kn$.*
- *There is a 1-level grammar G for w with at most k nonterminals and $|G| \leq 4n^2 - 2m + 3k - 2kn$.*

Proof We first prove the equivalence of the first and the third statement. Let I be an independent set for $\mathcal{G}$ with $|I| = k$. We define a grammar $G = (N, \Sigma, R, \mathsf{ax})$ by $N = \{A_i : v_i \in I\}$, $R = \{A_i \to \# v_i \# : A_i \in N\}$ and $\mathsf{ax} = w'$, where w' is obtained from w, by replacing, for every $v_i \in I$, all occurrences of $\# v_i \#$ by A_i (note that since I is an independent set, no two occurrences of factors $\# v_i \#$ and $\# v_j \#$ with $v_i, v_j \in I$ overlap). Obviously, G is a 1-level grammar for w with k nonterminals. For every $v_i \in I$, $|\mathsf{ax}|_{A_i} = |N(v_i)| + (n - |N(v_i)|) = n$; thus, $\mathsf{p}(A_i) = 2n - 3$ (recall that the concept of the profit $\mathsf{p}(A)$ of a nonterminal A of a 1-level grammar is defined on page 11). Consequently, $|G| = |w| - \sum_{A \in V} \mathsf{p}(A) = 4n^2 - 2m - k(2n - 3)$.

Let $G = (N, \Sigma, R, \mathsf{ax})$ be a 1-level grammar of size at most $4n^2 - 2m - 2kn + 3k$, with at most k nonterminals. We note that, for every $A \in N$, $\mathsf{p}(A) \leq 2n - 3$, since in w every repeated factor has size of at most 3 and is repeated at most n times. Since, by assumption, $|G| \leq 4n^2 - 2m - k(2n - 3)$ and $|G| = 4n^2 - 2m - \sum_{A \in N} \mathsf{p}(A)$, we conclude that $\sum_{A \in N} \mathsf{p}(A) \geq k(2n - 3)$. Hence, there are exactly k nonterminals $A \in N$ each with a right side of length 3, which implies $A \to \# v_i \#$, for some i, $1 \leq i \leq n$, and, furthermore, $|\mathsf{ax}|_A = n$. It can be easily verified that this is only possible if $\{v_i : \text{there is } (A \to \# v_i \#) \in R\}$ is an independent set for $\mathcal{G}$.

The third statement obviously implies the second statement. We assume that the second statement holds, i. e., there is a grammar $G = (N, \Sigma, R, \mathsf{ax})$ for w with at most k nonterminals and $|G| \leq 4n^2 - 2m + 3k - 2kn$. If G is not a 1-level grammar, then it has a rule $A \to \alpha$ with $\alpha \notin \Sigma^+$ and, since the only repeated factors of w with a length of at least 3 have the form $\# x \#$, for some $x \in \{v_1, \ldots, v_n\}$, we also know that $\mathfrak{D}(A) = \# x \#$. In particular, this implies that $\alpha = B\#$ or $\alpha = \# B$ with $B \to \# x \in R$ or $B \to x \# \in R$. Generally, each rule in G has a length (and hence cost) of at least 2, compresses a factor of length at most 3 and occurs in the axiom at most n times. The rules A and B together can occur at most n times in ax, as they both derive the symbol x. This means that the axiom has a length of at least $|w| - (k-1)2n$ and therefore the overall grammar has size of at least $|\mathsf{ax}| + 2k = 4n^2 - 2m - 2kn + 2n + 2k$. Since we assumed that $|G| \leq 4n^2 - 2m + 3k - 2kn$, this implies $4n^2 - 2m - 2kn + 2n + 2k \leq 4n^2 - 2m + 3k - 2kn$, so $2n \leq k$ which contradicts the assumption $k \leq n$. □

Lemma 12 directly yields the following result:

Theorem 11 1-SGP *and* SGP *parameterised by $|N|$ are* $\mathsf{W}[1]$*-hard.*

We emphasise that Theorem 11 shows $\mathsf{W}[1]$-hardness for the smallest grammar problem parameterised by $|N|$ only for the case where the terminal alphabet Σ is unbounded. The most important respective question, which, unfortunately, is left open here, is whether the smallest grammar problem is fixed-parameter tractable with respect to the combined parameter $(|N|, |\Sigma|)$ (we discuss the open cases of the parameterised complexity of the smallest grammar problem in more detail in Section 6).

Finally, we note that we can use Lemma 11 in order to obtain a simple exact exponential-time algorithm for the smallest grammar problem. More precisely, we compute for each subset $F \subseteq \mathsf{F}_{\geq 2}(w)$ a smallest F-grammar, which yields an algorithm with an overall running-time of $2^{\mathcal{O}(|w|^2)}$. In the next section, we present more advanced exact exponential-time algorithms for SGP and 1-SGP.

5 Exact Exponential-Time Algorithms

An obvious approach for an exact exponential-time algorithm for SGP is to enumerate all ordered trees with $|w|$ leaves and to interpret them as derivation trees of a grammar for w. More precisely, for a given ordered tree with $|w|$ leaves, we first label the leaves with the symbols of w and then we inductively label each internal node with $u_1 u_2 \dots u_k$, where u_i are the labels of its children nodes. Finally, for every factor u that occurs as a label of some internal node, we substitute all occurrences of this label by a nonterminal A_u. In order to estimate the number of such trees, we first note that the i^{th} Catalan number C_i is the number of full binary trees (i. e., every non-leaf has exactly two children) with $i + 1$ leaves. Moreover, every tree with $|w|$ leaves can be obtained from a full binary tree with $|w|$ leaves by contracting some of its 'non-leaf' edges (i. e., edges not incident to a leaf). Since every full binary tree with $|w|$ leaves has less than $|w|$ such 'non-leaf' edges, the number of trees that we have to consider is at most $C_{|w|-1} \cdot 2^{|w|}$. Since $C_{|w|-1} \in \mathcal{O}(4^{|w|-1})$, this leads to an algorithm with running-time $\mathcal{O}^*(8^{|w|})$.

In the following, we shall give more sophisticated exact exponential-time algorithms with running times $\mathcal{O}^*(1.8392^{|w|})$, for the 1-level case, and $\mathcal{O}^*(3^{|w|})$, for the multi-level case. First, we need to introduce some helpful notations.

Let $G = (N, \Sigma, R, \mathsf{ax})$ be a grammar for w and let $\alpha = A_1 \dots A_k$, $A_i \in (\Sigma \cup N)$, $1 \leq i \leq k$. The *factorisation of* $\mathfrak{D}(\alpha)$ *induced by* α is the tuple $(\mathfrak{D}_G(A_1), \dots, \mathfrak{D}_G(A_k))$. Furthermore, the factorisation of w induced by ax is called the *factorisation of w induced by G*. A factorisation $q = (u_1, u_2, \dots, u_k)$ of a word w with $|w| = n$ can be characterised by the vector $v_q \in \{0,1\}^{n-1}$ defined by setting $v_q[i] = 1$ if and only if $i = |u_1 \dots u_j|$ for some $1 \leq j < k$. For the sake of convenience, we implicitly assume $v_q[0] = v_q[n] = 1$, and treat vectors as words over the alphabet $\mathbb{N}$, which allows us to use notations already defined for words. From now on, we shall use these two representations of factorisations, i. e., tuples of factors and vectors in $\{0,1\}^{n-1}$, interchangeably, without mentioning it.

5.1 The 1-Level Case

In the 1-level case, as long as we are only concerned with smallest grammars, the factorisation induced by the axiom already fully determines the grammar. More formally, let $q = (u_1, u_2, \dots, u_k)$ be a factorisation for a word w and let $F_q = \{u_i : 1 \leq i \leq k, |u_i| \geq 2\}$. We define the 1-level grammar $G_q = (N_q, \Sigma, R_q, \mathsf{ax}_q)$ by $R_q = \{(A_u, u) : u \in F_q\}$, $N_q = \{A_u : u \in F_q\}$ and $\mathsf{ax}_q = B_1 \dots B_k$ with $B_j = A_{u_j}$, if $u_j \in F_q$ and $B_j = u_j$, otherwise.

Lemma 13 *For any factorisation $q = (u_1, u_2, \dots, u_k)$ for w, G_q is a smallest grammar among all 1-level grammars for w that induce the factorisation q.*

Proof Let $q = (u_1, u_2, \dots, u_k)$ be a factorisation for a word w. Every 1-level grammar $G = (N, \Sigma, R, \mathsf{ax})$ for w that induces q satisfies $|G| = k + \sum_{A \in N} |\mathfrak{D}(A)| \geq k + \sum_{u \in F_q} |u|$. Since $|G_q| = k + \sum_{u \in F_q} |u|$, G_q is a smallest 1-level grammar for w that induces q. □

Choosing the smallest among all grammars $\{G_q : q \text{ is a factorisation of } w\}$ yields an $\mathcal{O}^*(2^n)$ algorithm for 1-SGP. However, it is not necessary to enumerate factorisations that contain at least two consecutive factors of length 1, which improves this result as follows.

Theorem 12 1-SGP *can be solved exactly in polynomial space and in time $\mathcal{O}^*(1.8392^{|w|})$.*

Proof For any $k \in \mathbb{N}$, let Γ_k contain all $q \in \{0,1\}^k$, such that v has no prefix 11, no suffix 11 and no factor 111; furthermore, let Γ'_k contain all $q \in \{0,1\}^k$, such that v has no suffix 11 and no factor 111. Clearly, $\Gamma_{|w|-1}$ contains exactly the factorisations for w that have no consecutive factors of length 1. In order to solve the smallest 1-level grammar problem, we enumerate $\Gamma_{|w|-1}$ and for every $q \in \Gamma_{|w|-1}$, we construct G_p, where p is obtained from q, by replacing every non-repeated factor u of q with the factors $u[1], u[2], \dots, u[|u|]$. It remains to prove the correctness of this algorithm and to estimate its running-time.

To this end, let G be a smallest 1-level grammar for w and let $p = (u_1, u_2, \dots, u_k)$ be the factorisation induced by G. Furthermore, let q be the factorisation obtained from p by joining any maximal sequence $u_i, u_{i+1}, \dots, u_j$, $1 \leq i < j \leq k$, of factors with $|u_\ell| = 1$, $i \leq \ell \leq j$ (note that $q \in \Gamma_{|w|-1}$). If none of the newly constructed factors of q is repeated, then the algorithm, when enumerating q, constructs grammar G_p that, according to Lemma 13, is smallest among all 1-level grammars for w that induce p; thus, G_p is a smallest 1-level grammar. If, on the other hand, any of these newly constructed factors is repeated and has a length of at least 3, or has length 2 and is repeated for at least 3 times, then a 1-level grammar smaller than G could be constructed, which is a contradiction. This leaves the case where all newly constructed factors of q have length 2 and are repeated exactly twice. In this case the algorithm will, when enumerating q, construct a grammar that differs from G_p only in that it compresses some factors of length 2 that are repeated only twice, and that G_p does not compress. This grammar has obviously the same size as G_p and is therefore a smallest 1-level grammar as well.

In order to estimate the running-time, let $T(k) = |\Gamma_k|$ and $T'(k) = |\Gamma'_k|$, for every $k \in \mathbb{N}$. Obviously,

$$T(k) = |\{q \in \Gamma_k \colon q[1] = 0\}| + |\{q \in \Gamma_k \colon q[1] = 1\}|\,,$$

so, in the following, we shall determine $|\{q \in \Gamma_k \colon q[1] = 0\}|$ and $|\{q \in \Gamma_k \colon q[1] = 1\}|$ separately. To this end, we first note that $|\{q \in \Gamma_k \colon q[1] = 1\}| = T(k-1) - T'(k-3)$ (this is due to the fact that $T(k-1)$ also counts all $q = 110q' \ldots$ with $q' \in \Gamma'_{k-3}$, so we have to subtract $T'(k-3)$). Moreover,

$$\begin{aligned}|\{q \in \Gamma_k \colon q[1]q[2] = 01\}| &= T(k-2)\,,\\ |\{q \in \Gamma_k \colon q[1]q[2]q[3] = 001\}| &= T(k-3)\,,\\ |\{q \in \Gamma_k \colon q[1]q[2]q[3] = 000\}| &= T'(k-3)\,.\end{aligned}$$

This is due to the fact that extending the prefix 01 or 001 with 11 yields a factor 111, where the prefix 000 can be extended by 11. With the above observations, we can now conclude the following:

$$\begin{aligned}T(k) &= |\{q \in \Gamma_k \colon q[1] = 0\}| + |\{q \in \Gamma_k \colon q[1] = 1\}|\\ &= |\{q \in \Gamma_k \colon q = 01\ldots\}| + |\{q \in \Gamma_k \colon q = 001\ldots\}| +\\ &\quad |\{q \in \Gamma_k \colon q = 000\ldots\}| + |\{q \in \Gamma_k \colon q[1] = 1\}|\\ &= T(k-2) + T(k-3) + T'(k-3) + T(k-1) - T'(k-3)\\ &= T(k-1) + T(k-2) + T(k-3)\,.\end{aligned}$$

This yields $T(k) = \mathcal{O}(1.8392^k)$; since we can also enumerate $\Gamma_{|w|-1}$ in time $\mathcal{O}^*(1.8392^{|w|})$, the algorithm has a running-time of $\mathcal{O}^*(1.8392^{|w|})$. □

5.2 The Multi-Level Case

The obvious idea for a dynamic programming algorithm is to build up grammars level by level, e. g., by starting with a 1-level grammar, then extending it by a new axiom, which can derive the old axiom in one derivation step, and iterating this procedure. Obviously, we have to try an exponential number of axioms, which will lead to an exponential-time algorithm (as suggested by the NP-completeness of the problem). However, there is a more fundamental problem with this general approach, which shall be pointed out by going a bit more into detail.

For every i and every factorisation p of w, we store in entry $T[i, p]$ of a table the size of a smallest i-level grammar with an axiom ax that induces factorisation p (in the sense defined at the beginning of this section). Then, for every factorisation q, such that p is a refinement of q, we construct a new axiom ax' that induces factorisation q and that can derive ax in one step, which is treated as the axiom of a new $(i+1)$-level grammar. We subtract the profit of the rules needed to derive ax from ax' to $T[i,p]$ and store the obtained number in $T[i+1,q]$. Note that the axioms ax and ax' are fully determined by the factorisations p and q (similar as a factorisation determines a smallest 1-level grammar with an axiom inducing this factorisation, see Lemma 13). However, this approach is fundamentally flawed, since in order to compute the size of the new $(i+1)$-level grammar, we need to know whether the rules needed to derive ax from ax' have already been used earlier in the i-level grammar and therefore are already counted by $T[i,p]$, or whether they are newly introduced. On the other hand, it should clearly be avoided to additionally store all previously used rules as well.

To overcome this problem, we do not consider the levels of a grammar as strings $\mathsf{ax}, \mathsf{D}(\mathsf{ax}), \mathsf{D}(\mathsf{D}(\mathsf{ax})), \ldots, w$, which is the obvious choice, but we define them in such a way that all occurrences of a nonterminal are on the same level. With this definition, all the rules that are needed for the extension to the new level must be completely new rules without prior application; thus, a dynamic programming approach similar to the one described above will be successful. Next, we give the required definitions (which are also illustrated by Example 2).

For a d-level grammar $G = (N, \Sigma, R, \mathsf{ax})$, we partition the set of nonterminals N according to the number of derivation steps that are necessary to derive a terminal word (or, equivalently, according to their height, i. e., the maximum distance to a leaf in the derivation tree). More precisely, let $N_1, \ldots, N_d$ be the partition of N into $N_i = \{A \in N \colon (\mathsf{D}^i_G(A) \in \Sigma^+) \wedge (\mathsf{D}^{i-1}_G(A) \notin \Sigma^+)\}$. We recall that the morphism $\mathsf{D} : (N \cup \Sigma)^* \to (N \cup \Sigma)^*$ replaces every occurrence of a nonterminal by the right side of its rule. For every i, $1 \leq i \leq d$, we modify D, such that it only considers nonterminals from N_i and ignores the rest. More formally, for every i, $1 \leq i \leq d$, we define a morphism $\widehat{\mathsf{D}}_i \colon (N \cup \Sigma)^* \to (N \cup \Sigma)^*$ component-wise by $\widehat{\mathsf{D}}_i(x) = \mathsf{D}(x)$, if $x \in N_i$ and $\widehat{\mathsf{D}}_i(x) = x$, otherwise. Using these morphisms, we now inductively define the *levels* L_i, $0 \leq i \leq d$, of G by $\mathsf{L}_d = \mathsf{ax}$ and, for every i, $0 \leq i \leq d-1$, $\mathsf{L}_i = \widehat{\mathsf{D}}_{i+1}(\mathsf{L}_{i+1})$.

Observation 4 *The sequence $\mathsf{L}_d, \mathsf{L}_{d-1}, \ldots, \mathsf{L}_1, \mathsf{L}_0$ is a derivation with $\mathsf{L}_d = \mathsf{ax}$, $\mathsf{L}_0 = w$ and, by a simple induction over i, it can be verified that, for every i, $1 \leq i \leq d$, all applications of rules for nonterminals from N_i happen in the single derivation step from L_i to L_{i-1}. In particular, this implies that, for every i, $1 \leq i \leq d$, L_i contains all occurrences of nonterminals $A \in N_i$ that are ever derived in the derivation of w or, in other words, for every j, $0 \leq j \leq i-1$, $\sum_{A \in N_i} |\mathsf{L}_j|_A = 0$.*

Since in the derivation $\mathsf{L}_d, \mathsf{L}_{d-1}, \ldots, \mathsf{L}_1, \mathsf{L}_0$ occurrences of a nonterminal A are not derived until all of them are collected in L_i and then they are derived all at once in the same derivation step, we can conveniently define the term *profit* for all rules (of the d-level grammar G) as follows. For every i, $1 \leq i \leq d$, we define the profit of every $A \in N_i$ by $\mathsf{p}(A) = |\mathsf{L}_j|_A(|\mathsf{D}(A)| - 1) - |\mathsf{D}(A)|$. Note that for $d = 1$ this corresponds to the definition of profit for 1-level grammars as introduced on page 11. In particular, we can now express the size of a grammar in terms of the profit of its rules:

Proposition 7 *Let G be a grammar. Then* $|G| = |w| - (\sum_{i=1}^{d} \sum_{A \in N_i} \mathsf{p}(A))$.

Proof We recall that, by definition of the size of a grammar and as a conclusion of Observation 4, we have

$$|G| = \left(\sum_{i=1}^{d} \sum_{A \in N_i} |\mathsf{D}(A)| \right) + |\mathsf{ax}| \,, \quad |w| = \left(\sum_{i=1}^{d} \sum_{A \in N_i} |\mathsf{L}_i|_A (|\mathsf{D}(A)| - 1) \right) + |\mathsf{ax}| \,.$$

Consequently,

$$\begin{aligned}
&|w| - \left(\sum_{i=1}^{d} \sum_{A \in N_i} \mathsf{p}(A) \right) = |w| - \left(\sum_{i=1}^{d} \sum_{A \in N_i} |\mathsf{L}_i|_A (|\mathsf{D}(A)| - 1) - |\mathsf{D}(A)| \right) = \\
&|w| - \left(\left(\sum_{i=1}^{d} \sum_{A \in N_i} |\mathsf{L}_i|_A (|\mathsf{D}(A)| - 1) \right) - \left(\sum_{i=1}^{d} \sum_{A \in N_i} |\mathsf{D}(A)| \right) \right) = \\
&|w| - ((|w| - |\mathsf{ax}|) - (|G| - |\mathsf{ax}|)) = |G| \,.
\end{aligned}$$

□

Example 2 Let $G = (N, \Sigma, R, \mathsf{ax})$ with $N = \{A, B, C, D\}$, $\Sigma = \{\mathsf{a}, \mathsf{b}\}$, $R = \{A \to D\mathsf{bb}, B \to \mathsf{ab}, C \to AB, D \to \mathsf{aaa}\}$ and $\mathsf{ax} = CDC$ be the 3-level grammar illustrated in Figure 4. According to the definitions from above, the partition of N is $N_1 = \{B, D\}$, $N_2 = \{A\}$, $N_3 = \{C\}$, and the levels are

$$\begin{array}{lclcl}
\mathsf{L}_3 & = & \mathsf{ax} & = & CDC \,, \\
\mathsf{L}_2 & = & \widehat{\mathsf{D}}_3(CDC) & = & ABDAB \,, \\
\mathsf{L}_1 & = & \widehat{\mathsf{D}}_2(ABDAB) & = & D\mathsf{bb}BDD\mathsf{bb}B \,, \\
\mathsf{L}_0 & = & \widehat{\mathsf{D}}_1(D\mathsf{bb}BDD\mathsf{bb}B) & = & \mathtt{aaabbabaaaaaabbab} \,.
\end{array}$$

Note that, for every i, $1 \leq i \leq 3$, L_i contains all occurrences of all nonterminals from N_i and the rules for all nonterminals N_i are exclusively applied in deriving L_{i-1} from L_i. In particular, note that in the derivation $\mathsf{L}_3, \ldots, \mathsf{L}_0$, the derivation of occurrences of nonterminals B and D is delayed until the very last derivation step.

Furthermore, the profits are as follows

$$\begin{aligned}
\mathsf{p}(A) &= |\mathsf{L}_2|_A (|\mathsf{D}(A)| - 1) - |\mathsf{D}(A)| = 2(3 - 1) - 3 = 1, \\
\mathsf{p}(B) &= |\mathsf{L}_1|_B (|\mathsf{D}(B)| - 1) - |\mathsf{D}(B)| = 2(2 - 1) - 2 = 0, \\
\mathsf{p}(C) &= |\mathsf{L}_3|_C (|\mathsf{D}(C)| - 1) - |\mathsf{D}(C)| = 2(2 - 1) - 2 = 0, \\
\mathsf{p}(D) &= |\mathsf{L}_1|_D (|\mathsf{D}(D)| - 1) - |\mathsf{D}(D)| = 3(3 - 1) - 3 = 3 \,.
\end{aligned}$$

Moreover, $|w| - \sum_{A \in N} \mathsf{p}(A) = 17 - 4 = 13$ and $|G| = |\mathsf{ax}| + |\mathsf{D}(A)| + |\mathsf{D}(B)| + |\mathsf{D}(C)| + |\mathsf{D}(D)| = 3 + 3 + 2 + 2 + 3 = 13$.

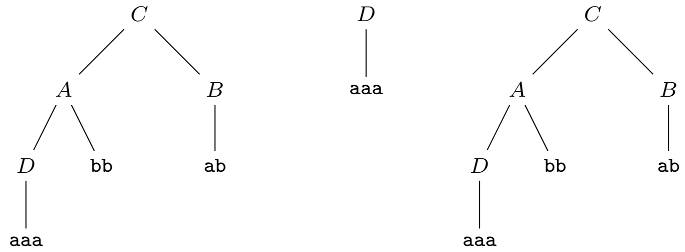

Fig. 4 A derivation tree for 3-level grammar (neighbouring leaves are combined, the start rule is omitted).

Before we formally present the dynamic programming algorithm, we sketch its behaviour in a more intuitive way. We first need the following definition. A factorisation $p = (u_1, u_2, \ldots, u_k)$ is a *refinement* of a factorisation $q = (v_1, v_2, \ldots, v_m)$, denoted by $p \preceq q$, if $(u_{j_{i-1}+1}, u_{j_{i-1}+2}, \ldots, u_{j_i})$ is a factorisation of v_i, $1 \leq i \leq m$, for some $\{j_i\}_{0 \leq i \leq m}$, with $0 = j_0 < j_1 < \ldots < j_m = k$.

The algorithm runs through steps $i = 1, 2, \ldots, \frac{w}{2}$ and in step i, it considers all possibilities for two factorisations q_{i-1} and q_i of w induced by L_{i-1} and L_i, respectively (note that this implies $q_{i-1} \preceq q_i$). The differences between q_{i-1} and q_i implicitly define N_i as follows. Let $q_i = (v_1, v_2, \ldots, v_k)$ and let $q_{i-1} = (u_1, u_2, \ldots, u_\ell)$, which, since $q_{i-1} \preceq q_i$, means that for some j_i, $0 \leq i \leq k$, with $1 = j_0 < j_1 < \ldots < j_k = \ell + 1$, $(u_{j_{i-1}}, u_{j_{i-1}+1}, \ldots, u_{j_i-1})$ is a factorisation of v_i, $1 \leq i \leq k$. If $j_s - j_{s-1} > 1$ for some $1 \leq s \leq k$, N_i contains a nonterminal A with $|\mathsf{D}(A)| = j_s - j_{s-1}$ and $\mathfrak{D}(A) = v_s$. The number $|\mathsf{L}_i|_A$ is also implicitly given by counting how often the sequence of factors $(u_{j_{s-1}+1}, \ldots, u_{j_s})$ independently occurs in q_{i-1} and is combined into one single factor in q_i; more precisely, $|\mathsf{L}_i|_A = |\{t \colon (u_{j_{t-1}+1}, \ldots, u_{j_t}) = (u_{j_{s-1}+1}, \ldots, u_{j_s})\}|$. This allows to calculate the profit of the rule for A without knowing the exact structure of the rules for nonterminals in N_j with $j \neq i$. By Lemma 13, this choice of nonterminals for N_i is optimal for the fixed induced factorisations, which means that a search among all choices for q_{i-1} and q_i yields a smallest i-level grammar for w. The running time of this algorithm is dominated by enumerating all pairs q_{i-1} and q_i of factorisations of w. However, due to $q_{i-1} \preceq q_i$, these pairs can be compressed as vectors $\{0, 1, 2\}^{|w|-1}$ (the entries denote whether the corresponding position in w is factorised by both (entry '1'), only by the refinement (entry '2') or none (entry '0') of the factorisations). Hence, enumerating these pairs of vectors can be done in time $\mathcal{O}(3^{|w|})$.

Theorem 13 SGP *can be solved in time and space* $\mathcal{O}^*(3^{|w|})$.

Proof Let $n = |w|$. We use dynamic programming to consider all possible factorisations of w and refinements for each level $i = 1, \ldots, d$. A factorisation of w is stored as a vector $q \in \{0, 1\}^{n-1}$ and, furthermore, we use vectors $q \in \{0, 1, 2\}^{n-1}$ in order to represent a factorisation together with a refinement, as explained above (for the sake of convenience, we implicitly assume $q[0] = q[n] = 1$). For such a vector $q \in \{0, 1, 2\}^{n-1}$ that describes two factorisations

p and p' with $p \preceq p'$, we denote by $F(q)$ the factorisation p' (represented as a vector from $\{0,1\}^{n-1}$) and by $R(q)$ the refinement p (represented as a vector from $\{0,1\}^{n-1}$). More formally, let $F\colon \{0,1,2\}^{n-1} \to \{0,1\}^{n-1}$ be a mapping that replaces each '2'-entry by a '0'-entry (and leaves all other entries unchanged), and let $R\colon \{0,1,2\}^{n-1} \to \{0,1\}^{n-1}$ be a mapping that replaces each '2'-entry by a '1'-entry (and leaves all other entries unchanged).

The dynamic program uses the following tables:

- $T[i,q]$ for $i \in \{2,\ldots,\frac{n}{2}\}$ and all $q \in \{0,1,2\}^{n-1} \setminus \{0,1\}^{n-1}$ stores the size of a smallest i-level grammar for w for which the axiom ax induces the factorisation $F(q)$ and for which $\widehat{\mathsf{D}}_i(\mathsf{ax})$ induces the factorisation $R(q)$.
- $S[i,q]$ for all $i \in \{1,\ldots,\frac{n}{2}\}$ and all $q \in \{0,1\}^{n-1}$ stores the size of a smallest i-level grammar for w for which the axiom induces the factorisation q.
- $P[i,q]$ for all $i \in \{2,\ldots,\frac{n}{2}\}$ and all $q \in \{0,1\}^{n-1}$ stores the refinement of q which equals the factorisation induced by $\widehat{\mathsf{D}}_i(\mathsf{ax})$ for an optimal i-level grammar for which ax induces factorisation q.
- opt_i for all $i \in \{1,\ldots,\frac{n}{2}\}$ stores the value of a smallest i-level grammar for w.

We point out that the tables T and S are sufficient to compute the size of a smallest grammar; the purpose of table P is to construct an actual grammar of minimal size after termination of the algorithm. Intuitively speaking, in order to determine $S[i,q]$, i. e., the size of a smallest i-level grammar for which the axiom induces the factorisation q, we have to check all entries $T[i,q']$ for which the factorisation of q' (note that q' represents a factorisation *and* a refinement) equals q and for a minimal one of these entries, we store the actual refinement (which is not needed anymore to compute the size of a minimal grammar) in $P[i,q]$. In this way, the entries of $P[i,q]$ allow us to restore an actual smallest grammar.

We first initialise S by setting $S[1,q] = |G_q|$, for every $q \in \{0,1\}^{n-1}$, where, according to Lemma 13, G_q is a smallest 1-level grammar for w that induces factorisation q, and we set $opt_1 = \min\{S[1,q] \colon q \in \{0,1\}^{n-1}\}$.

We then compute iteratively for each $i = 2,\ldots,\frac{n}{2}$ the entries $T[i,q]$, $S[i,q']$ and $P[i,q']$, for every $q \in \{0,1,2\}^{n-1} \setminus \{0,1\}^{n-1}$ and $q' \in \{0,1\}^{n-1}$ as follows.

First, for any $q \in \{0,1,2\}^{n-1} \setminus \{0,1\}^{n-1}$, we define the set $I(q)$ of consecutive factors in $R(q)$ which are combined into one factor in $F(q)$:

$$\begin{aligned} I(q) := \{(j_0, j_1, \ldots, j_k)\colon\ & |q[j_0 - 1..j_k]|_1 = |q[j_0 - 1]q[j_k]|_1 = 2, \\ & |q[j_0..j_k]|_2 = |q[j_1]\ldots q[j_{k-1}]|_2 = k - 1 \geq 1\}\,. \end{aligned}$$

Furthermore, from $I(q)$, we can extract the set $N(q)$ of nonterminals which create these factors on level i, i. e., $N(q) := \{w(j_0, j_1, \ldots, j_k)\colon (j_0, \ldots, j_k) \in I(q)\}$, where

$$w(j_0, j_1, \ldots, j_k) := (w[j_0 + 1..j_1], w[j_1 + 1..j_2], \ldots, w[j_{k-1} + 1..j_k])\,.$$

The corresponding number of occurrences of the nonterminal $w(j_0, j_1, \dots, j_k)$ on level i is given by

$$c(j_0, j_1, \dots, j_k) := |\{(j'_0, j'_1, \dots, j'_k) \in I(q) \colon w(j_0, j_1, \dots, j_k) = w(j'_0, j'_1, \dots, j'_k)\}| \, .$$

The entry $T[i, q]$ can now be computed as follows:

$$T[i, q] = S[i-1, R(q)] - \left(\sum_{w(j_0, j_1, \dots, j_k) \in N(q)} c(j_0, j_1, \dots, j_k)(k-1) - k \right)$$

Then, for every $q' \in \{0,1\}^{n-1}$, we can compute entries $S[i, q']$ and $P[i, q']$ by

$$S[i, q'] = \min\{T[i, q] \colon F(q) = q'\} \text{ and}$$
$$P[i, q'] = q \, ,$$

where $q \in \{0,1,2\}^{n-1} \setminus \{0,1\}^{n-1}$ with $F(q) = q'$ and $T[i, q] = S[i, q']$. Finally, the value opt_i is computed by $opt_i = \min\{S[i, q'] \colon q' \in \{0,1\}^{n-1}\}$.

After termination of step $\frac{n}{2}$, the size of a smallest grammar for the word w is $\min\{opt_i \colon 1 \leq i \leq \frac{n}{2}\}$. Since the values in $T[i, q]$ for any $i = 2, 3, \dots, \frac{n}{2}$ and $q \in \{0,1,2\}^{n-1} \setminus \{0,1\}^{n-1}$ are constructively computed from $S[i, R(q)]$ by defining the rules in $N(q)$, the set $\bigcup_{j=1}^{i} N(q_i)$ with $q_i := q$ and $q_{j-1} := P[j, q_j]$ for $j = i-1, \dots, 1$ yields an i-level grammar for w of size $T[i, q]$. For the index i with $opt_i = \min\{opt_i \colon 1 \leq i \leq \frac{n}{2}\}$ and a vector $q \in \{0,1,2\}^{n-1} \setminus \{0,1\}^{n-1}$ such that $opt_i = S[i, R(q)]$, this construction gives a smallest grammar for w.

In order to prove the correctness of the algorithm, we show for each $q \in \{0,1\}^{n-1}$, inductively for each $i = 1, \dots, \frac{n}{2}$ that $S[i, q]$ equals the size of a smallest i-level grammar for w which induces the factorisation q. For $i = 1$ this is implied by Lemma 13. Assuming that this statement is true for some value $i-1$, let $G_i = (N, \Sigma, R, \mathsf{ax})$ be a smallest i-level grammar for w with $i \leq \frac{n}{2}$. Let q_i and q_{i-1} be the vector-representations of the factorisations induced by ax and $\widehat{\mathsf{D}}_i(\mathsf{ax})$ respectively. The grammar $G_{i-1} := (N \setminus N_i, \Sigma, R \setminus \{(A, \mathsf{D}(A)) \colon A \in N_i\}, \widehat{\mathsf{D}}_i(\mathsf{ax}))$ is an $(i-1)$-level grammar for w with induced factorisation q_{i-1} and the size of G_{i-1} can be computed by $|G_i| + \sum_{A \in N_i} \mathsf{p}(A)$ and is at least $S[i-1, q_{i-1}]$ by the induction hypothesis. By definition of the profit, the term $|G_i| + \sum_{A \in N_i} \mathsf{p}(A)$ can be re-written to $|G_i| + |\widehat{\mathsf{D}}_i(\mathsf{ax})| - |\mathsf{ax}| - \sum_{A \in N_i} |\mathsf{D}(A)|$.

Let $q \in \{0,1,2\}^{n-1}$ be such that $F(q) = q_i$ and $R(q) = q_{i-1}$, i. e., for every j, $1 \leq j \leq n-1$, $q[j] = 2$, if $q_i[j] \neq q_{i-1}[j]$ and $q[j] = q_i[j]$, otherwise. The value $T[i, q]$ is computed from $S[i-1, q_{i-1}]$ by subtracting

$$\sum_{w(j_0, j_1, \dots, j_k) \in N(q)} c(j_0, j_1, \dots, j_k)(k-1) - k =$$
$$\left(\sum_{(j_0, \dots, j_k) \in I(q)} (k-1) \right) - \left(\sum_{w(j_0, \dots, j_k) \in N(q)} k \right) .$$

Each 2-entry in q occurs in exactly one set in $I(q)$ which, by definition of q, yields:

$$\sum_{(j_0,j_1,\ldots,j_k)\in I(q)} (k-1) = \sum_{j=1}^{n-1}(q_{i-1}[j] - q_i[j]) = |\widehat{\mathsf{D}}_i(\mathsf{ax})| - |\mathsf{ax}|\,.$$

For each $w(j_0, j_1, \ldots, j_k) \in N(q)$, N_i contains a nonterminal $A \in N_i$ with $|\mathsf{D}(A)| = k$, which means that $\sum_{A\in N_i}|\mathsf{D}(A)| \geq \sum_{w(j_0,j_1,\ldots,j_k)\in N(q)} k$; thus,

$$\begin{aligned} |G_i| &= |G_{i-1}| - |\widehat{\mathsf{D}}_i(\mathsf{ax})| + |\mathsf{ax}| + \sum_{A\in N_i} |\mathsf{D}(A)| \\ &\geq S[i-1, q_{i-1}] - \sum_{w(j_0,j_1,\ldots,j_k)\in N(q)} c(j_0, j_1, \ldots, j_k)(k-1) - k \\ &= T[i, q] \geq S[i, F(q)] = S[i, q_i]\,. \end{aligned}$$

Consequently, the algorithm computes the size of a grammar for w that is smallest among all grammars for w with at most $\frac{n}{2}$ levels and since for any word w there always exists a smallest grammar with at most $\frac{|w|}{2}$ levels, we conclude that the described algorithm finds a smallest grammar for w. □

We conclude this section by pointing out some features of the algorithm of Theorem 13. First, note that the brute-force enumeration of all $q \in \{0,1,2\}^{n-1} \setminus \{0,1\}^{n-1}$, which dominates the running-time, provides some possibilities for modifications. For example, if we only consider q such that at most 2 neighbouring factors of $R(q)$ are combined in $F(q)$ (which are much less than the full set $\{0,1,2\}^{n-1} \setminus \{0,1\}^{n-1}$), then we automatically compute smallest grammars in *Chomsky normal form*.[11] Moreover, for a fixed i and two $q_1, q_2 \in \{0,1,2\}^{n-1} \setminus \{0,1\}^{n-1}$, the computations that are necessary to compute $T[i, q_1]$ and $T[i, q_2]$ are independent from each other and only require the previously computed values $S[i-1, \cdot]$ (an analogous observation can be made for the computation of the $S[i, \cdot]$ and $P[i, \cdot]$). Hence, the brute-force enumeration of the $q \in \{0,1,2\}^{n-1} \setminus \{0,1\}^{n-1}$ and of the $q' \in \{0,1\}^{n-1}$ can be easily done in parallel.

6 Conclusions

We conclude this work by discussing some important open problems and additional questions that are motivated by our results.

[11] The restriction to grammars in Chomsky normal form is quite common, since also many of the existing approximation algorithms compute grammars in Chomsky normal form.

6.1 Small Alphabets

For hard problems on strings, we usually encounter the situation that either the problem becomes polynomial-time solvable for constant alphabets, or there is a hardness reduction that works for some constant alphabet, which, by simple encoding techniques, extends to binary alphabets as well. Moreover, the unary case is often trivially solvable in polynomial time, even if the problem becomes intractable for larger alphabets. However, the smallest grammar problem shows a drastically different behaviour: it is not polynomial-time solvable for every constant alphabet (unless $\mathsf{P} = \mathsf{NP}$), but the NP-hardness for very small alphabets (even for the binary or unary case) is still open. Thus, we consider the following as one of the most important open questions:

Open Problem 1 *Is it possible to compute smallest grammars for binary alphabets in polynomial time?*

We believe that answering this question in the negative might be rather difficult. In fact, the substantial effort that was necessary to prove Theorem 3 suggests that further strengthening our reduction to the case of binary alphabets is problematic. Thus, a completely different kind of reduction seems necessary. However, the main technical challenge seems to be the necessity to control the compression of factors that function as codewords for parts of the source problem of the reduction. It is arguably difficult to think about reductions that somehow circumvents this issue.

On the other hand, it is not apparent how a small alphabet could help in order to efficiently compute smallest grammars and, if this is possible, it seems that deeper combinatorial insights with respect to grammar-based compression are necessary.

6.2 Approximation

So far, no constant-factor approximation algorithm is known for the smallest grammar problem (as already mentioned in Section 1.3, the best approximation algorithms achieve a ratio in $\mathcal{O}\left(\log\left(\frac{|w|}{m^*}\right)\right)$ [54,14,39]) and, although not backed by any hardness results, the existing literature suggests that no such algorithm exists. Moreover, this apparent hardness of approximating smallest grammars also applies to the case of fixed alphabets, since, as shown in [33], if there is an approximation algorithm for the smallest grammar problem over a binary alphabet with a constant approximation ratio c, then there also is a $6c$-approximation algorithm for arbitrary alphabets. This especially means that disproving the existence of a 6-approximation for the smallest grammar problem for unbounded alphabets, under some complexity theoretic assumption, implies, under the same assumption, that there is no polynomial algorithm for the restriction to binary alphabets. Considering the substantial effort that went into designing a reduction for alphabet size 17 in this paper, such an inapproximability result for unbounded alphabets might actually be an easier way to show computational lower bounds for binary alphabets.

Aside from these consequences for binary alphabets, an inapproximability result (with some ratio significantly larger than the current bound of $\frac{8569}{8568}$) for the smallest grammar problem would be very interesting, yet not unexpected. The common belief that general constant-factor approximations probably do not exist is based on the fact that, despite substantial effort, such algorithms have not been found so far, but also on the close relation to the problem of computing shortest *addition chains* for a set of integers — a problem which has been extensively studied for over 100 years (see [59] for a survey on addition chains and [39,14] for their connections to the smallest grammar problem). Formally, an addition chain is a strictly increasing sequence $(a_1, a_2, \dots, a_k) \in \mathbb{N}^k$ with $a_1 = 1$ and, for every i, $2 \leq i \leq k$, there are $b, c \in \{a_1, \dots, a_{i-1}\}$ with $a_i = b + c$; the task is to compute a desirably short addition chain that contains a given set of integers. In a sense, grammars can be seen as the natural extension of addition chains (i. e., instead of integers, we are concerned with strings and integer-addition becomes string-concatenation).

It has been shown in [39,14], that a set of integers can be translated into a word (over an alphabet that grows with the number of integers), the smallest grammar of which is larger than the length of a shortest addition chain of the integers by only a constant factor. Consequently, an approximation algorithm for the smallest grammar problem with approximation ratio in $\mathrm{o}(\frac{\log n}{\log \log n})$ would imply an improvement of long-standing results for addition chains, for which the best known approximation algorithm achieves an approximation ratio in $\mathcal{O}(\frac{\log n}{\log \log n})$ (see [39] for details). Note that, with the results of [33] mentioned above, this statement also holds for the case of constant, even binary, alphabets.

Moreover, we can also observe that the fundamental technique of the approximation algorithms of [54,14,39], which links smallest grammars with the size of LZ77-factorisations, is unlikely to prove an approximation with ratio in $\mathrm{o}(\frac{\log n}{\log \log n})$. More precisely, by bounding the size of a smallest grammar of a word from below by the length of its shortest LZ77-factorisation, the performance of these algorithms is shown by comparison with this LZ77-bound. However, it is also shown (see [54,14]) that there are words, for which a smallest grammar is $\mathcal{O}(\frac{\log n}{\log \log n})$-times as large as the size of a smallest LZ77-factorisation; thus, for such algorithms, an approximation-ratio better than $\mathcal{O}(\frac{\log n}{\log \log n})$ cannot be shown by this technique. Moreover, note that this result is improved in [33], where *binary* words are presented, for which a smallest grammar is $\mathcal{O}(\frac{\log n}{\log \log n})$-times as large as the size of a smallest LZ77-factorisation.

Open Problem 2 *Is there a constant-factor approximation algorithm for the smallest grammar problem? (Note that a negative result disproving a ratio of* 6 *or larger, yields a bound for the restriction to binary alphabets.)*

6.3 Parameterised Complexity

This work can also be seen as the starting point of a comprehensive parameterised complexity analysis of the smallest grammar problem. More precisely, our results show that the problem is most likely not in FPT, if parameterised by $|\Sigma|$, $|N|$ or the number of levels. However, with respect to parameter $|N|$, we saw that it is at least in XP. A simple fixed-parameter tractable case can be obtained, if we parameterise by both $|\Sigma|$ and $\ell = \max\{|\mathfrak{D}(A)| \colon A \in N\}$. More precisely, for every $F \subseteq \{u \colon u \in \Sigma^+, 2 \leq |u| \leq \ell\}$, we compute a smallest F-grammar according to Lemma 11 and we output one that is minimal among them. Since the number of the sets F is bounded by a function of the parameters, this yields an fpt-algorithm. However, we consider the following parameterised variant, for which the existence of an fpt-algorithm is still open, the most interesting:

Open Problem 3 *Is the smallest grammar problem parameterised by $|\Sigma|$ and $|N|$ fixed-parameter tractable?*

6.4 A More Abstract View

From a rather abstract point of view, one could generally interpret *any* set of factors $F \subseteq 2^{\Sigma^*}$ as a grammar. More precisely, an *F-grammar* is then a triple $G_F = (N, \Sigma, R)$ (the axiom or start symbol is intentionally missing) with $N = \{A_u \colon u \in F\}$ and R is a set of rules over Σ and N that satisfies $\mathfrak{D}(A_u) = u$, for every $u \in F$. In this way, an F-grammar is a representation of F (just that none of the words in F is the designated compressed word). Obviously, there is a large element of freedom in this definition of F-grammars, since many choices for R are possible. However, as long as we are only interested in small grammars, this is justified, since a grammar that is a smallest among all F-grammars (in the sense described above) can be computed in polynomial time. To see this, we can slightly adapt the approach from Section 4 as follows. For every $u \in F$, we first construct the subgraph with vertices $V_{4,u}$ and edges $E_{4,u}$, then we delete all vertices (u, i, j) with $i < j$ and $u[i..j] \notin F$ (and adjacent edges). As before, it can be shown that an independent dominating set for the resulting interval graph corresponds to a smallest F-grammar. In the following, we denote by G_F the smallest F-grammar obtained in this way.

In a sense, this abstracts away the question of how factors are compressed by other factors and boils the problem of computing small grammar down to its core of hardness, which relies in choosing the right factors. While this perspective is interesting from a theoretical point of view, it also yields questions that might have algorithmic application. For example, as an alternative to the exponential brute-force enumeration of all $F \subseteq \mathsf{F}_{\geq 2}(w)$ in order to obtain an F-grammar that is smallest among all grammars, one could compute G_F for a factor set F that is *inclusion maximal* in the sense that, for every $F' \supsetneq F$, $|G_F| < |G_{F'}|$ (or *inclusion minimal*, which can be defined analogously). However, this approach only seems applicable in a reasonable way, if this concept of inclusion maximality is monotone, i. e., the inclusion maximality of F is characterised by $|G_F| < |G_{(F \cup \{u\})}|$, for every $u \in \Sigma^*$. In this regard, note that $|G_F| = |G_{F'}|$ is possible for $F \subsetneq F'$, as witnessed by $F = \{\mathtt{a}^4\}$ and $F = \{\mathtt{a}^4, \mathtt{a}^2\}$.

Open Problem 4 *Are there $F_1 \subsetneq F_2 \subsetneq F_3 \subseteq \mathsf{F}_{\geq 2}(w)$, such that $|G_{F_1}| < |G_{F_2}|$ and $|G_{F_3}| < |G_{F_1}|$?*

If the inclusion maximality is monotone, then every inclusion maximal F (thus, also an optimal F for which G_F is a smallest grammar) can be computed by starting with $F = \{w\}$ and iteratively adding factors from w, until every possible new factor would increase the size of G_F. This also yields an obvious greedy strategy: always choose the new factor that results in a smallest G_F. In this regard, we stress the fact that this kind of greedy strategy differs from the algorithm Greedy [4], analysed in [39,14], since the latter iteratively changes an existing grammar and the greediness is with respect to the rules of the intermediate grammars.

This also points out an interesting fact (and a potential difficulty) of this approach: The grammars corresponding to the factor sets F, $F \cup \{u\}$, $F \cup \{u, u'\}$ and so on, i. e., the grammars G_F, $G_{(F \cup \{u\})}$, etc., could be quite different and do not necessarily share the incremental character of the factor sets, in the sense that one grammar can be obtained from the previous one by small, local modifications.

Acknowledgments

Katrin Casel was supported by the Deutsche Forschungsgemeinschaft (FE 560/6-1). Serge Gaspers is the recipient of an Australian Research Council (ARC) Future Fellowship (FT140100048) and acknowledges support under the ARC's Discovery Projects funding scheme (DP150101134). NICTA is funded by the Australian Government through the Department of Communications and the ARC through the ICT Centre of Excellence Program.

We thank Gabriele Fici for pointing us to the "rule number" variant of the smallest grammar problem that we discussed at the end of Section 3.3.

References

1. Akutsu, T.: A bisection algorithm for grammar-based compression of ordered trees. Information Processing Letters **110**(18-19), 815–820 (2010)
2. Alimonti, P., Kann, V.: Some APX-completeness results for cubic graphs. Theoretical Computer Science **237**(1-2), 123–134 (2000)
3. Alspach, B., Eades, P., Rose, G.: A lower-bound for the number of productions required for a certain class of languages. Discrete Applied Mathematics **6**(2), 109–115 (1983)
4. Apostolico, A., Lonardi, S.: Off-line compression by greedy textual substitution. Proceedings of the IEEE **88**, 1733–1744 (2000)
5. Arpe, J., Reischuk, R.: On the complexity of optimal grammar-based compression. In: 2006 Data Compression Conference (DCC 2006), 28-30 March 2006, Snowbird, UT, USA, pp. 173–182 (2006)

6. Ausiello, G.: Complexity and approximation: combinatorial optimization problems and their approximability properties. Springer (1999)
7. Benz, F., Kötzing, T.: An effective heuristic for the smallest grammar problem. In: Genetic and Evolutionary Computation Conference, GECCO '13, Amsterdam, The Netherlands, July 6-10, 2013, pp. 487–494 (2013)
8. Berman, P., Karpinski, M., Larmore, L.L., Plandowski, W., Rytter, W.: On the complexity of pattern matching for highly compressed two-dimensional texts. Journal of Computer and System Sciences **65**(2), 332–350 (2002)
9. Bille, P., Lohrey, M., Maneth, S., Navarro, G.: Computation over compressed structured data (dagstuhl seminar 16431). Dagstuhl Reports **6**(10), 99–119 (2016). DOI 10.4230/DagRep.6.10.99. URL `https://doi.org/10.4230/DagRep.6.10.99`
10. Böttcher, S., Lohrey, M., Maneth, S., Rytter, W.: 08261 abstracts collection - structure-based compression of complex massive data. In: Structure-Based Compression of Complex Massive Data, 22.06. - 27.06.2008 (2008). URL `http://drops.dagstuhl.de/opus/volltexte/2008/1694/`
11. Bourgeois, N., Croce, F.D., Escoffier, B., Paschos, V.T.: Fast algorithms for min independent dominating set. Discrete Applied Mathematics **161**(4-5), 558–572 (2013)
12. Bucher, W., Maurer, H.A., II, K.C., Wotschke, D.: Concise description of finite languages. Theoretical Computer Science **14**, 227–246 (1981)
13. Carrascosa, R., Coste, F., Gallé, M., López, G.G.I.: Searching for smallest grammars on large sequences and application to DNA. Journal of Discrete Algorithms **11**, 62–72 (2012)
14. Charikar, M., Lehman, E., Liu, D., Panigrahy, R., Prabhakaran, M., Sahai, A., Shelat, A.: The smallest grammar problem. IEEE Transactions on Information Theory **51**(7), 2554–2576 (2005)
15. Cygan, M., Fomin, F., Kowalik, L., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., Saurabh, S.: Parameterized Algorithms. Springer (2015)
16. Downey, R.G., Fellows, M.R.: Fixed parameter tractability and completeness. Congressus Numerantium **87**, 161–187 (1992)
17. Downey, R.G., Fellows, M.R.: Fundamentals of Parameterized Complexity. Texts in Computer Science. Springer (2013)
18. Eberhard, S., Hetzl, S.: Compressibility of finite languages by grammars. In: Descriptional Complexity of Formal Systems - 17th International Workshop, DCFS 2015, Waterloo, ON, Canada, June 25-27, 2015. Proceedings, pp. 93–104 (2015)
19. Farber, M.: Independent domination in chordal graphs. Operations Research Letters **1**(4), 134–138 (1982)
20. Filmus, Y.: Lower bounds for context-free grammars. Information Processing Letters **111**, 895–898 (2011)
21. Flum, J., Grohe, M.: Parameterized Complexity Theory. Springer (2006)
22. Fournier, J.C.: Colorations des arêtes d'un graphe. Cahiers Centre Études Recherche Opér. **15**, 311–314 (1973). Colloque sur la Théorie des Graphes (Brussels, 1973)
23. Gallé, M.: Searching for compact hierarchical structures in DNA by means of the smallest grammar problem. Ph.D. thesis, University of Rennes 1, France (2011)
24. Ganardi, M., Jez, A., Lohrey, M.: Balancing straight-line programs. In: 60th Annual Symposium on Foundations of Computer Science, FOCS '19, Baltimore, Maryland, USA, November 9-12, 2019 (2019)
25. Garey, M.R., Johnson, D.S.: Computers and Intractability. New York: Freeman (1979)
26. Garey, M.R., Johnson, D.S., Stockmeyer, L.: Some simplified NP-complete graph problems. Theoretical Computer Science **1**(3), 237–267 (1976)
27. Gascón, A., Godoy, G., Schmidt-Schauß, M.: Unification with singleton tree grammars. In: Rewriting Techniques and Applications, 20th International Conference, RTA 2009, Brasília, Brazil, June 29 - July 1, 2009, Proceedings, pp. 365–379 (2009)
28. Gascón, A., Lohrey, M., Maneth, S., Reh, C.P., Sieber, K.: Grammar-based compression of unranked trees. In: Computer Science - Theory and Applications - 13th International Computer Science Symposium in Russia, CSR 2018, Moscow, Russia, June 6-10, 2018, Proceedings, pp. 118–131 (2018)
29. Griggs, J.R., West, D.B.: Extremal values of the interval number of a graph. SIAM Journal on Matrix Analysis and Applications **1**(1), 1–7 (1980)

30. Gruber, H., Holzer, M., Wolfsteiner, S.: On minimal grammar problems for finite languages. In: Developments in Language Theory - 22nd International Conference, DLT 2018, Tokyo, Japan, September 10-14, 2018, Proceedings, pp. 342–353 (2018)
31. Haynes, T.W., Hedetniemi, S.T., Slater, P.J.: Fundamentals of Domination in Graphs, *Monographs and Textbooks in Pure and Applied Mathematics*, vol. 208. Marcel Dekker (1998)
32. Holzer, M., Wolfsteiner, S.: On the grammatical complexity of finite languages. In: Descriptional Complexity of Formal Systems - 20th IFIP WG 1.02 International Conference, DCFS 2018, Halifax, NS, Canada, July 25-27, 2018, Proceedings, pp. 151–162 (2018)
33. Hucke, D., Lohrey, M., Reh, C.P.: The smallest grammar problem revisited. In: String Processing and Information Retrieval - 23rd International Symposium, SPIRE 2016, Beppu, Japan, October 18-20, 2016, Proceedings, pp. 35–49 (2016)
34. Jez, A.: Recompression: A simple and powerful technique for word equations. J. ACM **63**(1), 4:1–4:51 (2016)
35. Kieffer, J.C., Yang, E.: Grammar-based codes: A new class of universal lossless source codes. IEEE Transactions on Information Theory **46**(3), 737–754 (2000)
36. Kieffer, J.C., Yang, E., Nelson, G.J., Cosman, P.C.: Universal lossless compression via multilevel pattern matching. IEEE Transactions on Information Theory **46**(4), 1227–1245 (2000)
37. Lanctôt, J.K., Li, M., Yang, E.: Estimating DNA sequence entropy. In: Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2000, January 9-11, 2000, San Francisco, CA, USA., pp. 409–418 (2000)
38. Larsson, N.J., Moffat, A.: Off-line dictionary-based compression. Proceedings of the IEEE **88**, 1722–1732 (2000)
39. Lehman, E.: Approximation algorithms for grammar-based data compression. Ph.D. thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology (2002)
40. Li, M., Vitányi, P.: An introduction to Kolmogorov complexity and its applications, 2nd edn. Springer (1997)
41. Lohrey, M.: Algorithmics on SLP-compressed strings: A survey. Groups, Complexity, Cryptology **4**(2), 241–299 (2012)
42. Lohrey, M.: The Compressed Word Problem for Groups, Springer Briefs in Mathematics edn. Springer (2014)
43. Lohrey, M., Maneth, S.: The complexity of tree automata and XPath on grammar-compressed trees. Theoretical Computer Science **363**(2), 196–210 (2006)
44. Lohrey, M., Maneth, S., Mennicke, R.: XML tree structure compression using RePair. Information Systems **38**(8), 1150–1167 (2013)
45. Lohrey, M., Maneth, S., Schmidt-Schau, M.: Parameter reduction and automata evaluation for grammar-compressed trees. Journal of Computer and System Sciences **78**(5), 1651–1669 (2012)
46. Maneth, S., Navarro, G.: Indexes and computation over compressed structured data (dagstuhl seminar 13232). Dagstuhl Reports **3**(6), 22–37 (2013). DOI 10.4230/DagRep.3.6.22. URL `https://doi.org/10.4230/DagRep.3.6.22`
47. Manlove, D.F.: On the algorithmic complexity of twelve covering and independence parameters of graphs. Discrete Applied Mathematics **91**(1-3), 155–175 (1999)
48. de Marcken, C.: Unsupervised language acquisition. Ph.D. thesis, Department of Electrical Engineering and Computer Science, MIT, USA (1996)
49. Nevill-Manning, C.G.: Inferring sequential structure. Ph.D. thesis, University of Waikato, NZ (1996)
50. Nevill-Manning, C.G., Witten, I.H.: Identifying hierarchical structure in sequences: A linear-time algorithm. Journal of Artificial Intelligence Research **7**, 67–82 (1997)
51. Nevill-Manning, C.G., Witten, I.H.: On-line and off-line heuristics for inferring hierarchies of repetitions in sequences. Proceedings of the IEEE **88**, 1745–1755 (2000)
52. Papadimitriou, C.H.: Computational Complexity. Addison-Wesley (1994)
53. Plandowski, W., Rytter, W.: Application of Lempel-Ziv encodings to the solution of words equations. In: Automata, Languages and Programming, 25th International Colloquium, ICALP 1998, Aalborg, Denmark, July 13-17, 1998, Proceedings, pp. 731–742 (1998)

54. Rytter, W.: Application of Lempel-Ziv factorization to the approximation of grammar-based compression. Theoretical Computer Science **302**(1-3), 211–222 (2003)
55. Shannon, C.E.: A theorem on coloring the lines of a network. Journal of Mathematics and Physics **28**, 148–151 (1949)
56. Skulrattanakulchai, S.: Δ-list vertex coloring in linear time. Information Processing Letters **98**(3), 101–106 (2006)
57. Storer, J.A.: NP-completeness results concerning data compression. Tech. Rep. 234, Dept. Electrical Engineering and Computer Science, Princeton University, USA (1977)
58. Storer, J.A., Szymanski, T.G.: Data compression via textual substitution. Journal of the ACM **29**(4), 928–951 (1982)
59. Thurber, E.G.: Efficient generation of minimal length addition chains. SIAM Journal on Computing **28**, 1247–1263 (1999)
60. Vizing, V.G.: The chromatic class of a multigraph. Kibernetika (Kiev) **1**(3), 29–39 (1965)
61. Welch, T.A.: A technique for high-performance data compression. IEEE Computer **17**(6), 8–19 (1984)
62. Yang, E., Kieffer, J.C.: Efficient universal lossless data compression algorithms based on a greedy sequential grammar transform - part one: Without context models. IEEE Transactions on Information Theory **46**(3), 755–777 (2000)
63. Ziv, J., Lempel, A.: Compression of individual sequences via variable-rate coding. IEEE Transactions on Information Theory **24**(5), 530–536 (1978)