

Grammatical Complexity of Finite Languages

DISSERTATION

submitted in partial fulfillment of the requirements for the degree of

Doktor der Technischen Wissenschaften

by

Simon Peter Wolfsteiner

Registration Number 00705422

to the Faculty of Informatics

at the TU Wien

Advisor: Associate Prof. Dr. Stefan Hetzl

The dissertation has been reviewed by:

Markus Holzer

Cezar Câmpeanu

Vienna, 3rd June, 2020

Simon Peter Wolfsteiner

Erklärung zur Verfassung der Arbeit

Simon Peter Wolfsteiner
Wiedner Hauptstraße 8–10
A-1040 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 3. Juni 2020

Simon Peter Wolfsteiner

Acknowledgements

First and foremost, I would like to express my sincerest gratitude to my advisor Stefan Hetzl who offered me a position in his research group and, moreover, guided me—with great patience—through this journey called PhD. I am also very grateful for the vast amount of time he has spent on proofreading and discussing my work. His valuable suggestions and comments definitely made me a better and more precise researcher and the thesis would not be in the current shape without him. Finally, I also want to thank Stefan for his understanding and giving me the possibility to work from Upper Austria from time to time during the time of my mother's illness.

Next, I want to thank Markus Holzer for giving me the opportunity to visit his research group in Giessen on numerous occasions. He has also taught me a lot regarding scientific writing and doing research in the field of formal language theory. In some sense, he has also been an unofficial second advisor of mine, since quite a few of the results in this thesis are based on papers that we wrote together. I also enjoyed our numerous discussions that went beyond the scope of science. Of course, I also want to thank all the other people at the Institut für Informatik in Giessen for their hospitality: Andreas, Bianca, Heinz, Martin, Matthias, Simon, and Susanne. Furthermore, I want to thank Hermann Gruber for a great collaboration together with Markus Holzer that culminated in a DLT paper.

A great thank you also goes to my master's thesis advisor Alex Leitsch who has encouraged me to continue with a PhD. On top of that, he has also played a vital role in shaping my understanding of doing science, and of science in general.

Special thanks also go to my office colleagues: Jan Bydžovský, Gabriel Ebner, Jannik Vierling, and Sebastian Zivota.

I am also very grateful to Sonja Rees who has been a huge help regarding the tedious bureaucratic stuff that accompanies scientific day-to-day routine.

Also, I want to thank Cezar Câmpeanu and, again, Markus Holzer for agreeing to act as reviewers for this thesis—a task that clearly takes a lot of time and effort. Their comments and suggestions were of great value and definitely lead to an improvement in the overall shape of the thesis.

As a PhD student it is vital that you take your mind off of scientific things from time to time. Therefore, I want to thank the following friends (in no particular order) for great

times during and beyond my time as a PhD student: Alex, André, Anela, Bene, Joachim, Jony, Laura, Lukas, Manuel, Matthias, Phipo, Simone, Steines, Surni, Thomas, Toby, Vali, and Viola.

Of course, I want to mention the latest and most wonderful addition to my life: Charlotte, thank you for being the most loving and lovable human being there is.

Last but not least, I want to thank my family for always believing in me and being there no matter how tough times have been. Thank you Adrian, Daniela, Fabia, Junior, Markus, and Sonja. My greatest thanks go to my late parents Leopold and Monika who, unfortunately, were not able to witness the completion of my PhD studies. I therefore dedicate this thesis to the both of you, since it surely would not have happened without the two of you.

To my parents Leopold and Monika

Kurzfassung

Die vorliegende Arbeit beschäftigt sich – aus den Blickwinkeln der Beschreibungskomplexität sowie der Komplexitätstheorie – mit endlichen formalen Sprachen, welche von verschiedenen Arten von kontextfreien Grammatiken erzeugt werden. Insbesondere werden verschiedene Aspekte der exakten Komplexität und der Überdeckungskomplexität von endlichen Sprachen untersucht: was ist die minimale Anzahl an Produktionen die eine kontextfreie Grammatik benötigt um eine gegebene endliche Sprache genau zu erzeugen bzw. zu überdecken? Aus der komplexitätstheoretischen Perspektive werden vor allem folgende Varianten des Smallest Grammar Problem betrachtet: welche symbolische Komplexität bzw. welche Anzahl an Produktionen hat eine minimale kontextfreie Grammatik, welche eine gegebene endliche Sprache erzeugt? Wir werden beweisen, dass weder die minimale symbolische Komplexität noch die minimale Anzahl an Produktionen einer kontextfreien Grammatik, welche eine gegebene endliche Sprache erzeugt, bis auf einen gewissen Faktor approximiert werden kann, solange $P = NP$ nicht gilt. Zusätzlich werden wir auch Probleme unter der Annahme der sogenannten Exponentialzeithypothese untersuchen. Ferner betrachten wir die exakte Komplexität sowie die Überdeckungskomplexität von verschiedenen Sprachoperationen, angewandt auf endliche Sprachen, aus dem Blickwinkel der Beschreibungskomplexität. Zu guter Letzt werden wir verschiedene Komplexitätsmaßtypen basierend auf der Anzahl der Produktionen vergleichen, sodass sich jeweils eine auf Grammatik- sowie eine auf Maßtypen basierende Klassifikation ergibt.

Abstract

In this thesis, we will investigate finite languages which are generated by different types of context-free grammars from the points of view of descriptive and computational complexity. Specifically, we will study several aspects of the exact and cover complexity of finite languages, i.e., the minimal number of productions a grammar needs in order to generate and cover, respectively, a given finite language. From the point of view of computational complexity, we will consider the following variants of the smallest grammar problem: what is the size or number of productions of a minimal grammar that generates a given finite language? It will be shown that both the minimal size and the minimal number of productions needed in order to generate a given finite language cannot be approximated within certain factors, unless $P = NP$. In addition, we will also consider problems under the assumption of the so-called Exponential Time Hypothesis. Moreover, from a mere descriptive complexity point of view, we will investigate the exact and cover complexity of different language operations on finite languages. Finally, we will give a relative succinctness classification of several grammar as well as several complexity measure types based on the number of productions.

Contents

Kurzfassung	x
Abstract	xi
Contents	xiii
1 Introduction	1
2 Preliminaries	11
2.1 Formal Language Theory	11
2.2 Automata	18
2.3 Computational Complexity	21
3 Complexity Measures	27
3.1 Exact and Cover Complexity	27
3.2 Unboundedness of Cover Complexity Measures	30
3.3 Unboundedness of Grammatical Cover Complexity	34
3.4 Computing Cover Complexity from Exact Complexity	39
4 Bounds on Production Complexity	45
4.1 Basic Bounds on Production Complexity	45
4.2 Lower Bounds on Exact Production Complexity	55
4.3 Lower Bounds on Cover Production Complexity	62
5 Relating Finite and Infinite Complexity Measures	69
5.1 Infinite Complexity Measures	71
5.2 Relating Grammar Types	75
5.3 Relating Measure Types	96
6 Bounds on Language Operations	121
6.1 Intersection	123
6.2 Union	125
6.3 Concatenation	131
	xiii

7 Complexity of The Smallest Grammar Problem for Finite Languages	145
7.1 Inapproximability of the Minimal Number of Productions	146
7.2 The Smallest Grammar Problem for Finite Languages	161
7.3 The Uniform-Length Universality Problem and the ETH	165
8 Conclusion	169
List of Figures	175
Index of Notation and Abbreviations	177
Bibliography	181

CHAPTER

1

Introduction



OMPLEXITY is, in general, a highly ambiguous noun that carries—especially in the context of science—several different meanings. This particular issue is also emphasised by physicist Neil F. Johnson in his book “Simply Complexity: A Clear Guide to Complexity Theory”:

“Well, unfortunately, Complexity is not easy to define. Worse still, it can mean different things to different people. Even among scientists, there is no unique definition of Complexity.” [Joh09, p. 24]

Even if we restrict the meaning of complexity only to notions within the realm of theoretical computer science, we are still confronted with several different definitions of complexity. Consider, for instance, the areas of computational and descriptonal complexity. While, in the former, one deals with the asymptotic worst-case time and space complexity of algorithms that solve a specific given decision problem, in the latter, one deals with complexity in the sense of the smallest description—by means of a descriptonal system such as Turing machines, automata, grammars, etc.—of a given formal language or string. As the title of this thesis already suggests, we are concentrating on questions regarding the complexity of context-free grammars that approximate (i.e., generate, cover, etc.) a given finite language in various finite ways. Due to the fact that we will be dealing with problems regarding context-free grammars from both a descriptonal and computational complexity point of view, the ambiguity of the term (grammatical) complexity will also shine through in this thesis.

Since the advent of computer science, the field of descriptonal complexity has been the subject of intensive research. Besides investigating various different complexity measures for descriptonal systems, this broad field is also concerned with the relative succinctness of these systems, i.e., with the question of which systems allow for a more concise

description of a given formal language than others.¹ A prominent and well-studied example of a problem in descriptonal complexity is the so-called *Smallest Grammar Problem*. This elegant problem asks for the smallest (in terms of grammar size) context-free grammar that generates exactly a single given word, and has ties to several different lines of research such as approximation algorithms, computational complexity theory, data compression, and Kolmogorov complexity [CLL⁺05]. In [CLL⁺05], it was shown that the smallest grammar problem has an approximation ratio of at least

$$\frac{8569}{8568},$$

unless $P = NP$, and, moreover, that its decision version is NP-complete. Its relation to grammar-based compression [ZL78, NM96, NMW97, KY00, KYNC00, YK00] is reflected in the fact that instead of storing a long string, one can also store a small grammar that generates this string. Afterwards, one can reconstruct the string from that grammar when needed [CLL⁺05]. The field of grammar-based compression has a similar relation to descriptonal complexity as the field of algorithm design has to complexity theory in the following sense: while words that can be compressed by grammars give rise to grammar-based compression algorithms for these words, tractable computational problems give rise to efficient algorithms that solve this problem. The size of the smallest context-free grammar that generates a given word can be viewed as a natural, but computable version of *Kolmogorov complexity*. For a given word w , the Kolmogorov complexity of w is the description size of the smallest Turing machine (or computer program) that outputs w [CLL⁺05, IV08].

As already mentioned, we are mainly interested in the *grammatical complexity* of a language, i.e., in the minimal (w.r.t. a specific complexity measure) context-free grammar that approximates a given formal language. Depending on the type of grammar and the notion of complexity, one obtains a variety of different grammatical complexity measures. Historically, the systematic study of grammatical complexity of context-free languages can be traced back to [Gru67], where, among other things, it was shown that context-free definability with n nonterminals forms a strict hierarchy. This line of research was continued and surveyed in [Gru69, Gru71, Gru72] and [Gru76], respectively, where, among others, the number of productions of a context-free grammar was considered as complexity measure. The concept of orthogonality of two complexity measures² with respect to the class of context-free languages was introduced and studied in [Geo96]. There, among other results, it was shown that the exact production complexity measure is orthogonal to the exact nonterminal complexity measure, but not vice versa.

In our investigations, we are focusing on the complexities of different types of context-free grammars that describe *finite languages*. The complexity measures that we are mostly

¹For an introductory survey of the main aspects and results regarding the field of descriptonal complexity, see, e.g., [HK11].

²For two context-free grammar-based complexity measures μ_1 and μ_2 , we say that μ_1 is *orthogonal* to μ_2 if there is some n_0 such that for every $n \geq n_0$, there is some k_n such that for every $m \geq k_n$, there is some context-free language L such that $\mu_1(L) = m$ and $\mu_2(L) = n$.

dealing with are the *minimal* number of productions of a grammar G that

- (i) generates a finite language, i.e., $L(G) = L$ (*exact complexity*), and
- (ii) covers a finite language, i.e., $L(G) \supseteq L$ and $L(G)$ is finite (*cover complexity*).

The theory of the grammatical complexity of finite languages in terms of the number of productions was initiated in [BMCIW81], where a relative succinctness classification for various kinds of context-free grammars was given. Further results along these lines can be found in, e.g., [Buc81, AER83, BMCI83, Tuz87, DH12b, Das17]. Every finite language L can obviously be generated by a *trivial* grammar, i.e., a grammar with a single nonterminal S whose productions are given by the set

$$\{S \rightarrow w \mid w \in L\}.$$

Therefore, we are interested in the question of whether a given finite language L can be *compressed* or *cover-compressed*, that is, whether there exists a context-free grammar G with $L(G) = L$ (i.e., G generates L) or $L(G) \supseteq L$ (i.e., G covers L) and $L(G)$ is finite, respectively, such that G has fewer productions than there are words in L . Some finite languages were already shown to be incompressible by certain types of grammars in [BMCIW81, Buc81, BMCI83].

Our interest in the cover complexity of a finite language L —i.e., the minimal number of productions of a grammar G such that $L(G)$ is finite and $L(G) \supseteq L$ —is primarily motivated by applications in proof theory. As was shown in [Het12], there is an intimate relationship between a certain class of formal proofs (i.e., proofs with Π_1 -cuts) in first-order predicate logic and a certain class of grammars (i.e., totally rigid acyclic tree grammars). In particular, the number of production rules in the grammar characterises the number of certain inference rules in the proof. This relationship was exploited for a number of results in proof theory and automated deduction. In [HLW12], a method based on the aforementioned relationship between proofs and grammars was presented that allows the compression of a proof in first-order logic by the introduction of a lemma. This method was then extended to the introduction of an arbitrary number of lemmas, and to first-order logic with equality in [HLRW14, HLR⁺14, EHL⁺19]. Furthermore, an application to inductive theorem proving was presented in [EH15b]. By constructing a sequence of cover-incompressible languages, the authors of [EH15a, EH18] employed the connection between proofs and grammars in order to obtain a lower bound on the size of a certain class of proofs. A practically efficient algorithm that cover-compresses a finite tree language by a special type of tree grammar was devised in [EEH17]. Moreover, in [EEH18], it was shown that the minimal cover problem for acyclic regular grammars with a fixed bound on the number of nonterminals is NP-complete. The minimal cover problem is defined as follows: given a finite language L and a non-negative integer k , is there an acyclic regular grammar G such that G has at most k productions and satisfies $L(G) \supseteq L$? However, the computational complexity of this problem for an arbitrary number of nonterminals is still open.

Note that the condition $L(G) \supseteq L$ used in the definition of grammatical cover complexity of a finite language L is similar to (but different from) the one imposed on *cover automata* [CSY99, CSY01]. There, an automaton \mathcal{A} is sought such that $L(\mathcal{A}) \supseteq L$, but, in addition, it is required that $L(\mathcal{A}) \setminus L$ consists solely of words longer than any word in L . The importance of cover automata for the representation of finite languages is rooted in the fact that a minimal deterministic finite cover automaton (DFCA) for a finite language L usually has a smaller size than a minimal deterministic finite automaton (DFA) that accepts the same language L [CSY99]. In [CSY01], the authors presented an efficient algorithm that, for a given finite language L (which is either given as a DFA or a DFCA), constructs a minimal cover automaton for L . On top of that, in the same paper, algorithms for the Boolean operations intersection, union, symmetric difference, and difference on DFCA, that is, on the finite languages they represent, were given. In [PSY01, CPY02], an algorithm for the construction of a minimal DFCA was presented which runs in time and space $\mathcal{O}(n^2)$ and thus outperforms the $\mathcal{O}(n^4)$ -time and -space algorithm of [CSY01], where n is the number of states of the given DFA. The $\mathcal{O}(n \cdot \log n)$ -time and $\mathcal{O}(n)$ -space algorithm devised in [Kör03a, Kör03b] outperforms both of these previous algorithms. All of the aforementioned algorithms for constructing minimal DFCA for a given finite language are not incremental, but in [CPS06a, CPS06b] this gap was closed by giving the first incremental algorithm for this endeavour. A comparison on a specific finite language between implementations of the incremental algorithm and the algorithm of [Kör03a, Kör03b] even showed that the former algorithm constructs automata with less states and needs both less memory and less time than the latter. The aforesaid algorithms for the construction of a minimal cover automaton from a given DFA are based on the so-called *similarity relation*.³ Since the similarity relation is not an equivalence relation, the minimal DFCA for a finite language is usually not unique [CP03b]. This raised the question of how many distinct DFCA a minimisation algorithm can yield which is based on the notion of similarity relation. The answer to this question for k -ary alphabets with $k \geq 2$ is bounded above by

$$\frac{k_0!}{(2 \cdot k_0 - n + 1)!},$$

where n is the number of states in the given minimal DFA and $k_0 = \left\lceil \frac{4 \cdot n - 9 + \sqrt{8 \cdot n + 1}}{8} \right\rceil$. For unary alphabets the bound is $n - 1$, and both of these bounds are tight, as was shown in [CP03a, CP03b]. In [CKP05, CP05], a lower bound on the maximum state complexity of deterministic finite cover automata was obtained from nondeterministic finite automata with n states defined over binary alphabets. Additionally, it was shown that the result of transforming an n -state NFA that accepts a finite language over a binary alphabet into an equivalent minimal DFCA has at least $2^{\lceil \frac{n}{2} \rceil - 2}$ fewer states than the number of states in the minimal DFA that is obtained by a transformation from the NFA. As was shown in [Câm14, Câm15], nondeterministic finite cover automata allow a more compact representation of

³Let $w, v \in \Sigma^*$. Then the similarity relation \sim_L on words is defined as: $w \sim_L v$ if, for all $u \in \Sigma^*$, it holds that $wu \in L$ iff $vu \in L$ [CP03b].

finite languages than both NFAs and DFAs. Moreover, in [Câm14, Câm15], it was also shown that minimising NFAs can be as hard as minimising NFAs for regular languages. In [GHJ15, GHJ17], the authors studied how to adapt well-established lower bound techniques in order to be applicable to NFAs. That particular paper also contains, alongside an investigation of the average size of finite cover automata, an investigation of the trade-off between DFAs and NFAs as well as between finite cover automata and ordinary finite automata. Further results related to finite cover automata can be found in, e.g., [CSY00, CGH05, CMR11, JM11, Ipa12, CMR16].

In this thesis, we will explore and gain new insights into the very nature of finite languages from both a descriptive and computational complexity theoretic point of view. However, the focus will be on the descriptive complexity side of things, where we will address several problems that have received only little attention so far. We will investigate the notion of cover complexity of finite languages on three different levels of generality. In particular, we will consider the notion of cover complexity from an abstract point of view for arbitrary complexity measures and characterise the situations in which these measures collapse to a bounded measure. In a less general sense, we will also consider the cover complexity of a finite language L as the minimal number of productions a grammar needs in order to cover L with a finite language. More precisely, we will show that a cover complexity measure is unbounded if it is induced by a certain restricted class of context-free grammars in which the right-hand side of each production contains a bounded number of nonterminals. An analogous result will also be shown for strict linear and strict regular grammars. For these restricted types of context-free grammars, we obtain that the cover complexity of a finite language L can be reduced to the minimum of the exact complexities of a finite number of supersets L' of L . Furthermore, we will investigate the grammatical cover-complexity of the classic language operations intersection, union, and concatenation on finite languages for strict regular, strict linear, regular, and linear grammars. By generalising the cover-incompressible sequence of finite languages constructed in [EH15a, EH18], we will obtain a new cover-incompressible sequence that allows us to show a lower bound on the cover-complexity of union with respect to a fixed alphabet.

Similarly to the relative succinctness classification with respect to the nonterminal complexity in [DP89] (which is based on four different categories of complexity gaps), we will give, for strict regular (SREG), strict linear (SLIN), regular (REG), linear (LIN), and context-free (CF) grammars, a relative succinctness classification with respect to the production complexity of finite languages. In particular, for all grammar types $X \in \{\text{SREG}, \text{SLIN}, \text{REG}, \text{LIN}, \text{CF}\}$, we will consider the complexity measures X_c , X_{cc} , and X_{sc} , that is, with respect to the grammar type X , the exact, cover, and scattered complexity, respectively. The X scattered complexity of a finite language L is defined as the number of productions of a minimal grammar G of type X such that $L(G) \geq L$ and $L(G)$ is finite, where \geq refers to the scattered subword relation. Additionally, we will also consider corresponding infinite complexity measures, which are inspired by the earlier mentioned condition imposed on cover automata. The infinite measures X_{c_∞} , X_{cc_∞} , and X_{sc_∞} are

obtained from X_c , X_{cc} , and X_{sc} , respectively, by dropping the requirement that, for a minimal grammar G of type X , $L(G)$ is finite, and, moreover, replacing, for $\geq \in \{=, \supseteq, \geq\}$, the condition $L(G) \geq L$ by $L(G) \cap \Sigma^{\leq \ell} \geq L$, where the non-negative integer ℓ refers to the length of a longest word in L . All of these complexity measures will then be compared with each other according to the taxonomy introduced in [DP89] by

- (i) fixing the grammar type and varying the measure type,⁴ and
- (ii) fixing the measure type and varying the grammar type.

As a byproduct of this succinctness classification, we will also show that there are languages that need an exponential number of productions in order to be generated by a grammar of a certain type. More precisely, we will show that the language of even length palindromes

$$P_n = \{ w\$w^R \mid w \in \{a, b\}^{\leq n} \}$$

needs at least 2^n many regular productions. This is done by showing that the sublanguage of fixed-length palindromes

$$P'_n = \{ w\$w^R \mid w \in \{a, b\}^n \}$$

is regular incompressible. Moreover, it will also be shown that the language of triples

$$T_n = \{ w\$w\#w \mid w \in \{a, b\}^n \}$$

is regular, linear, and context-free incompressible.

From the point of view of computational complexity theory and approximation algorithms, we will consider the following variant of the smallest grammar problem: what is the grammatical complexity of a smallest grammar that generates a given *finite language*? In this regard, it will be shown that, for fixed alphabets consisting of at least 5 elements, given an arbitrary context-free grammar with p productions that generates a finite language L , the minimal number of productions necessary to generate L cannot be approximated within a factor of

$$o(p^{1/6}),$$

unless $P = NP$. Since the size of a context-free grammar depends on the number of its productions, the above result also implies that, given an arbitrary context-free grammar with size s that generates a finite language L , the minimal size necessary to generate L with a context-free grammar cannot be approximated within a factor of

$$o(s^{1/7}),$$

unless $P = NP$. The latter result is related to the aforementioned result of [CLL⁺05] that any approximation algorithm for the classic version of the smallest grammar problem

⁴The considered measure types are: exact (c), cover (cc), infinite exact (c_∞), infinite cover (cc_∞), and infinite scattered (sc_∞) complexity.

cannot achieve a better ratio than about 1.00012. The proofs of our inapproximability results rely on estimates on the exact complexity of union and concatenation of finite languages. Therefore, we will study the exact complexity of the operations union and concatenation of finite languages. This also complements the results obtained in [DH12b, Das17] regarding these operations on infinite languages. Inspired by investigations of problems on finite automata under the assumption of the so-called *Exponential Time Hypothesis* in [FK17], we will study the uniform-length universality problem under the assumption of this hypothesis. Roughly speaking, the Exponential Time Hypothesis is a conjecture which expresses that the 3-SAT problem cannot be decided in deterministic subexponential time and was introduced in the paper [IP99] (see also [CFK⁺15]). The previously mentioned uniform-length universality problem asks, for a given context-free grammar $G = (N, \Sigma, P, S)$ and an integer $\ell \geq 0$, whether $L(G)$ generates all words of length ℓ over Σ , that is, whether $L(G) = \Sigma^\ell$. In particular, we will show that, under the assumption of the Exponential Time Hypothesis, there is no deterministic algorithm that decides uniform-length universality in time⁵

$$\mathcal{O}^*\left(2^{o(p^{1/4})}\right),$$

where p is the number of productions of the given grammar. Additionally, we will also show that under the assumption of the Exponential Time Hypothesis, there is no deterministic algorithm that decides the uniform-length universality problem in time

$$\mathcal{O}^*\left(2^{o(s^{1/4})}\right),$$

where s is the size of the given grammar.

Structure of the Thesis. This thesis is organised as follows:

- Subsequent to this introductory chapter, in Chapter 2, we will introduce the basic notions of formal language theory, context-free grammars, automata, and computational complexity theory which are relevant to this thesis.
- In Chapter 3, we will consider the notions of exact and cover complexity of finite languages from both an abstract and a grammatical point of view. In particular, we will give a characterisation of the situations in which arbitrary cover complexity measures collapse to a bounded complexity measure. Moreover, for a restricted class of context-free grammars as well as for strict regular and strict linear grammars, it will be shown that the corresponding cover complexity measures for finite languages are unbounded. Finally, for these kinds of context-free grammars, it will be shown that the cover complexity of a finite language L can be computed from the exact complexities of a finite number of finite languages L' with $L' \supseteq L$.

⁵Note that the \mathcal{O}^* -notation suppresses polynomial factors measured in the input length.

- Chapter 4 is devoted to proving several upper and lower bounds on various production complexity measures for finite languages. For some finite languages, it will be shown that they are incompressible with respect to certain complexity measures. In particular, we will construct a regular cover-incompressible sequence of finite languages that generalises a previously known one from [EH15a, EH18].
- A relative succinctness classification of several different complexity measures for finite languages will be given in Chapter 5. In particular, we will consider several different production complexity measures for finite languages with respect to different interpretations of approximation (i.e., equivalence, cover, and scattered cover), where the underlying grammar either generates a finite or an infinite language. If the underlying grammar generates an infinite language, then the intersection with all words up to a certain length has to be considered. These measures will then be related according to a group of relations that are inspired by the taxonomy with respect to nonterminal complexity which was introduced in [DP89].
- Chapter 6 is dedicated to proving, with respect to (strict) regular, (strict) linear, and context-free grammars, upper and lower bounds on both the exact and the cover complexity of applying the classic language operations intersection, union, and concatenation.
- The penultimate Chapter 7 deals with problems regarding context-free grammars that belong to the realms of computational complexity theory and approximation algorithms. In particular, we will show that, for fixed alphabets of cardinality at least 5, given an arbitrary context-free grammar G with p productions that generates a finite language L , the minimal number of productions necessary to generate L with a context-free grammar cannot be approximated within a factor of $o(p^{1/6})$, unless $P = NP$. In addition, we will show that, given an arbitrary context-free grammar G with size s that generates a finite language L , the minimal size of a context-free grammar that generates L cannot be approximated within a factor of $o(s^{1/7})$, unless $P = NP$. Furthermore, we will also show that, under the assumption of the Exponential Time Hypothesis, there is no algorithm that decides the uniform-length universality problem in time $\mathcal{O}^*\left(2^{o(p^{1/4})}\right)$, where p is the number of productions of the given grammar. We will also show that, under the assumption of the Exponential Time Hypothesis the uniform-length universality problem cannot be decided by an algorithm that runs in time $\mathcal{O}^*\left(2^{o(s^{1/4})}\right)$, where s is the size of the given grammar.
- This thesis will come to a conclusion in the final Chapter 8.

Publications. This thesis is partially based on the following publications:

- [GHW18] Hermann Gruber, Markus Holzer, and Simon Wolfsteiner. On Minimal Grammar Problems for Finite Languages. In Mizuho Hoshi and Shinnosuke Seki, editors, *Proceedings of the 22nd International Conference on Developments in Language Theory (DLT 2018)*, volume 11088 of *Lecture Notes in Computer Science*, pages 342–353, Cham, 2018. Springer.
- [HW18a] Stefan Hetzl and Simon Wolfsteiner. Cover Complexity of Finite Languages. In Stavros Konstantinidis and Giovanni Pighizzini, editors, *Proceedings of the 20th IFIP WG 1.02 International Conference on Descriptive Complexity of Formal Systems (DCFS 2018)*, volume 10952 of *Lecture Notes in Computer Science*, pages 139–150, Cham, 2018. Springer.
- [HW18b] Markus Holzer and Simon Wolfsteiner. On the Grammatical Complexity of Finite Languages. In Stavros Konstantinidis and Giovanni Pighizzini, editors, *Proceedings of the 20th IFIP WG 1.02 International Conference on Descriptive Complexity of Formal Systems (DCFS 2018)*, volume 10952 of *Lecture Notes in Computer Science*, pages 151–162, Cham, 2018. Springer.
- [HW19] Stefan Hetzl and Simon Wolfsteiner. On the cover complexity of finite languages. *Theoretical Computer Science*, 798:109–125, 2019.

The following two publications also came into being during the course of the PhD, but are beyond the scope of this thesis:

- [EHR⁺16] Gabriel Ebner, Stefan Hetzl, Giselle Reis, Martin Riener, Simon Wolfsteiner, and Sebastian Zivota. System Description: GAP2 2.0. In Nicola Olivetti and Ashish Tiwari, editors, *Proceedings of the 8th International Joint Conference on Automated Reasoning (IJCAR 2016)*, volume 9706 of *Lecture Notes in Computer Science*, pages 293–301, Cham, 2016. Springer.
- [CLRW17] David Cerna, Alexander Leitsch, Giselle Reis, and Simon Wolfsteiner. Ceres in intuitionistic logic. *Annals of Pure and Applied Logic*, 168(10):1783–1836, 2017.

CHAPTER 2

Preliminaries



VEN though we assume that the reader is familiar with the basic notions of formal language theory, context-free grammars, automata, and computational complexity theory, we will—in order to fix notation and terminology—introduce the basic notions of these areas which are relevant to this thesis in this chapter. In Section 2.1, we will introduce the relevant notions from formal language theory and context-free grammars. Section 2.2 contains the definitions of both finite automata and finite cover automata. Finally, in Section 2.3, we will introduce the basic notions of propositional logic, computational complexity theory, and approximation algorithms.

2.1 Formal Language Theory

In this section, we will introduce the basic notions of formal language theory and context-free grammars.

All definitions in this section are based on [Woo87, HMU01, Sha08, HW18a, HW18b, GHW18, HW19].

2.1.1 Sets

We denote the number of elements or *cardinality* of a finite set S by $|S|$. The *empty set* is denoted by \emptyset . For two sets A and B , we write $A \cap B$, $A \cup B$, $A \uplus B$, $A \times B$, and $A \setminus B$, for the *intersection*, *union*, *disjoint union*, *Cartesian product*, and *set difference*, respectively, of A and B . Finally, $\mathcal{P}(A)$ and $\mathcal{P}_{\text{fin}}(A)$ denote the *power set* (i.e., the set of all subsets of A) and the *set of all finite subsets of A* , respectively.

Some special sets of numbers that we consider include: $\mathbb{N} = \{0, 1, 2, \dots\}$, the set of *natural numbers*, $\mathbb{Z} = \{\dots, -2, -1, 0, 1, 2, \dots\}$, the set of *integers*, and \mathbb{R} , the set of *real numbers*. By \mathbb{R}^+ , we denote the set of all *non-negative* real numbers.

Let S be a set. Then we write id_S for the *identity function* on S , that is,

$$id_S(x) = x,$$

for each element $x \in S$.

2.1.2 Alphabets, Words, and Languages

An *alphabet*, frequently denoted by Σ , is a non-empty set of symbols. The symbols of an alphabet are also referred to as *letters*. Throughout this thesis, if not stated otherwise, alphabets are finite sets. An alphabet Σ is called *unary*, *binary*, and *n -ary*, for $n \geq 3$, if $|\Sigma| = 1$, $|\Sigma| = 2$, and $|\Sigma| = n$, respectively.

A *word* (or *string*) is a finite sequence of symbols chosen from some alphabet. The *empty word* is denoted by ε . Let

$$w = a_1 a_2 \dots a_n$$

be a word. Then the natural numbers $1, 2, \dots, n$ are called the *positions* of w . Furthermore, we say that a sequence

$$a_i a_{i+1} \dots a_j,$$

where $1 \leq i \leq j \leq n$, *occupies the positions i through j within w* . The *length* of a word w , i.e., the number of occurrences of symbols in w , written as $|w|$, is defined inductively as follows: the length of the empty word ε is 0, i.e., $|\varepsilon| = 0$, and if $w = aw'$, where a is a single symbol and w' is a word, then $|w| = |w'| + 1$. If a is a symbol and w is a word, then $|w|_a$ denotes the number of occurrences of the symbol a in w . For an alphabet Σ , the set of all words of length $k \geq 0$, each of whose symbols is in Σ , is denoted by Σ^k . Furthermore, we define

$$\Sigma^{\leq k} = \bigcup_{i=0}^k \Sigma^i, \quad \Sigma^+ = \bigcup_{i \geq 1} \Sigma^i, \quad \text{and} \quad \Sigma^* = \Sigma^+ \cup \{\varepsilon\} = \bigcup_{i \geq 0} \Sigma^i.$$

In particular, Σ^+ is the set of all non-empty and Σ^* the set of all words over the alphabet Σ . The language Σ^* is also called the *universal language*.

Remark. Typically, symbols are denoted by digits or lower-case letters at the beginning of the alphabet, e.g., $a, b, c, a_1, b_1, a', b', \dots$, and words by lower-case letters near the end of the alphabet, e.g., $u, v, w, x, y, z, u_1, v_1, u', v'$, etc.

We say that a word u is a *subword* of a word w , denoted by

$$u \sqsubseteq w,$$

if there exist words v_1 and v_2 such that

$$w = v_1 u v_2.$$

If u is a subword of w , we say that w *contains* u and that u is *contained in* w . A word u is called a *prefix* of w if there exists a word v such that

$$w = uv.$$

A prefix is called *proper* if $v \neq \varepsilon$ and *non-trivial* if $u \neq \varepsilon$. Similarly, we say that u is a *suffix* of w if there exists a v such that

$$w = vu.$$

A suffix is called *proper* if $v \neq \varepsilon$ and *non-trivial* if $u \neq \varepsilon$.

One of the most fundamental operations on words is *concatenation*. Let w_1 and w_2 be words. Then

$$w_1 w_2$$

denotes the concatenation (or product) of w_1 and w_2 . For any word w , we have that

$$w\varepsilon = \varepsilon w = w,$$

i.e., ε is the *identity for concatenation*. Moreover, concatenation is associative, but not commutative. In general, concatenation is treated notationally like multiplication, for instance,

$$w^n$$

denotes the string

$$ww \cdots w \quad (n \text{ times}).$$

If

$$w = a_1 a_2 \cdots a_n$$

is a word, then by

$$w^R,$$

we denote the *reversal* of the word w , that is,

$$w^R = a_n a_{n-1} \cdots a_2 a_1.$$

Note that

$$\varepsilon^R = \varepsilon \quad \text{and} \quad (w_1 w_2)^R = w_2^R w_1^R.$$

A word w is called a *palindrome* if $w = w^R$.

Let Σ be an alphabet. If L is a (finite or infinite) subset of Σ^* , then L is called a *language over* Σ . Particularly, an element L of $\mathcal{P}_{\text{fin}}(\Sigma)$ is called a *finite language over* Σ . For every finite language L over Σ , there is some $k \geq 0$ such that $L \subseteq \Sigma^{\leq k}$. If all words in a language L over Σ have the same length, then L is called a *uniform language over* Σ . Moreover, a language L over an alphabet Σ with $|\Sigma| = 1$ is called a *unary language over* Σ . We omit the term “over Σ ” if the alphabet of the language is clear from the context.

Remark. If not stated otherwise, we always assume that if L is a language over some alphabet Σ , then Σ is the smallest alphabet with $L \subseteq \Sigma^*$, i.e., Σ does not contain letters that are not occurring in L .

Example 2.1.1. Some examples of languages are

$$\begin{aligned} C &= \{ww \mid w \in \{0,1\}^*\} && \text{the copy language over } \{0,1\}, \\ E &= \{w \in \{0,1\}^* \mid |w|_0 = |w|_1\} && \text{the set of words with an equal number of each letter,} \\ P &= \{w \in \{0,1\}^* \mid w = w^R\} && \text{the language of palindromes over } \{0,1\}. \end{aligned}$$

Let L_1 and L_2 be languages over some alphabet Σ . Then we define the *concatenation (or product)* of L_1 and L_2 , denoted by L_1L_2 , as follows:

$$L_1L_2 = \{w_1w_2 \mid w_1 \in L_1, w_2 \in L_2\}.$$

Moreover, if $L \subseteq \Sigma^*$, then the *reversal* of L , denoted by L^R , is defined as

$$L^R = \{w^R \mid w \in L\}.$$

For languages $L_1, L_2 \subseteq \Sigma^*$, the *right* and *left quotient* of L_1 with L_2 is defined as

$$\begin{aligned} L_1L_2^{-1} &= \{v \in \Sigma^* \mid \text{there is some } w \in L_2 \text{ such that } vw \in L_1\} \text{ and} \\ L_1^{-1}L_2 &= \{v \in \Sigma^* \mid \text{there is some } w \in L_1 \text{ such that } wv \in L_2\}, \end{aligned}$$

respectively.

Remark. We also use the notation

$$L_1 \cdot L_2$$

for the concatenation of the languages L_1 and L_2 . Moreover, we omit braces if L_1 or L_2 is a singleton, i.e., for $w_1, w_2 \in \Sigma^*$, we write

$$L_1w_2 \quad \text{and} \quad w_1L_2$$

instead of

$$L_1\{w_2\} \quad \text{and} \quad \{w_1\}L_2,$$

respectively. Similarly, we write

$$L_1w_2^{-1} \quad \text{and} \quad w_1^{-1}L_2$$

instead of

$$L_1\{w_2\}^{-1} \quad \text{and} \quad \{w_1\}^{-1}L_2,$$

respectively.

2.1.3 Context-Free Grammars

A *context-free* (CF) grammar is a quadruple $G = (N, \Sigma, P, S)$, where

- N and Σ are disjoint finite sets of *nonterminals* and *terminals*, respectively,
- P is a finite set of *production rules* (or *productions* for short) of the form

$$A \rightarrow \alpha,$$

where $A \in N$ and $\alpha \in (N \cup \Sigma)^*$, and

- $S \in N$ is the *start symbol* (or *axiom*).

Henceforth, we will often abbreviate the term “context-free grammar” as CFG.

Let $G = (N, \Sigma, P, S)$ be a CF-grammar and let $A \in N$. Then a production with A on its left-hand side is called an *A-production*. We write P_A for the subset of P consisting of all A -productions, that is,

$$P_A = \{A \rightarrow \alpha \mid A \rightarrow \alpha \in P\}.$$

For $N' \subseteq N$, we define

$$P_{N'} = \bigcup_{A \in N'} P_A.$$

A production of the form

$$A \rightarrow \varepsilon$$

is called *ε -production*. If P contains no ε -productions, then G is called *ε -free*. For $A, B \in N$, a production of the form

$$A \rightarrow B$$

is called *unit production*. An element of the set $(N \cup \Sigma)^*$ is called a *word over $N \cup \Sigma$* . The *length of a word α over $N \cup \Sigma$* , written as $|\alpha|$, is the number of occurrences of terminal and nonterminal symbols in α .

Remark. We typically denote nonterminals by upper-case Latin letters A, B, A_1, B_1 , etc., and words over $N \cup \Sigma$ by lower-case Greek letters $\alpha, \beta, \alpha_1, \beta_1$, etc. Moreover, we often write the productions of a context-free grammar by listing each nonterminal A once and then listing all the right-hand sides of A -productions separated by vertical bars. That is, instead of

$$A \rightarrow \alpha_1, A \rightarrow \alpha_2, \dots, A \rightarrow \alpha_n,$$

we write

$$A \rightarrow \alpha_1 \mid \alpha_2 \mid \dots \mid \alpha_n.$$

A context-free grammar is called *trivial* if it only contains a single nonterminal and the right-hand side of each production consists of terminals only.

Definition 2.1.2 (Trivial Grammar). Let $G = (N, \Sigma, P, S)$ be a context-free grammar. A production of the form

$$S \rightarrow w,$$

for $w \in \Sigma^*$, is called *trivial*; all other productions are called *non-trivial*. We define $G_t = (N, \Sigma, P_t, S)$, where P_t is the set of trivial productions of G , i.e.,

$$P_t = \{ S \rightarrow w \in P \mid w \in \Sigma^* \}.$$

If $G = G_t$, then G is called *trivial* and *non-trivial* otherwise.

We will also consider various restricted variants of context-free grammars. These restrictions mainly arise from the way in which the right-hand sides of productions are restricted.

Definition 2.1.3 (CFG Restrictions). A context-free grammar $G = (N, \Sigma, P, S)$ is called

- *linear context-free* (LIN) if all productions in G are of the form

$$A \rightarrow \alpha \text{ with } \alpha \in \Sigma^*(N \cup \{\varepsilon\})\Sigma^*;$$

- *right-linear* if all productions in G are of the form

$$A \rightarrow \alpha \text{ with } \alpha \in \Sigma^*(N \cup \{\varepsilon\});$$

- *left-linear* if all productions in G are of the form

$$A \rightarrow \alpha \text{ with } \alpha \in (N \cup \{\varepsilon\})\Sigma^*;$$

- *regular* (REG) if G is either right-linear or left-linear;
- *strict linear* (SLIN) if all productions are of the form

$$A \rightarrow aBb \text{ or } A \rightarrow c \text{ with } B \in N \text{ and } a, b, c \in \Sigma^{\leq 1};$$

- *strict right-linear* if all productions are of the form

$$A \rightarrow aB \text{ or } A \rightarrow b \text{ with } B \in N \text{ and } a, b \in \Sigma^{\leq 1};$$

- *strict left-linear* if all productions are of the form

$$A \rightarrow Ba \text{ or } A \rightarrow b \text{ with } B \in N \text{ and } a, b \in \Sigma^{\leq 1};$$

- *strict regular* (SREG) if G is either strict right-linear or strict left-linear.

Remark. Unless explicitly stated otherwise, we implicitly assume that all (strict) regular grammars considered in this thesis are (strict) right-linear.

We will write SREG, REG, SLIN, LIN, and CF for the sets of strict regular, regular, strict linear, linear, and context-free, respectively, grammars and define the sets of grammar types as

$$\Gamma = \{\text{REG}, \text{LIN}, \text{CF}\}, \quad \Gamma_s = \{\text{SREG}, \text{SLIN}\}, \quad \text{and} \quad \Delta = \Gamma \cup \Gamma_s.$$

If a grammar G is an element of some set $X \in \Delta$, then we say that G is an X -grammar or, equivalently, that G is a grammar of type X .

Next, we will formally define the notion of derivability in a grammar.

Definition 2.1.4 (Derivation). Let $G = (N, \Sigma, P, S)$ be a context-free grammar. Then we write

$$\alpha A \beta \Rightarrow_G \alpha \gamma \beta,$$

for $\alpha, \beta, \gamma \in (N \cup \Sigma)^*$, if there is a production $A \rightarrow \gamma \in P$. The *derivation* relation \Rightarrow_G^* is the reflexive and transitive closure of \Rightarrow_G , that is, we write $\alpha \Rightarrow_G^* \beta$ if there are words $\alpha = \alpha_0, \alpha_1, \dots, \alpha_n = \beta$ over $N \cup \Sigma$ such that

$$\alpha_0 \Rightarrow_G \alpha_1 \Rightarrow_G \dots \Rightarrow_G \alpha_n.$$

If

$$\delta : \alpha \Rightarrow_G^* \beta,$$

then we say that δ is a *derivation of β from α* or, equivalently, that β is *derivable from α* . In particular, if $\alpha = S$, then δ is called a *G -derivation of β* (or *derivation of β in G*). If the grammar is clear from the context, we often omit the subscript G .

Remark. Note that the derivation in Definition 2.1.4

$$\alpha_0 \Rightarrow_G \alpha_1 \Rightarrow_G \dots \Rightarrow_G \alpha_n$$

consists of n derivation steps. So, for $k \geq 0$, we sometimes write

$$\alpha \Rightarrow_G^k \beta, \quad \alpha \Rightarrow_G^{\leq k} \beta, \quad \text{or} \quad \alpha \Rightarrow_G^{\geq k} \beta$$

if β can be derived from α in exactly k , at most k , or at least k steps, respectively. Moreover, if $\alpha \Rightarrow_G^{\geq 1} \beta$, then we also write $\alpha \Rightarrow_G^+ \beta$.

Let $G = (N, \Sigma, P, S)$ be a context-free grammar and $A, B \in N$. If α is an element of the set $(N \cup \Sigma)^*$, then α is called a *sentential form* if $S \Rightarrow_G^* \alpha$. We say that B is *reachable from A* if

$$A \Rightarrow_G^+ \alpha_1 B \alpha_2,$$

where $\alpha_1, \alpha_2 \in (N \cup \Sigma)^*$.

Based on the notion of derivability, we can finally define the language of a context-free grammar.

Definition 2.1.5 (Language of a Nonterminal/Grammar). Let $G = (N, \Sigma, P, S)$ be a context-free grammar and let $A \in N$. Then the *language of A in G* , denoted by $L_A(G)$, is defined as

$$L_A(G) = \{w \in \Sigma^* \mid A \Rightarrow_G^* w\}.$$

The *language of G* is then defined as $L(G) = L_S(G)$.

A language $L \subseteq \Sigma^*$ is called *regular*, *linear*, or *context-free* if there exists a regular, linear, or context-free, respectively, grammar G such that $L(G) = L$.

Remark. Note that all finite languages are regular, since any finite language L can be generated by a trivial grammar that simply lists all words in L using only the start symbol.

We say that two context-free grammars G_1 and G_2 are *equivalent* if $L(G_1) = L(G_2)$.

Let $G = (N, \Sigma, P, S)$ be a context-free grammar. Then we write $|G|$ for the number of productions in G , i.e., $|G| = |P|$. Furthermore, we write $|G|_s$ for the *size* of (or the number of symbols in) G . That is,

$$|G|_s = \sum_{A \rightarrow \alpha \in P} (|\alpha| + 2).$$

2.2 Automata

Now, we will introduce deterministic and nondeterministic finite automata in the classic sense as well as deterministic and nondeterministic finite cover automata.

All definitions in this section are based on [Woo87, CSY99, HMU01, Sha08, Câm14].

Definition 2.2.1 (Deterministic Finite Automaton). A *deterministic finite automaton* (or DFA for short) is a quintuple $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$, where

- Q is a finite non-empty *set of states*,
- Σ is a finite non-empty *input alphabet*,
- $\delta : Q \times \Sigma \rightarrow Q$ is a *transition function*,
- $q_0 \in Q$ is the *start or initial state*, and
- $F \subseteq Q$ is the *set of final states*.

The transition function δ can be extended to a transition function

$$\delta^* : Q \times \Sigma^* \rightarrow Q$$

as follows:

- $\delta^*(q, \varepsilon) = q$, for all $q \in Q$, and
- $\delta^*(q, wa) = \delta(\delta^*(q, w), a)$, for all $q \in Q$, $w \in \Sigma^*$, and $a \in \Sigma$.

Since δ^* agrees with δ on the domain of δ , we often just write δ instead of δ^* .

Moreover, we write $L(\mathcal{A})$ for the language accepted by the DFA \mathcal{A} , which is defined as follows:

$$L(\mathcal{A}) = \{w \in \Sigma^* \mid \delta^*(q_0, w) \in F\}.$$

Definition 2.2.2 (Nondeterministic Finite Automaton). A *nondeterministic finite automaton* (or NFA for short) is a quintuple $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$, where

- Q is a finite non-empty set of states,
- Σ is a finite non-empty input alphabet,
- $\delta : Q \times \Sigma \rightarrow \mathcal{P}(Q)$ is a transition function,
- $q_0 \in Q$ is the start or initial state, and
- $F \subseteq Q$ is the set of final states.

The extended transition function

$$\delta^* : Q \times \Sigma^* \rightarrow \mathcal{P}(Q)$$

is defined as follows:

- $\delta^*(q, \varepsilon) = \{q\}$, for all $q \in Q$, and
- $\delta^*(q, wa) = \bigcup_{r \in \delta^*(q, w)} \delta(r, a)$, for all $q \in Q$, $w \in \Sigma^*$, and $a \in \Sigma$.

Moreover, we write $L(\mathcal{A})$ for the language accepted by the NFA \mathcal{A} , which is defined as follows:

$$L(\mathcal{A}) = \{w \in \Sigma^* \mid \delta^*(q_0, w) \cap F \neq \emptyset\}.$$

Let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ be a DFA or an NFA. Then we say that an element of $Q\Sigma^*$ is a *configuration* of \mathcal{A} . It represents the current state as well as the remaining input of \mathcal{A} .

Remark. A configuration of a finite automaton \mathcal{A} contains all the information necessary to continue \mathcal{A} 's computation. Initially, \mathcal{A} is in configuration q_0w , where w is the input word. Finally, when \mathcal{A} has read all letters of its input, \mathcal{A} is in a configuration q , for some state q [Woo87].

Definition 2.2.3. Let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ be a DFA. If pw and qv are two configurations of \mathcal{A} , then we write

$$pw \xrightarrow{\mathcal{A}} qv \quad (\text{one execution step of } \mathcal{A} \text{ on } pw),$$

if $w = av$, for some $a \in \Sigma$, and $\delta(p, a) = q$. For $k \geq 1$, we write

$$pw \xrightarrow[\mathcal{A}]^k qv \quad (k \text{ execution steps of } \mathcal{A} \text{ on } pw)$$

if either $k = 1$ and $pw \mid_{\mathcal{A}} qv$, or $k > 1$ and there exists a configuration ru such that

$$pw \mid_{\mathcal{A}} ru \quad \text{and} \quad ru \mid_{\mathcal{A}}^{k-1} qv.$$

Moreover, we write

$$pw \mid_{\mathcal{A}}^{+} qv$$

if $pw \mid_{\mathcal{A}}^k qv$, for some $k \geq 1$. Similarly, we write

$$pw \mid_{\mathcal{A}}^{*} qv$$

if either $pw = qv$ or $pw \mid_{\mathcal{A}}^{+} qv$.

If, on the other hand, \mathcal{A} is an NFA, then we write

$$pw \mid_{\mathcal{A}} qv$$

if $w = av$, for some $a \in \Sigma$ and $q \in \delta(p, a)$. The binary relations $\mid_{\mathcal{A}}^k$, for $k \geq 1$, $\mid_{\mathcal{A}}^{+}$, and $\mid_{\mathcal{A}}^{*}$ for NFAs are defined just as for DFAs.

In addition to ordinary finite automata, we will also consider both deterministic and nondeterministic finite *cover automata* (DFCA and NFCA, respectively, for short) in this thesis. Deterministic cover automata have first¹ been introduced in [CSY99] and are used particularly for finite languages. In contrast to an ordinary finite automaton, a cover automaton for a given finite language L is allowed to accept—in addition to the words in L —other words having a length longer than the length of a longest word in L . To make this acceptance condition more precise, we will formally define cover automata as follows:

Definition 2.2.4 (Deterministic Finite Cover Automaton). Let $L \subseteq \Sigma^{\leq \ell}$ be a finite language with $\ell = \max\{|w| \mid w \in L\}$ and let \mathcal{A} be a DFA. Then \mathcal{A} is called a *deterministic finite cover automaton for L* (or DFCA for L for short) if

$$L(\mathcal{A}) \cap \Sigma^{\leq \ell} = L.$$

Similarly to the deterministic case, one can also define nondeterministic cover automata with the help of the modified acceptance condition.

Definition 2.2.5 (Nondeterministic Finite Cover Automaton). Let $L \subseteq \Sigma^{\leq \ell}$ be a finite language with $\ell = \max\{|w| \mid w \in L\}$ and let \mathcal{A} be an NFA. Then \mathcal{A} is called a *nondeterministic finite cover automaton for L* (or NFCA for L for short) if

$$L(\mathcal{A}) \cap \Sigma^{\leq \ell} = L.$$

¹However, as stated in [CSY99, Yu07], concepts similar to finite cover automata have already been studied before in different contexts, see, for instance, [BDG85, DS90, KF90, SB96].

2.3 Computational Complexity

We will assume that the reader is familiar with the syntax and semantics of classical propositional logic as well as the basic concepts of computational complexity theory as covered in, e.g., the books [Pap95, AB09, Sip13].

Let $f : \{0, 1\}^* \rightarrow \{0, 1\}$ be a *Boolean function*. Then, we identify the function f with the language

$$\Pi_f = \{I \in \{0, 1\}^* \mid f(I) = 1\} \subseteq \{0, 1\}^*$$

and call such sets Π_f *decision problems*. An instance I of a decision problem Π is called a *yes-instance* of Π if $I \in \Pi$. If, on the other hand, $I \notin \Pi$, then I is called a *no-instance* of Π . We identify the problem of computing f (i.e., given I , compute $f(I)$) with the problem of deciding the language Π_f (i.e., given I , decide whether $I \in \Pi_f$).

Remark. It is not a real restriction to consider only functions that operate on bit strings, since encodings can be used to represent different objects such as integers, pairs of integers, graphs, matrices, etc. as words over $\{0, 1\}$ [AB09].

By $|I|$, we denote the *size of an instance* I . As usual in complexity theory, by P and NP, we denote the classes of problems that are decidable in polynomial time on a deterministic and nondeterministic, respectively, Turing machine. The complexity class coNP contains the problems which are complements of problems in NP, that is, a problem Π is in coNP if and only if its complement $\text{co}\Pi$ is in NP. A decision problem Π is called *NP-hard* if every problem in NP is polynomial-time many-one reducible to Π . If Π is both in NP and NP-hard, then Π is called *NP-complete*. The canonical NP-complete problem is the *satisfiability* (or SAT for short) problem and it was the first decision problem that was shown to be NP-complete [Coo71]. Before we can define the satisfiability problem, we need to introduce some basic notions of classical propositional logic:

In the following, we denote the set of *propositional variables* by

$$\mathcal{V} = \{x_1, x_2, \dots\}.$$

The set of *propositional formulae* is the smallest set such that:

- (i) all propositional variables are propositional formulae, and
- (ii) if φ_1 and φ_2 are propositional formulae, then so are $\neg\varphi_1$, $\varphi_1 \wedge \varphi_2$, and $\varphi_1 \vee \varphi_2$.

We write $\text{var}(\varphi)$ for the set of variables occurring in a propositional formula φ and expressions of the form x and $\neg x$, for $x \in \mathcal{V}$, are called *literals*. A *clause*

$$\ell_1 \vee \ell_2 \vee \dots \vee \ell_n$$

is a finite disjunction of literals $\ell_1, \ell_2, \dots, \ell_n$. Similarly, a *conjunctive clause*

$$\ell_1 \wedge \ell_2 \wedge \dots \wedge \ell_n$$

is a finite conjunction of literals $\ell_1, \ell_2, \dots, \ell_n$.

Remark. For the sake of convenience, we will identify a conjunctive clause $C = \ell_1 \wedge \ell_2 \wedge \ell_3$ also with a set of literals $C = \{\ell_1, \ell_2, \ell_3\}$.

For an integer $k \geq 1$, we say that a formula φ is in k -CNF (*k-conjunctive normal form*) if it is of the form

$$\varphi = C_1 \wedge C_2 \wedge \dots \wedge C_m,$$

where each clause C_i , for $1 \leq i \leq m$, consists of at most k literals. Similarly, for an integer $k \geq 1$, we say that a formula φ is in k -DNF (*k-disjunctive normal form*) if it is of the form

$$\varphi = C_1 \vee C_2 \vee \dots \vee C_m,$$

where each conjunctive clause C_i , for $1 \leq i \leq m$, consists of at most k literals.

As usual, a *truth assignment* σ is a mapping

$$\sigma: \mathcal{V} \rightarrow \{0, 1\}$$

from propositional variables to the truth values 0 (*false*) and 1 (*true*). We can inductively extend σ from variables to formulae using truth tables. Then $\sigma(\varphi)$ denotes the truth value of the propositional formula φ under σ . To identify truth assignments with words over $\{0, 1\}$, we extend σ (by a slight abuse of notation) to a mapping

$$\sigma: \mathcal{V}^n \rightarrow \{0, 1\}^n, (x_1, x_2, \dots, x_n) \mapsto \sigma(x_1)\sigma(x_2)\dots\sigma(x_n),$$

that is,

$$\sigma(x_1, x_2, \dots, x_n) = \sigma(x_1)\sigma(x_2)\dots\sigma(x_n).$$

We extend the mapping $\sigma: \mathcal{V}^n \rightarrow \{0, 1\}^n$ to formulae φ as follows:

$$\sigma(x_1, x_2, \dots, x_n)(\varphi) = \sigma(\varphi).$$

We write

$$\sigma \models \varphi \quad \text{and} \quad \sigma(x_1, x_2, \dots, x_n) \models \varphi$$

if

$$\sigma(\varphi) = 1 \quad \text{and} \quad \sigma(x_1, x_2, \dots, x_n)(\varphi) = 1,$$

respectively. Similarly, we write

$$\sigma \not\models \varphi \quad \text{and} \quad \sigma(x_1, x_2, \dots, x_n) \not\models \varphi$$

if

$$\sigma(\varphi) = 0 \quad \text{and} \quad \sigma(x_1, x_2, \dots, x_n)(\varphi) = 0,$$

respectively.

A propositional formula φ is called *satisfiable* if there is a truth assignment σ such that $\sigma \models \varphi$, otherwise it is called *unsatisfiable*. We say that φ is a *tautology* if, for every truth assignment σ , we have that $\sigma \models \varphi$.

Now, we are ready for the definition of the SAT problem:

SATISFIABILITY

INSTANCE: A propositional formula φ .

QUESTION: Is φ satisfiable?

In order to show that problems are NP-hard, it is often convenient to reduce from a variant of the SAT problem in which all instances are in k -CNF. This leads to the k -SAT problem:

 k -SATISFIABILITY

INSTANCE: A propositional formula φ in k -CNF.

QUESTION: Is φ satisfiable?

Note that k -SAT is NP-complete only for $k \geq 3$.

A coNP-complete problem that is very important for showing the results in Chapter 7 is the *tautology* problem.²

TAUTOLOGY

INSTANCE: A propositional formula φ in 3-DNF.

QUESTION: Is φ a tautology?

Next, we will briefly recap the well-known big \mathcal{O} notation: Let f and g be functions from \mathbb{N} to \mathbb{R}^+ . Then we write

$$f(n) = \mathcal{O}(g(n))$$

if there are natural numbers c and n_0 such that, for all natural numbers $n \geq n_0$, we have that

$$f(n) \leq c \cdot g(n).$$

If

$$f(n) = \mathcal{O}(g(n)),$$

then we say that f grows as fast as g or slower and that $g(n)$ is an (asymptotic) upper bound for $f(n)$. Conversely, if

$$g(n) = \mathcal{O}(f(n)),$$

then we write

$$f(n) = \Omega(g(n))$$

and we say that f grows at least as fast as g and $g(n)$ is an (asymptotic) lower bound for $f(n)$. In the case that we have both

$$f(n) = \mathcal{O}(g(n)) \quad \text{and} \quad f(n) = \Omega(g(n)),$$

²The coNP-completeness of tautology follows from the NP-completeness of 3-SAT. For a proof w.r.t. the variant for arbitrary propositional formulae, we refer the reader to [AB09].

we write

$$f(n) = \Theta(g(n))$$

and say that f and g have the same rate of growth. Finally, we write that

$$f(n) = o(g(n))$$

if

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0.$$

Equivalently,

$$f(n) = o(g(n))$$

means that, for any real number $c > 0$, there exists a natural number n_0 such that, for all $n \geq n_0$, we have that

$$f(n) < c \cdot g(n).$$

If

$$f(n) = o(g(n)),$$

then we say that f grows slower than g , that is, f is asymptotically less than g .

In Chapter 7, we will also make use of a modified big \mathcal{O} notation which suppresses polynomial factors measured in the input length. Following [Woe08], for a function $f : \mathbb{N} \rightarrow \mathbb{R}^+$, we write

$$\mathcal{O}^*(f(n))$$

for a time complexity of the form

$$\mathcal{O}(f(n) \cdot \text{poly}(n)),$$

where $\text{poly}(n)$ is a polynomial in n .

The following part on hardness of approximation of NP-optimisation problems is based on [Cre97, WS11, DKH11].

An NP-optimisation problem (NPO) Π is a quadruple $(\mathcal{I}, F_{\text{sol}}, \text{cost}_{\Pi}, \text{type})$ such that

- \mathcal{I} is the set of instances of Π which is recognisable in polynomial time,
- for each instance $I \in \mathcal{I}$, $F_{\text{sol}}(I)$ denotes the set of feasible solutions of I . The size of these solutions is polynomial in the size of I , and the feasibility of a solution can be decided in polynomial time.
- for each instance $I \in \mathcal{I}$ and each feasible solution $s \in F_{\text{sol}}(I)$, $\text{cost}_{\Pi}(I, s)$ denotes the positive integer cost function of s which is computable in polynomial time.
- $\text{type} \in \{\min, \max\}$.

If $\text{type} = \min$, then Π is called a *minimisation problem*, and if $\text{type} = \max$, then Π is called a *maximisation problem*.

The goal of an NPO w.r.t. a given instance I is to find an *optimal solution* of I , i.e., to find a feasible solution s of I such that

$$\text{cost}_{\Pi}(I, s) = \text{opt}_{\Pi}(I) = \text{type}\{\text{cost}_{\Pi}(I, s') \mid s' \in F_{\text{sol}}(I)\}.$$

Remark. If not stated otherwise, we will assume that all considered NP-optimisation problems $\Pi = (\mathcal{I}, F_{\text{sol}}, \text{cost}_{\Pi}, \text{type})$ are minimisation problems, i.e., $\text{type} = \min$.

An approximation algorithm for an NPO produces solutions for a given instance whose cost is guaranteed to differ only within a certain factor from the cost of an optimal solution for that particular instance.

Definition 2.3.1 (Approximation Algorithm). Let $\Pi = (\mathcal{I}, F_{\text{sol}}, \text{cost}_{\Pi}, \text{type})$ be an NPO, let A be a polynomial-time algorithm, and let α be a positive real number. We say that A is an α -approximation algorithm for Π if, for all instances $I \in \mathcal{I}$, it holds that

- $A(I)$ is a feasible solution for I , i.e., the algorithm A run on instance I produces a feasible solution for I , and
- if $\text{type} = \min$, then $\alpha \geq 1$ and $\text{cost}_{\Pi}(I, A(I)) \leq \alpha \cdot \text{opt}_{\Pi}(I)$,
- if $\text{type} = \max$, then $\alpha \leq 1$ and $\text{cost}_{\Pi}(I, A(I)) \geq \alpha \cdot \text{opt}_{\Pi}(I)$.

The factor α is also called the *approximation factor* or *approximation ratio* of an α -approximation algorithm A for Π .

An NP-optimisation problem Π is called *inapproximable within factor α* (or α -inapproximable) if there is no α -approximation algorithm A for Π .

Now, we will introduce a useful tool for showing inapproximability within a certain factor for some NP-minimisation problems.

Definition 2.3.2 (Gap-Reduction). Let Π be an NP-minimisation problem and Π' an NP-hard decision problem. Then a polynomial-time many-one reduction f from Π' to Π is called *gap-reduction* from Π' to Π if there are positive real numbers γ_1 and γ_2 with $0 < \gamma_1 < \gamma_2$ such that

1. if I is a yes-instance of Π' , then $\text{opt}_{\Pi}(f(I)) \leq \gamma_1$, and
2. if I is a no-instance of Π' , then $\text{opt}_{\Pi}(f(I)) > \gamma_2$.

If such a gap-reduction from an NP-hard decision problem to Π exists, then we say that Π has an NP-hard gap of $\frac{\gamma_2}{\gamma_1}$.

We conclude this section with a theorem that shows that gap-reductions are indeed a useful tool for proving that some NP-minimisation problems are inapproximable within a certain factor.

Theorem 2.3.3 ([DKH11, Lemma 10.2]). *Let Π be an NP-optimisation problem with an NP-hard gap of $\frac{\gamma_2}{\gamma_1}$, where $0 < \gamma_1 < \gamma_2$. Then Π is inapproximable within the factor $\frac{\gamma_2}{\gamma_1}$, unless $P = NP$.*

PROOF. We only prove the case for minimisation problems—the proof for maximisation problems is analogous. To this end, assume that f is a gap-reduction from an NP-hard decision problem Π' to Π with gap $\frac{\gamma_2}{\gamma_1}$, i.e., γ_1 and γ_2 are positive real numbers with $0 < \gamma_1 < \gamma_2$ such that

1. if I is a yes-instance of Π' , then $\text{opt}_{\Pi}(f(I)) \leq \gamma_1$, and
2. if I is a no-instance of Π' , then $\text{opt}_{\Pi}(f(I)) > \gamma_2$.

Towards contradiction, assume that there is a polynomial-time $\frac{\gamma_2}{\gamma_1}$ -approximation algorithm A for problem Π . Then, based on A , we could construct an algorithm A' to decide Π' in polynomial time as follows:

1. On input instance I of Π' , compute instance $f(I)$ of problem Π .
2. Run algorithm A on instance $f(I)$ and compute solution $A(f(I))$ for Π .
3. Return yes if and only if $\text{cost}_{\Pi}(f(I), A(f(I))) \leq \gamma_2$.

If I is a yes-instance of Π' , then

$$\text{opt}_{\Pi}(f(I)) \leq \gamma_1,$$

by definition of f . As a consequence,

$$\text{cost}_{\Pi}(f(I), A(f(I))) \leq \frac{\gamma_2}{\gamma_1} \cdot \text{opt}_{\Pi}(f(I)) \leq \frac{\gamma_2}{\gamma_1} \cdot \gamma_1 = \gamma_2.$$

If, on the other hand, I is a no-instance of Π' , then

$$\text{opt}_{\Pi}(f(I)) > \gamma_2,$$


by definition of f . As a consequence,

$$\text{cost}_{\Pi}(f(I), A(f(I))) \geq \text{opt}_{\Pi}(f(I)) > \gamma_2.$$

Thus, A' returns yes if and only if I is a yes-instance of Π' , that is, A' decides the NP-complete problem Π' in polynomial time. This, in turn, implies $P = NP$. Therefore, Π cannot be approximated within the factor $\frac{\gamma_2}{\gamma_1}$, unless $P = NP$. \square

CHAPTER 3

Complexity Measures


 N this chapter, we will consider the notions of exact and cover complexity of finite languages both on an abstract and on a grammatical level. The emphasis will, however, be on questions regarding cover complexity measures. In particular, we will give a characterisation of the situations in which arbitrary cover complexity measures collapse to a bounded complexity measure. Moreover, for a restricted class of context-free grammars as well as for strict regular and strict linear grammars, we will show that the corresponding cover complexity measures for finite languages are unbounded. For these kinds of context-free grammars, we will then show that the cover complexity of a finite language L can be computed from the exact complexities of a finite number of finite covers $L' \supseteq L$.

Some of the results in this chapter have been published in [HW18a, HW19].

3.1 Exact and Cover Complexity

In this section, we will define several complexity measures for finite languages, and also what it means that a given finite language is (in)compressible w.r.t. a certain type of context-free grammar. Finally, we will give examples of grammar types that induce a bounded cover complexity measure as well as of those that induce an unbounded one.

Let us start with the introduction of abstract notions of exact and cover complexity measures for finite languages w.r.t. a given alphabet.

Definition 3.1.1 ([Cover] Complexity Measure). Let Σ be an alphabet. Then a function

$$\mu : \mathcal{P}_{\text{fin}}(\Sigma^*) \rightarrow \mathbb{N}$$

is called Σ -complexity measure. For a Σ -complexity measure μ , the *cover complexity measure induced by μ* is the Σ -complexity measure μ_C defined as

$$\mu_C(L) = \min\{\mu(L') \mid L \subseteq L' \in \mathcal{P}_{\text{fin}}(\Sigma^*)\}.$$

If the alphabet is irrelevant or clear from the context, we will just speak about a complexity measure.

Remark. Note that the minimum is well-defined even though there are infinitely many languages $L' \in \mathcal{P}_{\text{fin}}(\Sigma^*)$ with $L \subseteq L'$, since μ maps to the natural numbers.

By definition of μ_c , we have $\mu_c(L) \leq \mu(L)$, for all $L \in \mathcal{P}_{\text{fin}}(\Sigma^*)$. Moreover, for every language $L \in \mathcal{P}_{\text{fin}}(\Sigma^*)$, there is an $L' \supseteq L$ such that $\mu_c(L) = \mu(L')$. A Σ -complexity measure μ is called *bounded* if there is a $k \in \mathbb{N}$ such that $\mu(L) \leq k$, for all $L \in \mathcal{P}_{\text{fin}}(\Sigma^*)$, and *unbounded* otherwise.

Now, we will define, for a given grammar type $X \in \Delta$, the exact grammatical X -complexity of a finite language. Similarly to the more abstract definition above, we can then obtain the X cover complexity of L based on the exact X -complexities of finite supersets L' of L .

Definition 3.1.2 (X-Complexity). Let $L \in \mathcal{P}_{\text{fin}}(\Sigma^*)$ and $X \in \Delta$. Then the (exact) X -complexity of L is defined as

$$Xc(L) = \min\{|G| \mid G \in X \text{ and } L = L(G)\}.$$

Clearly, Xc is a complexity measure and induces the X cover complexity measure

$$Xcc(L) = \min\{Xc(L') \mid L \subseteq L' \in \mathcal{P}_{\text{fin}}(\Sigma^*)\}.$$

Remark. Note that the X cover complexity Xcc can also be defined as

$$Xcc(L) = \min\{|G| \mid G \in X, L \subseteq L(G), \text{ and } L(G) \text{ finite}\}.$$

Here, it is crucial that the language of the covering grammar is finite. Otherwise, we could always use the grammar that generates Σ^* in order to cover any language over Σ with a grammar whose number of productions is constant in the cardinality of L .

In order to illustrate the above definitions of grammatical complexity measures, we give a small example, which we literally take from [BMCIW81]:

Example 3.1.3. Let $\Sigma = \{a_1, a_2, \dots, a_n\}$, for $n \geq 2$, and L be the finite language

$$L = \{a_i a_j \mid i \neq j \text{ and } 1 \leq i, j \leq n\}.$$

How many context-free productions are needed in order to generate L ? Since the language L contains $n \cdot (n-1)$ words, $n \cdot (n-1)$ productions surely suffice. Already $4n-6$ productions are enough, as shown by the grammar $G = (N, \Sigma, P, S)$ with $N = \{S, A_2, \dots, A_{n-1}\}$ whose set of productions P contains the rules

$$\begin{aligned} S &\rightarrow a_i A_{i+1} \mid A_{i+1} a_i && \text{for } 1 \leq i \leq n-2, \\ A_i &\rightarrow A_{i+1} \mid a_i && \text{for } 2 \leq i \leq n-2, \\ A_{n-1} &\rightarrow a_{n-1} \mid a_n, \\ S &\rightarrow a_{n-1} a_n \mid a_n a_{n-1}. \end{aligned}$$

Thus, $\text{CFc}(L) \leq 4n - 6$. With a more elaborate construction, $2n + \mathcal{O}(n^{1/2})$ productions were shown to be sufficient for the language L in [BMCI83]. Later, it was shown in [AER83] that $2n + \lceil 2n^{1/2} \rceil$ context-free productions are necessary, for every $n \geq 6$.

What about the context-free cover complexity of L ? In order to obtain an upper bound for $\text{CFcc}(L)$, one can use the language Σ^2 with $\Sigma = \{a_1, a_2, \dots, a_n\}$, which is obviously a superset of L . Clearly, this language is generated by the context-free grammar $G = (\{S, A\}, \Sigma, P, S)$ with the productions

$$\begin{aligned} S &\rightarrow AA \\ A &\rightarrow a_1 \mid a_2 \mid \dots \mid a_n. \end{aligned}$$

Thus, $\text{CFcc}(L) \leq n + 1$.

Let τ be a *measure type* (e.g., the exact complexity c or the cover complexity cc) and $X, Y \in \Delta$. Then we define

$$X \leq_\tau Y \quad \text{if and only if} \quad X\tau(L) \leq Y\tau(L), \text{ for all finite languages } L.$$

In the case that $X \leq_\tau Y$, we say that X is *more succinct* than Y w.r.t. the measure type τ . By definition, the following relations hold for $\tau \in \{c, cc\}$:

$$\text{CF} \leq_\tau \text{LIN} \leq_\tau \text{REG} \leq_\tau \text{SREG} \quad \text{and} \quad \text{CF} \leq_\tau \text{LIN} \leq_\tau \text{SLIN} \leq_\tau \text{SREG}. \quad (3.1)$$

For $X \in \Delta$ and $G \in \mathcal{X}$, we say that G is a *minimal X -grammar covering (or generating) the finite language L* if $L(G) \supseteq L$ and $|G| = Xcc(L)$ (or $L(G) = L$ and $|G| = Xc(L)$, respectively). In general, there can be more than one minimal X -grammar for a given finite language L .

A finite language L is called *X -incompressible*, for $X \in \Gamma$, if any X -grammar generating L contains at least as many productions as there are words in L . The notion of (in)compressibility can also be extended to sequences of finite languages as well as to the cover formulation of grammatical complexity.

Definition 3.1.4. Let L be a finite language and $X \in \Gamma$. Then L is called *X -compressible* (or *X cover-compressible*), if $Xc(L) < |L|$ (or $Xcc(L) < |L|$, respectively) and *X -incompressible* (or *X cover-incompressible*, respectively) otherwise.

A sequence $(L_n)_{n \geq 1}$ of finite languages is called *X (cover-)incompressible*, for $X \in \Gamma$, if there is an $M \in \mathbb{N}$ such that for all $n \geq M$, the language L_n is X (cover-)incompressible. A sequence $(L_n)_{n \geq 1}$ of finite languages is called *X (cover-)compressible* if for every $M \in \mathbb{N}$, there is an $n \geq M$ such that L_n is X (cover-)compressible.

Remark. Note that the above definition of (in)compressibility is not well-suited for strict regular and strict linear grammars, as the number of productions needed to generate or cover a given language also depends on the length of a longest word in the language under consideration.

By definition of the strict grammar types, $X\tau(L) \leq |L|$, for $X \in \Gamma_s$ and $\tau \in \{c, cc\}$, does not hold for all finite languages L , as the following example demonstrates:

Example 3.1.5. Let $L = \{a^3\}$. Then

$$\text{SREG}_{cc}(L) = \text{SREG}_c(L) = 3 > 1 = |L| \quad \text{and} \quad \text{SLIN}_{cc}(L) = \text{SLIN}_c(L) = 2 > 1 = |L|.$$

The following result shows the existence of regular cover-incompressible sequences of finite languages (w.r.t. a fixed alphabet) and has been proved in [EH15a, EH18].

Theorem 3.1.6. *There is an alphabet Σ such that, for all $n \geq 1$, there is a language $L_n \subseteq \Sigma^*$ with $|L_n| = n = \text{REG}_{cc}(L_n)$.*

On the other hand, for every finite language L , there is a context-free grammar covering L with a constant number of productions:

Theorem 3.1.7. *Let $L \in \mathcal{P}_{\text{fin}}(\Sigma^*)$, then $\text{CF}_{cc}(L) \leq |\Sigma| + 2$.*

PROOF. Let $\Sigma = \{a_1, a_2, \dots, a_n\}$, $\ell = \max\{|w| \mid w \in L\}$, and consider the grammar G consisting of the productions

$$\begin{aligned} S &\rightarrow A^\ell, \\ A &\rightarrow a_1 \mid a_2 \mid \dots \mid a_n \mid \varepsilon. \end{aligned}$$

Then $L(G) = \Sigma^{\leq \ell} \supseteq L$ and $|G| = |\Sigma| + 2$. □

A close inspection of the statements of Theorems 3.1.6 and 3.1.7 reveals that the regular cover complexity measure is unbounded, while, on the other hand, the context-free cover complexity measure is bounded.

3.2 Unboundedness of Cover Complexity Measures

As we have seen in the previous section, some grammar types induce an unbounded, whereas others induce a bounded cover complexity measure. Therefore, in this section, we will characterise the situations in which a cover complexity measure collapses to a bounded complexity measure. Before we can give this characterisation, we need some auxiliary results on “almost inverting” functions from \mathbb{N} to \mathbb{N} . These will be provided in Lemmas 3.2.1 and 3.2.2.

A function $f : \mathbb{N} \rightarrow \mathbb{N}$ is called *bounded* if there is a $k \in \mathbb{N}$ such that $f(n) \leq k$, for all $n \in \mathbb{N}$, and *unbounded* otherwise. Moreover, a function f is called *monotonic* if $n \leq m$ implies $f(n) \leq f(m)$.

The two subsequent lemmas show that for every monotonic and unbounded function from \mathbb{N} to \mathbb{N} , there exist corresponding “almost inverse” functions.

Lemma 3.2.1. *Let $f : \mathbb{N} \rightarrow \mathbb{N}$ be both monotonic and unbounded, and define*

$$g : \mathbb{N} \rightarrow \mathbb{N}, n \mapsto \min\{i \in \mathbb{N} \mid n \leq f(i)\}.$$

Then g is well-defined, monotonic, unbounded, and for all $x, y \in \mathbb{N}$, we have that

$$g(x) \leq y \text{ iff } x \leq f(y).$$

PROOF. The function g is well-defined because, due to unboundedness of f , there is at least one $i \in \mathbb{N}$ with $f(i) \geq n$.

Moreover, g is unbounded, for suppose there would be a $k \in \mathbb{N}$ such that

$$g(n) = \min\{i \in \mathbb{N} \mid n \leq f(i)\} \leq k,$$

for all $n \in \mathbb{N}$. Then, in particular by monotonicity of f ,

$$g(f(k+1)) = \min\{i \in \mathbb{N} \mid f(k+1) \leq f(i)\} = k+1,$$

which contradicts $g(n) \leq k$, for all $n \in \mathbb{N}$.

Also, g is monotonic, for if $n \leq m$, then for any $x \in \mathbb{N}$ with $m \leq f(x)$, we also have $n \leq f(x)$ and thus $g(n) \leq x$, in particular for $x = g(m)$.

Let $x, y \in \mathbb{N}$. If $x \leq f(y)$, then $g(x) = \min\{i \in \mathbb{N} \mid x \leq f(i)\} \leq y$. On the other hand, if $g(x) \leq y$, then $f(g(x)) \leq f(y)$ and $f(g(x)) = f(\min\{i \in \mathbb{N} \mid x \leq f(i)\}) \geq x$. \square

Lemma 3.2.2. *Let $g : \mathbb{N} \rightarrow \mathbb{N}$ be both monotonic and unbounded, and define*

$$f : \mathbb{N} \rightarrow \mathbb{N}, n \mapsto \max\{i \in \mathbb{N} \mid g(i) \leq n\}.$$

Then f is well-defined, monotonic, unbounded, and for all $x, y \in \mathbb{N}$, we have

$$g(x) \leq y \text{ iff } x \leq f(y).$$

PROOF. The function f is well-defined because, for each $n \in \mathbb{N}$, there are only finitely many $i \in \mathbb{N}$ with $g(i) \leq n$, for suppose there would be infinitely many such i , then, by monotonicity, $g(j) \leq n$, for all j after a certain $j_0 \in \mathbb{N}$. This, however, contradicts the unboundedness of g .

Moreover, f is unbounded, for suppose there is a $k \in \mathbb{N}$ such that

$$f(n) = \max\{i \in \mathbb{N} \mid g(i) \leq n\} \leq k,$$

for all $n \in \mathbb{N}$. Then, in particular by monotonicity of g ,

$$f(g(k+1)) = \max\{i \in \mathbb{N} \mid g(i) \leq g(k+1)\} = k+1,$$

which contradicts $f(n) \leq k$, for all $n \in \mathbb{N}$.

Also, f is monotonic, for if $n \leq m$, then for any $x \in \mathbb{N}$ with $g(x) \leq n$, we have $g(x) \leq m$ and thus $f(m) \geq x$, in particular for $x = f(n)$.

Now, let $x, y \in \mathbb{N}$. If $g(x) \leq y$, then $f(y) = \max\{i \in \mathbb{N} \mid g(i) \leq y\} \geq x$. On the other hand, if $x \leq f(y)$, then $g(x) \leq g(f(y)) = g(\max\{i \in \mathbb{N} \mid g(i) \leq y\}) \leq y$. \square

Examples for “almost inverting” functions on the natural numbers are the exponential function 2^n and the function that applies the ceiling function to the binary logarithm of n :

Example 3.2.3. Let $f : \mathbb{N} \rightarrow \mathbb{N}$ and $g : \mathbb{N} \rightarrow \mathbb{N}$ be functions defined as follows:

$$f(n) = 2^n \quad \text{and} \quad g(n) = \begin{cases} 0 & \text{if } n = 0, \\ \lceil \log n \rceil & \text{if } n > 0. \end{cases}$$

Clearly, both f and g are monotonic and unbounded functions that, moreover, satisfy

$$f(n) = \max\{i \in \mathbb{N} \mid g(i) \leq n\} \quad \text{and} \quad g(n) = \min\{i \in \mathbb{N} \mid n \leq f(i)\}.$$

Thus, by Lemmas 3.2.1 and 3.2.2, it holds that

$$g(x) \leq y \text{ iff } x \leq f(y),$$

for all $x, y \in \mathbb{N}$.

The following notion of reference complexity measure is inspired by the fact that $Xc(L)$, for $X \in \Gamma$, is trivially bounded by the number of words occurring in a finite language L . Thus, what we have in mind for reference complexity measures are, e.g., the number of words $|L|$ in a language or their cumulated lengths $\|L\| = \sum_{w \in L} |w|$.

Definition 3.2.4 (Reference Complexity Measure). A complexity measure

$$\rho : \mathcal{P}_{\text{fin}}(\Sigma^*) \rightarrow \mathbb{N}$$

is called *reference complexity measure* if ρ is unbounded and $L_1 \subseteq L_2$ implies $\rho(L_1) \leq \rho(L_2)$.

Let μ be a complexity measure, then a reference complexity measure ρ is called *reference complexity measure for μ* if $\mu(L) \leq \rho(L)$, for all finite languages L .

Typical examples for the above definitions include: $\mu = Xc$, for $X \in \Gamma$, and $\rho(L) = |L|$, or μ is the minimal size, that is, symbolic complexity of an X -grammar and $\rho(L) = \|L\|$.

In the following theorem, a characterisation of the unboundedness of a cover complexity measure μ_c in terms of the existence of a relation between μ and a reference complexity measure ρ for μ is provided.

Theorem 3.2.5. Let μ be an unbounded Σ -complexity measure and let ρ be a reference complexity measure for μ . Then the following conditions are equivalent:

1. μ_c is unbounded
2. there is a monotonic and unbounded function $f : \mathbb{N} \rightarrow \mathbb{N}$ such that

$$\rho(L) \leq f(\mu(L)),$$

for all $L \in \mathcal{P}_{\text{fin}}(\Sigma^*)$.

3. there is a monotonic and unbounded function $g : \mathbb{N} \rightarrow \mathbb{N}$ such that

$$g(\rho(L)) \leq \mu(L),$$

for all $L \in \mathcal{P}_{\text{fin}}(\Sigma^*)$.

PROOF. The direction 2. \Rightarrow 3. has been shown in Lemma 3.2.1, and 3. \Rightarrow 2. in Lemma 3.2.2.

For the direction 3. \Rightarrow 1., let $L \in \mathcal{P}_{\text{fin}}(\Sigma^*)$. Then, by definition of μ_C , there is some language $L' \in \mathcal{P}_{\text{fin}}(\Sigma^*)$ such that $L \subseteq L'$ and $\mu_C(L) = \mu(L')$. Therefore,

$$\mu_C(L) = \mu(L') \geq g(\rho(L')) \geq g(\rho(L)),$$

where the first inequality follows from the assumption of condition 3., and the second one follows by definition of ρ as well as from the fact that g is a monotonic function. This shows the unboundedness of μ_C based on the unboundedness of g and ρ .

For showing 1. \Rightarrow 3., we argue by contraposition. Assume that every function $g : \mathbb{N} \rightarrow \mathbb{N}$ with $g(\rho(L)) \leq \mu(L)$, for all $L \in \mathcal{P}_{\text{fin}}(\Sigma^*)$, is bounded or not monotonic. In particular, consider

$$h : \mathbb{N} \rightarrow \mathbb{N}, n \mapsto \min\{\mu(L) \mid \rho(L) \geq n \text{ and } L \in \mathcal{P}_{\text{fin}}(\Sigma^*)\}$$

and note that—due to the unboundedness of ρ — h is well-defined. Moreover, we have $h(\rho(L)) \leq \mu(L)$. Now, we prove that h is monotonic and thus, by assumption, must be bounded. For monotonicity, let $n \leq m$. Then we have

$$\{L \in \mathcal{P}_{\text{fin}}(\Sigma^*) \mid \rho(L) \geq m \geq n\} \subseteq \{L \in \mathcal{P}_{\text{fin}}(\Sigma^*) \mid \rho(L) \geq n\}.$$

Therefore,

$$h(n) = \min\{\mu(L) \mid \rho(L) \geq n, L \in \mathcal{P}_{\text{fin}}(\Sigma^*)\} \leq \min\{\mu(L) \mid \rho(L) \geq m, L \in \mathcal{P}_{\text{fin}}(\Sigma^*)\} = h(m).$$

Thus h is bounded, i.e., there is a $k \in \mathbb{N}$ and $(L_n)_{n \in \mathbb{N}}$ such that $n \mapsto \rho(L_n)$ is unbounded, but $\mu(L_n) \leq k$, for all $n \in \mathbb{N}$. Since $\mu_C(L_n) \leq \mu(L_n) \leq k$, μ_C is bounded too. \square

The subsequent theorem states that the cover complexity of a finite language L can be obtained from the minimum over the exact complexities of finite supersets of L whose reference complexity is bounded by a certain constant.

Theorem 3.2.6. *Let μ be a complexity measure and ρ be a reference complexity measure for μ . Then, for every finite language L , there is some $b \in \mathbb{N}$ such that*

$$\mu_C(L) = \min\{\mu(L') \mid L \subseteq L' \in \mathcal{P}_{\text{fin}}(\Sigma^*) \text{ and } \rho(L') \leq b\}.$$

PROOF. If μ_C is bounded by k , let $b = k$. If μ_C is unbounded, then, by Theorem 3.2.5, there is a monotonic and unbounded function $g : \mathbb{N} \rightarrow \mathbb{N}$ such that

$$g(\rho(K)) \leq \mu(K),$$

for all finite languages K , and, by Lemma 3.2.2, there is a monotonic and unbounded function $f : \mathbb{N} \rightarrow \mathbb{N}$ such that

$$g(x) > y \text{ iff } x > f(y),$$

for all $x, y \in \mathbb{N}$. Let $b = f(\rho(L))$ and $L'' \supseteq L$ with $\rho(L'') > f(\rho(L))$, then $g(\rho(L'')) > \rho(L)$, and, since $\mu(L'') \geq g(\rho(L''))$, we obtain $\mu(L'') > \rho(L)$. Moreover, since $\rho(L) \geq \mu(L) \geq \mu_{\text{c}}(L)$, we have $\mu(L'') > \mu_{\text{c}}(L)$. \square

The above theorem expresses μ_{c} in terms of μ and ρ . Depending on ρ , the set of finite covers L' of L that is used to determine $\mu_{\text{c}}(L)$ may or may not be a finite set. We will analyse the reduction of $\mu_{\text{c}}(L)$ to the value of $\mu(\cdot)$ on a finite set more thoroughly in Section 3.4.

3.3 Unboundedness of Grammatical Cover Complexity

After dealing with complexity measures in an abstract sense in the previous section, we will now come back to applications in the realm of context-free grammars. In particular, we will apply Theorem 3.2.5 to the number of productions in various types of context-free grammars. Hence, we will fix $\rho(L) = |L|$ as reference complexity measure. The main result of this section is that any class of context-free grammars with a bound on the number of nonterminals allowed on the right-hand side of each production induces an unbounded cover complexity measure.

The subsequent lemma was already shown in [BMCIW81] and implies in conjunction with Theorem 3.2.5 that X_{cc} , for $X \in \{\text{SREG}, \text{REG}, \text{SLIN}, \text{LIN}\}$, is an unbounded complexity measure.

Lemma 3.3.1 ([BMCIW81, Lemma 2.3]). *Let G be a linear grammar with n productions generating a finite language, then $|L(G)| \leq 2^{n-1}$.*

Corollary 3.3.2. *The measures SREG_{cc} , REG_{cc} , SLIN_{cc} , and LIN_{cc} are unbounded.*

PROOF. Define the function $f : \mathbb{N} \rightarrow \mathbb{N}, n \mapsto 2^n$. Clearly, f is both monotonic and unbounded. By Lemma 3.3.1, for all finite languages $L \in \mathcal{P}_{\text{fin}}(\Sigma^*)$, we have

$$\rho(L) = |L| \leq 2^{\text{LIN}_{\text{c}}(L)-1} \leq 2^{\text{LIN}_{\text{c}}(L)} = f(\text{LIN}_{\text{c}}(L))$$

and hence, by Theorem 3.2.5, LIN_{cc} is unbounded. The unboundedness of the measures SREG_{cc} , REG_{cc} , and SLIN_{cc} follows from the facts that

$$\text{LIN}_{\text{cc}}(L) \leq \text{SLIN}_{\text{cc}}(L) \leq \text{SREG}_{\text{cc}}(L) \quad \text{and} \quad \text{LIN}_{\text{cc}}(L) \leq \text{REG}_{\text{cc}}(L),$$

for all finite languages $L \in \mathcal{P}_{\text{fin}}(\Sigma^*)$. \square

The following definition of *class* of CFGs in terms of closure under identifying nonterminals, omission of productions, and containment of all trivial grammars is motivated by the subsequent proofs of Lemmas 3.3.8 and 3.4.1.

Definition 3.3.3 (Class of CFGs). A set X of context-free grammars is called *class of context-free grammars* if

1. $(N, \Sigma, P, S) \in X$ and $p \in P$ implies $(N, \Sigma, P \setminus \{p\}, S) \in X$,
2. X is closed under identifying two nonterminals, and
3. for each finite language L , X contains the trivial grammar generating L .

A grammar in which every nonterminal derives at least one non-empty word is said to be in *pruned normal form*. More formally, this is defined as follows:

Definition 3.3.4 (Pruned Normal Form). A context-free grammar $G = (N, \Sigma, P, S)$ is in *pruned normal form* (PNF) if, for all nonterminals $A \in N \setminus \{S\}$, we have that

- $L_A(G) \not\subseteq \{\varepsilon\}$, and
- there are $\alpha_1, \alpha_2 \in (N \cup \Sigma)^*$ and a word $w \in L(G)$ such that $S \Rightarrow_G^* \alpha_1 A \alpha_2 \Rightarrow_G^* w$.

Remark. Note that if $L_A(G) = \emptyset$, then there is no A -production that derives a string that solely consists of terminal symbols, i.e., no A -production contributes to the derivation of a word in $L(G)$. By definition, a grammar in pruned normal form does not contain such useless nonterminals.

We now show that any context-free grammar can be transformed into an equivalent context-free grammar in PNF without increasing the number of productions.

Lemma 3.3.5. *Let G be a context-free grammar. Then there is a context-free grammar G' in PNF with $|G'| \leq |G|$ and $L(G') = L(G)$.*

PROOF. Let $G = (N, \Sigma, P, S)$ be a context-free grammar. If G is already in PNF, let $G' = G$; so assume that G is not in PNF. In the case that $L(G) = \emptyset$, let $G' = (N, \Sigma, \emptyset, S)$, and if $L(G) = \{\varepsilon\}$, let $G' = (N, \Sigma, \{S \rightarrow \varepsilon\}, S)$. In both of these cases, we have that G' is a context-free grammar in PNF with $|G'| \leq |G|$ and $L(G') = L(G)$.

Thus, assume $L(G) \not\subseteq \{\varepsilon\}$ and that there is some $A \in N \setminus \{S\}$ such that A does not occur in any derivation of a word in $L(G)$. We construct a grammar G' from G by removing the nonterminal A and all A -productions (i.e., productions in which A occurs on the left-hand side) as well as all productions in which A occurs on the right-hand side. We repeat this step until we have constructed a grammar $G'' = (N'', \Sigma, P'', S)$ such that for all $A \in N'' \setminus \{S\}$, there are $\alpha_1, \alpha_2 \in (N'' \cup \Sigma)^*$ and a word $w \in L(G'')$ with

$$S \Rightarrow_{G''}^* \alpha_1 A \alpha_2 \Rightarrow_{G''}^* w.$$

Then it follows that $L_A(G) \neq \emptyset$, for all $A \in N''$. It is also easy to see that $L(G'') = L(G)$ and $|G''| \leq |G|$. In the case that there is some nonterminal $A \in N'' \setminus \{S\}$ with $L_A(G'') = \{\varepsilon\}$, we construct a context-free grammar G''' by omitting the nonterminal A and all A -productions, and replacing all occurrences of A on the right-hand sides of the remaining

productions in G'' by ε . Then, clearly, $L(G''') = L(G'') = L(G)$ and $|G'''| \leq |G''| \leq |G|$. We repeat this step until we have constructed a grammar G^* which contains no nonterminal $A \neq S$ with $L_A(G^*) = \{\varepsilon\}$. This transformation of G into G^* clearly terminates as in each step the number of nonterminals decreases. \square

In order to illustrate the construction steps carried out in the proof of Lemma 3.3.5, we give the following example of transforming a context-free grammar into pruned normal form.

Example 3.3.6. Let $G = (N, \Sigma, P, S)$ be a context-free grammar with the following set of productions P :

$$\begin{aligned} S &\rightarrow aA_1 \mid bA_1 \mid B \mid aBC, \\ A_1 &\rightarrow aA_2 \mid bA_2 \mid B, \\ A_2 &\rightarrow a \mid b \mid B, \\ B &\rightarrow \varepsilon, \\ C &\rightarrow D. \end{aligned}$$

Clearly, $L(G) = \{a, b\}^{\leq 3} \not\subseteq \{\varepsilon\}$, and observe that $L_B(G) = \{\varepsilon\}$ and $L_C(G) = L_D(G) = \emptyset$. We follow the proof of Lemma 3.3.5 in order to construct a context-free grammar in PNF that is equivalent to G . Thus, we omit all B - and C -productions as well as the production $S \rightarrow aBC$ and replace each occurrence of B by ε in the remaining productions. This yields a grammar $G' = (\{S, A_1, A_2\}, \Sigma, P', S)$ with the following set of productions P' :

$$\begin{aligned} S &\rightarrow aA_1 \mid bA_1 \mid \varepsilon, \\ A_1 &\rightarrow aA_2 \mid bA_2 \mid \varepsilon, \\ A_2 &\rightarrow a \mid b \mid \varepsilon. \end{aligned}$$

We clearly have that $L(G') = L(G) = \{a, b\}^{\leq 3}$ and $|G'| \leq |G|$.

In some cases, it will be helpful to ensure that finite languages can be generated by a grammar which does not contain nonterminals that are reachable from themselves. This leads to the notion of an acyclic grammar:

Definition 3.3.7 (Cyclic Grammar). A context-free grammar $G = (N, \Sigma, P, S)$ is called *cyclic*¹ if there is some nonterminal $A \in N$ such that

$$A \Rightarrow_G^+ \alpha_1 A \alpha_2,$$

for $\alpha_1, \alpha_2 \in (N \cup \Sigma)^*$; otherwise G is called *acyclic*.

¹Note that the definition of a *cyclic* grammar slightly differs from that of a *self-embedding* one: a grammar $G = (N, \Sigma, P, S)$ is called *self-embedding* if there is some nonterminal $A \in N$ such that $A \Rightarrow_G^* \alpha_1 A \alpha_2$, for $\alpha_1, \alpha_2 \in (N \cup \Sigma)^+$; otherwise G is called *non self-embedding*.

Next, we show that if a grammar G that generates a finite language belongs to a class of context-free grammars, then that same class also contains an equivalent acyclic grammar with at most $|G|$ productions.

Lemma 3.3.8. *Let X be a class of context-free grammars or $X \in \Gamma_s$. If $G \in X$ and $L(G)$ is finite, then there is an acyclic grammar $G' \in X$ with $|G'| \leq |G|$ and $L(G') = L(G)$.*

PROOF. If G is acyclic, then define $G' = G$. Therefore, assume that G is cyclic, i.e., there is some $A_1 \in N$ and $\beta_1, \beta_2 \in (N \cup \Sigma)^*$ such that $A_1 \Rightarrow_G^+ \beta_1 A_1 \beta_2$. By Lemma 3.3.5, we can, without loss of generality, assume that G is in PNF, i.e., for all $B \in N \setminus \{S\}$, we have $L_B(G) \not\subseteq \{\varepsilon\}$, and B is both reachable from S and used to derive a word w in $L(G)$: there are $\alpha_1, \alpha_2 \in (N \cup \Sigma)^*$ such that $S \Rightarrow_G^* \alpha_1 B \alpha_2 \Rightarrow_G^* w$. If $\beta_1 \beta_2 \neq \varepsilon$, then, since A_1 is reachable from S and we have $A_1 \Rightarrow_G^+ \beta_1 A_1 \beta_2$, we can derive infinitely many words, i.e., $L(G)$ is infinite. Contradiction. On the other hand, if $\beta_1 \beta_2 = \varepsilon$, then there is a derivation of the form

$$A_1 \Rightarrow_G A_2 \Rightarrow_G \dots \Rightarrow_G A_n \Rightarrow_G A_1,$$

for $A_1, A_2, \dots, A_n \in N$ and $n \geq 1$. Note that the case that $A_1 \Rightarrow_G^* \gamma \Rightarrow_G^* A_1$ with $\gamma \in N^{\geq 2}$ is impossible, for otherwise assume that the nonterminals B and C occur in γ and satisfy, without loss of generality, $B \Rightarrow_G^* A_1$ and $C \Rightarrow_G^* \varepsilon$. Moreover, assume, without loss of generality, that B occurs to the left of C in γ . Due to G being in PNF, it follows that $L_C(G) \not\subseteq \{\varepsilon\}$, i.e., there is a word $v \in \Sigma^*$ such that $C \Rightarrow_G^* v$. However, this implies that

$$A_1 \Rightarrow_G^* \gamma \Rightarrow_G^* A_1 v \Rightarrow_G^* A_1 v^2 \Rightarrow_G^* \dots,$$

i.e., we could derive infinitely many words. Contradiction. We define a new grammar G^* from G with $|G^*| \leq |G|$ by identifying the nonterminals A_1, A_2, \dots, A_n with a nonterminal $A \notin N$. That is, we replace all A_i in G with $1 \leq i \leq n$ by A . Thus, every G -derivation can be transformed into a G^* -derivation and, vice versa, every G^* -derivation can be transformed into a G -derivation by adding suitable productions of the form $A_i \rightarrow A_j$. Consequently, we have $L(G^*) = L(G)$. Note that G^* still contains a production of the form $A \rightarrow A$. Let G' be the grammar obtained from removing the production $A \rightarrow A$ from G^* . Then $G' \in X$, $|G'| < |G^*| \leq |G|$, and $L(G') = L(G)$. \square

The following result shows that Lemma 3.3.1 can be generalised from linear to context-free grammars which contain a bounded number of nonterminals on the right-hand side of each of their productions:

Lemma 3.3.9. *Let G be a grammar with n productions generating a finite language such that every production of G contains at most $k \geq 0$ nonterminals on its right-hand side. Then the language of G contains at most $n^{(k+1)^n}$ words, i.e., $|L(G)| \leq n^{(k+1)^n}$.*

PROOF. We proceed by induction on the number of nonterminals p in G and show that $|L(G)| \leq n^{(k+1)^p}$. In light of Lemma 3.3.8, we can assume that G is acyclic.

- **Base case:** Assume that $p = 1$, i.e., the grammar G contains a single nonterminal S . Since G generates a finite language, S cannot occur on the right-hand side of any production. Thus, $k = 0$ and $L(G)$ contains exactly $n = n^{(0+1)^1}$ words.
- **Induction step:** Assume that G consists of n productions and contains the nonterminals A_1, A_2, \dots, A_{p+1} such that every production with left-hand side A_i only contains nonterminals A_j with $j < i$ on its right-hand side. We can assume the latter, since, by acyclicity of G , we can fix a linear order on the nonterminals in the above sense. The nonterminal A_1 is clearly minimal, i.e., cannot contain any nonterminals on the right-hand side of its productions. Thus, the productions with left-hand side A_1 are of the form

$$A_1 \rightarrow w_1 \mid w_2 \mid \dots \mid w_m$$

with $w_i \in \Sigma^*$, for $1 \leq i \leq m \leq n$. Moreover, let $B \rightarrow \alpha$ be an arbitrary production of G with $B \neq A_1$. We define the grammar G' from G by replacing

$$B \rightarrow \alpha$$

by the productions

$$B \rightarrow \alpha_1 \mid \alpha_2 \mid \dots \mid \alpha_{m'}$$

such that the α_i , for $1 \leq i \leq m'$, are all possible combinations of replacing the occurrences of the nonterminal A_1 in α by the words w_1, w_2, \dots, w_m . Clearly, it holds that $m' \leq m^k \leq n^k$. Moreover, we remove the nonterminal A_1 together with all A_1 -productions. Since this step is repeated for all non-minimal nonterminals, the grammar G' contains at most $n \cdot n^k = n^{k+1}$ productions and p nonterminals. Furthermore, we have $L(G') = L(G)$. By induction hypothesis, we get that

$$|L(G')| = |L(G)| \leq (n^{k+1})^{(k+1)^p} = n^{(k+1)^{p+1}}.$$

This concludes the induction.

Since there are at most n nonterminals in such a grammar G , we immediately get that $|L(G)| \leq n^{(k+1)^n}$. \square

Now, we are finally in the position to prove the main result of this section, namely, that every class of context-free grammars with a bounded number of nonterminals on the right-hand side of each production induces an unbounded cover complexity measure.

Corollary 3.3.10. *Let \mathcal{X} be a class of CFGs with a bounded number of nonterminals occurring on the right-hand side of each production. Then \mathcal{X}_{cc} is unbounded.*

PROOF. Let $G \in \mathcal{X}$ contain n production rules and let k be the bound on the number of nonterminals occurring on the right-hand side of each production. Define the function

$$f : \mathbb{N} \rightarrow \mathbb{N}, n \mapsto n^{(k+1)^n}.$$

Clearly, f is both monotonic and unbounded. By Lemma 3.3.9, we have, for all finite languages $L \in \mathcal{P}_{\text{fin}}(\Sigma^*)$, that

$$\rho(L) = |L| \leq \text{Xc}(L)^{(k+1)^{\text{Xc}(L)}} = f(\text{Xc}(L)).$$

Hence, by Theorem 3.2.5, Xcc is unbounded. \square

Remark. Note that Corollary 3.3.10 is in a certain sense a generalisation of Corollary 3.3.2.

An immediate consequence of Corollary 3.3.10 is that for the class CNF that consists of all grammars in Chomsky normal form² as well as of all trivial grammars, CNF_{cc} is an unbounded complexity measure. Moreover, by Lemma 3.3.9, the number of words generated by a grammar G in CNF with n productions is bounded above by n^{3^n} , i.e., $|L(G)| \leq n^{3^n}$.

3.4 Computing Cover Complexity from Exact Complexity

We now turn to characterising the cover complexity of a finite language L based on the exact complexity of a finite set of finite languages related to L . To this end, we first show that in a minimal context-free grammar which covers a finite language L , the number of terminals occurring on the right-hand side of each production is bounded by the length of a longest word in L .

Lemma 3.4.1. *Let X be a class of CFGs or $X \in \Gamma_s$, and let L be a finite language, $\ell := \max\{|w| \mid w \in L\}$, and G be a minimal X -grammar with $L(G) \supseteq L$. Then, for all productions in G of the form $A \rightarrow u_0 B_1 u_1 B_2 \cdots B_n u_n$ with $u_0, u_1, \dots, u_n \in \Sigma^*$, we have $|u_0 u_1 \cdots u_n| \leq \ell$.*

PROOF. Let $G = (N, \Sigma, P, S)$ be an X -grammar and suppose that there is a production

$$p: A \rightarrow u_0 B_1 u_1 B_2 \cdots B_n u_n \in P$$

with $|u_0 \cdots u_n| > \ell$. Then there is no G -derivation of a word in L that uses p , and so, the X -grammar $G' = (N, \Sigma, P \setminus \{p\}, S)$ satisfies both $L(G') \supseteq L$ and $|G'| < |G|$. Contradiction to the minimality of G . \square

Any linear grammar that covers a finite language whose longest word has length ℓ can generate only words of length at most the number of words in L times ℓ .

Lemma 3.4.2. *Let L be a finite language, $\ell := \max\{|w| \mid w \in L\}$, and G be a minimal X -grammar, for $X \in \{\text{REG}, \text{LIN}\}$, with $L(G) \supseteq L$. Then $\max\{|w| \mid w \in L(G)\} \leq |L| \cdot \ell$.*

PROOF. In light of Lemma 3.3.8, we can assume that G is acyclic. Let $w \in L(G)$ be arbitrary. Then there is a derivation δ of w in G of the form

$$S \Rightarrow_G \alpha_1 \Rightarrow_G \alpha_2 \Rightarrow_G \dots \Rightarrow_G \alpha_n = w,$$

²A context-free grammar $G = (N, \Sigma, P, S)$ is said to be in *Chomsky normal form* if all productions are of the form $A \rightarrow BC$, $A \rightarrow a$, or $S \rightarrow \epsilon$, where $A, B, C \in N$ and $a \in \Sigma$.

where $\alpha_i \in (N \cup \Sigma)^*$, for $1 \leq i \leq n$. Due to the fact that G is acyclic, each production of G can occur at most once in a G -derivation. Thus, by Lemma 3.4.1 and the fact that each production contains at most one nonterminal on the right-hand side, it follows that each derivation step in δ can add at most ℓ letters to the previous intermediate string, i.e., $|w| \leq n * \ell$. Since $n \leq |G| \leq |L|$, we get $|w| \leq |L| * \ell$. \square

Since, in general, it does not hold that $|G| \leq |L|$ if G is a minimal X -grammar covering L , for $X \in \Gamma_s$, we get a different bound on the length of a longest word in strict regular and strict linear grammars. Before we can prove this, we need the following auxiliary result.

Lemma 3.4.3. *Let $X \in \Gamma_s$ and $L \subseteq \Sigma^{\leq \ell}$. Then $Xc(L) \leq 1 + \sum_{i=1}^{\ell} i \cdot |\Sigma|^i$.*

PROOF. Consider the trivial regular grammar G generating $L = \{w_1, w_2, \dots, w_n\}$, i.e., each production of G is of the form

$$S \rightarrow a_{i,1}a_{i,2}\dots a_{i,k}$$

with $w_i = a_{i,1}a_{i,2}\dots a_{i,k}$ and $a_{i,j} \in \Sigma \cup \{\varepsilon\}$, for $1 \leq i \leq n$ and $1 \leq j \leq k \leq \ell$. We break up each of these trivial productions $S \rightarrow a_{i,1}a_{i,2}\dots a_{i,k}$ with $a_{i,j} \in \Sigma \cup \{\varepsilon\}$, for $1 \leq i \leq n$ and $1 \leq j \leq k \leq \ell$, into the following strict regular productions:

$$\begin{aligned} S &\rightarrow a_{i,1}A_{i,2} \\ A_{i,2} &\rightarrow a_{i,2}A_{i,3} \\ &\vdots \\ A_{i,k-1} &\rightarrow a_{i,k-1}A_{i,k} \\ A_{i,k} &\rightarrow a_{i,k}, \end{aligned}$$

where, for each $i \in \{1, 2, \dots, n\}$, the $A_{i,1}, A_{i,2}, \dots, A_{i,k}$ are fresh nonterminals. Consequently, if we assume that $\ell = \max\{|w| \mid w \in L\}$, we get that, for each $k \in \{1, 2, \dots, \ell\}$, we need at most

$$k \cdot |\Sigma|^k$$

strict regular productions, since there are $|\Sigma|^k$ many words of length k . Taking also the empty word ε into account, this amounts to

$$\text{SREGc}(L) \leq 1 + \sum_{i=1}^{\ell} i \cdot |\Sigma|^i,$$

for every finite language $L \subseteq \Sigma^{\leq \ell}$. The result for strict linear grammars follows immediately, since strict linear grammars are more succinct than strict regular ones w.r.t. the exact complexity. \square

Now, we are able to prove an upper bound on the length of a longest word in strict regular and strict linear grammars which cover a finite language.

Lemma 3.4.4. *Let $\ell \geq 0$, $L \subseteq \Sigma^{\leq \ell}$ be a finite language, and G be a minimal X -grammar, for $X \in \Gamma_s$, with $L(G) \supseteq L$. Then $\max\{|w| \mid w \in L(G)\} \leq \ell + \ell^3 \cdot |\Sigma|^\ell$.*

PROOF. The proof is essentially the same as the proof of Lemma 3.4.2, but instead of $n \leq |G| \leq |L|$, we have that

$$n \leq |G| \leq 1 + \sum_{i=1}^{\ell} i \cdot |\Sigma|^i \leq 1 + \ell^2 \cdot |\Sigma|^\ell$$

by Lemma 3.4.3, since one can show by induction on ℓ that $\sum_{i=1}^{\ell} i \cdot |\Sigma|^i \leq \ell^2 \cdot |\Sigma|^\ell$. \square

In the case of a context-free grammar G that covers a finite language $L \subseteq \Sigma^{\leq \ell}$ where the number of nonterminals occurring on the right-hand side of each production in G is bounded by some $k \geq 2$, we get that any such grammar can generate only words of length at most ℓ times $k^{|L|}$.

Lemma 3.4.5. *Let X be a class of context-free grammars such that every production in an X -grammar contains at most $k \geq 2$ nonterminals on its right-hand side, let L be a finite language, $\ell := \max\{|w| \mid w \in L\}$, and G be a minimal X -grammar with $L(G) \supseteq L$. Then*

$$\max\{|w| \mid w \in L(G)\} \leq \ell \cdot k^{|L|}.$$

PROOF. In light of Lemma 3.3.8, we can assume that G is acyclic and therefore fix a linear order on the nonterminals A_1, A_2, \dots, A_p in the following sense: every production with left-hand side A_i only contains nonterminals A_j with $j < i$ on its right-hand side. We show, by induction on q , that every derivation that starts with an A_j with $j \leq q \leq p$ consists of at most

$$\sum_{i=0}^{q-1} k^i$$

steps.

- **Base case:** If $q = 1$, then a derivation has at most one step, since A_1 -productions do not contain nonterminals on their right-hand sides.
- **Induction step:** If $q > 1$, then the first step replaces A_j with at most k occurrences of nonterminals which are some A_h with $h \leq q - 1$. By induction hypothesis, each of them has a derivation of length at most $\sum_{i=0}^{q-2} k^i$ and there are at most k of them, so the total number of steps in the derivation is at most

$$1 + k \sum_{i=0}^{q-2} k^i = \sum_{i=0}^{q-1} k^i.$$

This concludes the induction.

Moreover, for $k \geq 2$, we have $\sum_{i=0}^{q-1} k^i \leq k^q$, since $1 \leq k-1$ implies $k^q \leq k^q \cdot (k-1)$. As a consequence, $\frac{k^q}{k-1} \leq k^q$ and thus

$$\sum_{i=0}^{q-1} k^i = \frac{k^q - 1}{k - 1} \leq k^q.$$

The result is then obtained from $q \leq p \leq |G| \leq |L|$. Similarly as in the proof of Lemma 3.4.2, from acyclicity of G and Lemma 3.4.1, it follows that any word $w \in L(G)$ has length at most $\ell \cdot k^{|L|}$. \square

What the following theorem tells us is that, for a certain class of context-free grammars, we can obtain the cover complexity of a given finite language L in terms of the minimum over the exact complexities of a finite number of finite covers L' of L .

Theorem 3.4.6. *Let \mathcal{X} be a class of CFGs such that every production in an \mathcal{X} -grammar contains at most k nonterminals on its right-hand side. Then there exist functions*

$$f : \mathbb{N} \rightarrow \mathbb{N} \quad \text{and} \quad g : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$$

such that, for all finite languages $L \subseteq \Sigma^$, we have that*

$$X_{cc}(L) = \min\{X_c(L') \mid L \subseteq L', |L'| \leq f(|L|), \text{ and } \max\{|w| \mid w \in L'\} \leq g(|L|, \ell)\},$$

where $\ell = \max\{|w| \mid w \in L\}$.

PROOF. Let G be an arbitrary minimal \mathcal{X} -grammar with n productions covering a finite language L , i.e., $X_{cc}(L) = n$, and let $\ell = \max\{|w| \mid w \in L\}$. Clearly, $n \leq |L|$. We distinguish two cases. In the case that $k = 1$, G is a linear grammar and we define the functions

$$f : \mathbb{N} \rightarrow \mathbb{N}, x \mapsto 2^{x-1} \quad \text{and} \quad g : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}, (x, y) \mapsto x \cdot y.$$

Moreover, according to Lemmas 3.3.1 and 3.4.2, every \mathcal{X} -grammar covering L is an \mathcal{X} -grammar generating a finite language $L' \supseteq L$ that satisfies both

$$|L'| \leq 2^{X_c(L')-1} = 2^{X_{cc}(L)-1} \leq 2^{|L|-1} = f(|L|) \quad \text{and} \quad \max\{|w| \mid w \in L'\} \leq |L| \cdot \ell = g(|L|, \ell).$$

Since, by definition, $X_{cc}(L) = \min\{X_c(L') \mid L \subseteq L' \in \mathcal{P}_{fin}(\Sigma^*)\}$, setting

$$\mathcal{S}_{L,1} = \{L' \in \mathcal{P}_{fin}(\Sigma^*) \mid L \subseteq L', |L'| \leq 2^{|L|-1}, \text{ and } \max\{|w| \mid w \in L'\} \leq |L| \cdot \ell\}$$

yields the conclusion that

$$X_{cc}(L) = \min\{X_c(L') \mid L' \in \mathcal{S}_{L,1}\}.$$

Similarly, in the case that $k \geq 2$, the conclusion

$$X_{cc}(L) = \min\{X_c(L') \mid L' \in \mathcal{S}_{L,k}\}$$

follows from Lemmas 3.3.9 and 3.4.5 by defining the functions

$$f : \mathbb{N} \rightarrow \mathbb{N}, x \mapsto x^{(k+1)^x} \quad \text{and} \quad g : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}, (x, y) \mapsto k^x \cdot y.$$

and setting

$$\mathcal{S}_{L,k} = \{L' \in \mathcal{P}_{\text{fin}}(\Sigma^*) \mid L \subseteq L', |L'| \leq |L|^{(k+1)^{|L|}}, \text{ and } \max\{|w| \mid w \in L'\} \leq k^{|L|} \cdot \ell\}$$

Clearly, each of the sets $\mathcal{S}_{L,k}$, for $k \geq 1$, satisfies the conditions of Theorem 3.4.6. \square

Both the set of strict regular and the set of strict linear grammars are not classes of context-free grammars in the sense of Definition 3.3.3, as both of these sets do not contain all trivial grammars. Therefore, we have to adapt the proof strategy in order to arrive at a result for strict regular and strict linear grammars that is analogous to Theorem 3.4.6.

Theorem 3.4.7. *Let $X \in \Gamma_s$. Then there exist functions*

$$f : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N} \quad \text{and} \quad g : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$$

such that, for all finite languages $L \subseteq \Sigma^$, we have that*

$$X_{\text{cc}}(L) = \min\{X_{\text{c}}(L') \mid L \subseteq L', |L'| \leq f(\ell, |\Sigma|), \text{ and } \max\{|w| \mid w \in L'\} \leq g(\ell, |\Sigma|)\},$$

where $\ell = \max\{|w| \mid w \in L\}$.

PROOF. Let G be an arbitrary minimal X -grammar with n productions covering a finite language $L \subseteq \Sigma^*$, i.e., $X_{\text{cc}}(L) = n$, and let $\ell = \max\{|w| \mid w \in L\}$. First, we define the functions

$$f : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}, (x, y) \mapsto 2^{x^2 \cdot y^x} \quad \text{and} \quad g : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}, (x, y) \mapsto x + x^3 \cdot y^x.$$

Since $X_{\text{cc}}(L) \leq X_{\text{c}}(L)$, for all $L \in \mathcal{P}_{\text{fin}}(\Sigma^*)$, and due to Lemma 3.4.3 and the fact that strict linear grammars are more succinct than strict regular ones w.r.t. the cover complexity, we get that

$$X_{\text{cc}}(L) = n \leq 1 + \sum_{i=1}^{\ell} i \cdot |\Sigma|^i \leq 1 + \ell^2 \cdot |\Sigma|^{\ell}.$$

According to Lemmas 3.3.1 and 3.4.4, every X -grammar covering L is an X -grammar generating a finite language $L' \supseteq L$ that satisfies both

$$|L'| \leq 2^{X_{\text{c}}(L')-1} = 2^{X_{\text{cc}}(L)-1} \leq 2^{\ell^2 \cdot |\Sigma|^{\ell}} = f(\ell, |\Sigma|) \quad \text{and} \quad \max\{|w| \mid w \in L'\} \leq \ell + \ell^3 \cdot |\Sigma|^{\ell} = g(\ell, |\Sigma|).$$

Therefore, by setting

$$\mathcal{S}_L = \{L' \in \mathcal{P}_{\text{fin}}(\Sigma^*) \mid L \subseteq L', |L'| \leq 2^{\ell^2 \cdot |\Sigma|^{\ell}}, \text{ and } \max\{|w| \mid w \in L'\} \leq \ell + \ell^3 \cdot |\Sigma|^{\ell}\},$$

the conclusion follows. Clearly, the set \mathcal{S}_L satisfies the conditions of Theorem 3.4.7. \square

So, for a class of context-free grammars as in Theorem 3.4.6 as well as for strict regular and strict linear grammars, determining the cover complexity of L boils down to computing the exact complexity of finitely many supersets of L that only depend on $|L|$ and ℓ or $|\Sigma|$ and ℓ , where $\ell = \max\{|w| \mid w \in L\}$.

Bounds on Production Complexity



HIS chapter is devoted to proving several upper and lower bounds on various production complexity measures for both arbitrary and some specific finite languages. In addition to the exact and cover complexity of finite languages, we consider a third complexity measure—the so-called *scattered* complexity. For some finite languages, we do not only prove lower bounds, but we also show that they are incompressible w.r.t. certain complexity measures. For instance, in Section 4.3, we construct a regular cover-incompressible sequence of finite languages that generalises a previously known result from [EH15a, EH18].

Some of the results in this chapter have been published in [HW18b, GHW18, HW19].

4.1 Basic Bounds on Production Complexity

For the grammar types in Δ , we prove several bounds on the exact, cover, and scattered complexity of *arbitrary* finite languages in this section. Before we can do this, we need to define the *scattered subword* and the *scattered sublanguage* relations as well as the *X scattered complexity*. Intuitively, a word w' is a scattered subword of a word w if w' can be obtained from w by simply omitting letters. More formally, the definition reads as follows:

Definition 4.1.1 (Scattered Subword/Sublanguage). Let Σ be an alphabet and

$$w = w_1 u_1 w_2 u_2 \cdots u_{n-1} w_n$$

be a word with $w_i, u_j \in \Sigma^*$, for $1 \leq i \leq n$ and $1 \leq j \leq n-1$. Then the word

$$w' = w_1 w_2 \cdots w_n$$

is called a *scattered subword* of w and we write $w' \leq w$ or, equivalently, $w \geq w'$. If a word w_1 is not a scattered subword of w_2 , then we write $w_1 \not\leq w_2$ or, equivalently, $w_2 \not\geq w_1$.

We extend the relations “ \leq ” and “ \geq ” from words to languages L_1 and L_2 as follows:

$$\begin{aligned} L_1 \leq L_2 & \text{ if and only if for all words } w_1 \in L_1, \text{ there is a word } w_2 \in L_2 \text{ s.t. } w_1 \leq w_2, \\ L_2 \geq L_1 & \text{ if and only if } L_1 \leq L_2. \end{aligned}$$

If $L_1 \leq L_2$ (or $L_2 \geq L_1$) holds, we say that L_1 is a *scattered sublanguage* of L_2 as well as that L_2 is a *scattered cover* (or *scattered superlanguage*) of L_1 . On the other hand, if $L_1 \leq L_2$ does not hold, we also write $L_1 \not\leq L_2$ or, equivalently, $L_2 \not\geq L_1$.

In order to illustrate both the scattered subword and the scattered sublanguage relations, we give the following simple example.

Example 4.1.2. Let $\Sigma = \{a, b, c, \dots, x, y, z, -\}$ and consider the words

$$\begin{aligned} w_1 &= \text{free}, \\ w_2 &= \text{context-free}, \text{ and} \\ w_3 &= \text{context-sensitive}. \end{aligned}$$

Then $w_1 \leq w_2$, but $w_1 \not\leq w_3$.

Moreover, consider the following languages:

$$\begin{aligned} L_1 &= \{\text{logic}, \text{auto}, \text{free}\}, \\ L_2 &= \{\text{logistics}, \text{automata}, \text{context-free}\}, \text{ and} \\ L_3 &= \{\text{logistics}, \text{automata}, \text{context-sensitive}\}. \end{aligned}$$

Here, we have that $L_1 \leq L_2$, but $L_1 \not\leq L_3$.

Based on the scattered subword relation, we can define an additional production complexity measure on finite languages:

Definition 4.1.3 (X Scattered Complexity). Let $L \in \mathcal{P}_{\text{fin}}(\Sigma^*)$ and $X \in \Delta$. Then the X scattered complexity of L is defined as

$$X_{\text{sc}}(L) = \min\{|G| \mid G \in X, L \leq L(G), \text{ and } L(G) \text{ finite}\}.$$

Let $X, Y \in \Delta$. Then we define

$$X \leq_{\text{sc}} Y \quad \text{if and only if} \quad X_{\text{sc}}(L) \leq Y_{\text{sc}}(L), \text{ for all finite languages } L.$$

In the case that $X \leq_{\text{sc}} Y$, we say that X is *more succinct* than Y w.r.t. the measure type sc . By definition, the following relations hold:

$$\text{CF} \leq_{\text{sc}} \text{LIN} \leq_{\text{sc}} \text{REG} \leq_{\text{sc}} \text{SREG} \quad \text{and} \quad \text{CF} \leq_{\text{sc}} \text{LIN} \leq_{\text{sc}} \text{SLIN} \leq_{\text{sc}} \text{SREG}. \quad (4.1)$$

Similarly as in the case of exact and cover complexity, for $X \in \Delta$ and $G \in X$, we say that G is a *minimal X-grammar scatter-covering a finite language* L if $L \leq L(G)$ and $|G| = X_{\text{sc}}(L)$.

First, we prove some easy results on the exact and cover complexity of some frequently used finite languages, namely, on the set of all words of uniform length as well as on the set of all words up to a fixed length.

Lemma 4.1.4. *Let Σ be an alphabet and $X \in \Delta$. Then*

$$X_C(\Sigma^\ell) \leq |\Sigma| \cdot \ell \quad \text{and} \quad X_C(\Sigma^{\leq \ell}) \leq (|\Sigma| + 1) \cdot \ell,$$

for any integer $\ell \geq 1$.

PROOF. In light of the fact that strict regular grammars are less succinct than all other grammar types under consideration w.r.t. the exact complexity, it suffices to consider the case for SREG; the results for $X \in \{\text{SLIN}, \text{REG}, \text{LIN}\}$ then follow immediately. We define the SREG-grammar $G = (N, \Sigma, P, S)$, where $N = \{A_1, A_2, \dots, A_\ell\}$, with $S = A_1$ and

$$P = \{A_i \rightarrow aA_{i+1} \mid a \in \Sigma \text{ and } 1 \leq i \leq \ell - 1\} \cup \{A_\ell \rightarrow a \mid a \in \Sigma\}.$$

Obviously, $L(G) = \Sigma^\ell$ and $|G| = |\Sigma| \cdot \ell$.

In order to generate $\Sigma^{\leq \ell}$, we can simply extend G to a strict regular grammar G' as follows: let $G' = (N, \Sigma, P', S)$, where

$$P' = P \cup \{A_i \rightarrow \varepsilon \mid 1 \leq i \leq \ell\}.$$

Then we have $L(G') = \Sigma^{\leq \ell}$ and $|G'| = |G| + \ell = (|\Sigma| + 1) \cdot \ell$, as desired. \square

For the exact complexity w.r.t. context-free grammars, the bound of Lemma 4.1.4 can be improved. This can be seen as follows: obviously the context-free grammar $G = (\{S, A\}, \Sigma, P, S)$ whose production set P contains the rules

$$\begin{aligned} S &\rightarrow A^\ell \\ A &\rightarrow a \end{aligned} \quad \text{for each } a \in \Sigma,$$

generates the language Σ^ℓ . Adding the rule $A \rightarrow \varepsilon$ results in a CFG generating $\Sigma^{\leq \ell}$. This proves the upper bounds $\text{CFc}(\Sigma^\ell)$ and $\text{CFc}(\Sigma^{\leq \ell})$ of $|\Sigma| + 1$ and $|\Sigma| + 2$, respectively, for $\ell \geq 2$. The upper bounds $\text{CFc}(\Sigma) \leq |\Sigma|$ and $\text{CFc}(\Sigma^{\leq 1}) \leq |\Sigma| + 1$ follow from simply listing all elements of Σ (and additionally the empty word ε in the case of $\Sigma^{\leq 1}$) using just the start symbol S . All these bounds are tight. Assume to the contrary that there is a minimal context-free grammar G with $L(G) = \Sigma^{\leq \ell}$ and $|G| < |\Sigma| + 2$. Then we must have $|G| \geq |\Sigma| + 1$, since G must generate the empty word ε as well as all words starting with any of the terminals in Σ . Moreover, there can only be one production in G of the form $A \rightarrow a\alpha$ with $\alpha \in (N \cup \Sigma)^*$ for every letter $a \in \Sigma$. So, if $\alpha \in \Sigma^*$, then G can produce at most one word starting with the letter a . If α contains some nonterminal $B \neq A$ (assuming that G is acyclic, see Lemma 3.3.8), then we cannot produce all words consisting only of the letter a . Hence, the grammar G cannot generate $\Sigma^{\leq \ell}$. This is a contradiction to our assumption. Therefore, $\text{CFc}(\Sigma^{\leq \ell}) \geq |\Sigma| + 2$. An analogous argument shows that $\text{CFc}(\Sigma^\ell) \geq |\Sigma| + 1$. The above argument also goes through for the context-free cover complexity. We summarise these results in the subsequent lemma.

Lemma 4.1.5. *Let Σ be a finite alphabet. Then, for any integer $\ell \geq 2$,*

$$\text{CFc}(\Sigma^\ell) = \text{CFcc}(\Sigma^\ell) = |\Sigma| + 1 \quad \text{and} \quad \text{CFc}(\Sigma^{\leq \ell}) = \text{CFcc}(\Sigma^{\leq \ell}) = |\Sigma| + 2.$$

If $\ell = 1$, then we have $\text{CFc}(\Sigma) = \text{CFcc}(\Sigma) = |\Sigma|$ and $\text{CFc}(\Sigma^{\leq 1}) = \text{CFcc}(\Sigma^{\leq 1}) = |\Sigma| + 1$.

With the help of Lemmas 3.4.3, 4.1.4, and 4.1.5, we will now prove (upper) bounds on the exact, cover and scattered complexity of arbitrary finite languages for the different grammar types in Δ .

Theorem 4.1.6. *Let $L \subseteq \Sigma^{\leq \ell}$ be a finite language. Then, for $X \in \Delta$, $Y \in \Gamma$, $Z \in \{\text{REG}, \text{LIN}\}$, and $Z_s \in \Gamma_s$, we have that*

1.

$$Y_c(L) \leq \begin{cases} \ell + 1 & \text{if } |\Sigma| = 1, \\ \frac{|\Sigma|^{\ell+1} - 1}{|\Sigma| - 1} & \text{otherwise,} \end{cases}$$

$$\text{SREGc}(L) \leq 1 + \sum_{i=1}^{\ell} i \cdot |\Sigma|^i,$$

$$\text{SLINc}(L) \leq 1 + \sum_{i=1}^{\ell} \frac{(i+2)}{2} \cdot |\Sigma|^i.$$

2.

$$\text{CFcc}(L) \leq |\Sigma| + 2,$$

$$Z_{cc}(L) \leq \begin{cases} \ell + 1 & \text{if } |\Sigma| = 1, \\ (|\Sigma| + 1) \cdot \ell & \text{otherwise,} \end{cases}$$

$$Z_{scc}(L) \leq (|\Sigma| + 1) \cdot \ell.$$

3.

$$Y_{sc}(L) = \begin{cases} 1 & \text{if } L \text{ is non-empty,} \\ 0 & \text{otherwise,} \end{cases}$$

$$\begin{aligned} Z_{sc}(L) &\leq |\Sigma| \cdot \ell && \text{if } L \text{ is non-empty,} \\ Z_{sc}(L) &= 0 && \text{otherwise.} \end{aligned}$$

PROOF. We argue as follows.

1. Every finite language L can be generated by a grammar of type $Y \in \Gamma$ by simply listing all words in L . Since there are at most $\sum_{i=0}^{\ell} |\Sigma|^i$ words of length at most ℓ in L , the upper bounds of

$$\ell + 1 \text{ and } \sum_{i=0}^{\ell} |\Sigma|^i = \frac{|\Sigma|^{\ell+1} - 1}{|\Sigma| - 1}$$

follow for the cases $|\Sigma| = 1$ and $|\Sigma| \geq 2$, respectively.

The result for strict regular grammars follows from Lemma 3.4.3. Since for strict linear grammars we cannot proceed as for Y -grammars, every trivial production

$$S \rightarrow a_{i,1}a_{i,2}\dots a_{i,k} \quad \text{for } 1 \leq k \leq \ell,$$

that is responsible for deriving a word $w_i = a_{i,1}a_{i,2}\dots a_{i,k}$ of L has to be broken up into the following strict linear productions:

$$\begin{aligned} S &\rightarrow a_{i,1}A_{i,2}a_{i,k} \\ A_{i,2} &\rightarrow a_{i,2}A_{i,3}a_{i,k-1} \\ &\vdots \\ A_{i,\lceil \frac{k}{2} \rceil - 1} &\rightarrow a_{i,\lceil \frac{k}{2} \rceil - 1}A_{i,\lceil \frac{k}{2} \rceil}a_{i,k - \lceil \frac{k}{2} \rceil + 2}, \end{aligned}$$

where, for each $i \in \{1, 2, \dots, |L|\}$, the $A_{i,1}, A_{i,2}, \dots, A_{i,\lceil \frac{k}{2} \rceil}$ are fresh nonterminals. Moreover, we also add either the production

$$A_{i,\lceil \frac{k}{2} \rceil} \rightarrow a_{i,\lceil \frac{k}{2} \rceil} \quad \text{if } k \text{ is odd,}$$

or the fresh nonterminal $A_{i,\frac{k}{2}+1}$ together with the two productions

$$A_{i,\frac{k}{2}} \rightarrow a_{i,\frac{k}{2}}A_{i,\frac{k}{2}+1} \text{ and } A_{i,\frac{k}{2}+1} \rightarrow a_{i,\frac{k}{2}+1} \quad \text{if } k \text{ is even.}$$

Consequently, since $\lceil \frac{k}{2} \rceil \leq \frac{k}{2} + 1 = \frac{(k+2)}{2}$, we get that we need at most

$$\frac{(k+2)}{2} \cdot |\Sigma|^k \quad \text{for } 1 \leq k \leq \ell,$$

strict linear productions in order to generate all words of length k . Taking also the empty word ε into account, this amounts to

$$\text{SLINc}(L) \leq 1 + \sum_{i=1}^{\ell} \frac{(i+2)}{2} \cdot |\Sigma|^i,$$

for every finite language L .

2. Let ℓ be the length of a longest word in L . By assumption, $L \subseteq \Sigma^{\leq \ell}$. Thus, every grammar generating $\Sigma^{\leq \ell}$ automatically covers the language L . As a consequence, by Lemma 4.1.5, the claim for context-free grammars follows. Moreover, the results for regular and linear grammars where $|\Sigma| \geq 2$ as well as the results for their strict variants where $|\Sigma| \geq 1$ follow from Lemma 4.1.4. Finally, the results for regular and linear grammars where $|\Sigma| = 1$ follow from simply listing all $\ell + 1$ words occurring in $\Sigma^{\leq \ell}$ using non-strict trivial productions.
3. Assume that $\Sigma = \{a_1, a_2, \dots, a_n\}$ and consider the singleton language

$$\{(a_1 a_2 \dots a_n)^\ell\},$$

which is generated by

$$G = (\{S\}, \Sigma, \{S \rightarrow (a_1 a_2 \dots a_n)^\ell\}, S),$$

a regular grammar with a single production rule. Clearly, we have

$$L \leq \{(a_1 a_2 \dots a_n)^\ell\},$$

for all non-empty languages $L \subseteq \Sigma^{\leq \ell}$. Thus, $Y_{sc}(L) \leq 1$. Since any grammar with an empty production set can only generate the empty language, we also have $Y_{sc}(L) \geq 1$. In the case that $L = \emptyset$, we obviously have $Y_{sc}(L) = 0$. Since any regular grammar is both linear and context-free, we immediately get the respective results for the other two grammar types.

For the strict grammar types, we can use the fact that $\Sigma^\ell \geq L$, for any finite language L over Σ whose longest word has length ℓ . As a consequence, by Lemma 4.1.4, we obtain $Z_{sc}(\Sigma^\ell) \leq |\Sigma| \cdot \ell$, and so $Z_{sc}(L) \leq |\Sigma| \cdot \ell$.

□

For the uniform language Σ^ℓ with $\ell \geq 1$, we can even show that there is a corresponding lower bound on the SREG-complexity. In order to arrive at this result, we need a preliminary statement on context-free grammars generating uniform languages.

Lemma 4.1.7. *Let $X \in \Delta$ and $G = (N, \Sigma, P, S)$ be a minimal X -grammar generating a finite uniform language. Then, for all $A \in N$, all words occurring in the set $L_A(G)$ have the same length.*

PROOF. Let $A \in N$ be arbitrary. If N contains a nonterminal B that is not used to derive a word in $L(G)$, then

$$L_B(G) = \{w \in \Sigma^* \mid B \Rightarrow_G^* w\} = \emptyset$$

and the claim is vacuously true. Now, assume towards contradiction that there are two words w_1 and w_2 with $|w_1| \neq |w_2|$ such that $A \Rightarrow_G^* w_1$ and $A \Rightarrow_G^* w_2$. Note that minimal grammars do not contain useless productions, i.e., productions that are not used to

derive a word in Σ^* . For otherwise the grammar would not be minimal. Thus, there are derivations

$$S \Rightarrow_G^* \alpha_1 A \alpha_2 \Rightarrow_G^* \alpha_1 w_1 \alpha_2 \Rightarrow_G^* v_1 w_1 v_2 \quad \text{and} \quad S \Rightarrow_G^* \alpha_1 A \alpha_2 \Rightarrow_G^* \alpha_1 w_2 \alpha_2 \Rightarrow_G^* v_1 w_2 v_2,$$

for $\alpha_1, \alpha_2 \in (N \cup \Sigma)^*$ and $v_1, v_2 \in \Sigma^*$. Since $|w_1| \neq |w_2|$, it follows that $|v_1 w_1 v_2| \neq |v_1 w_2 v_2|$, i.e., $L(G)$ is not a uniform language. Contradiction! \square

An easy consequence of the previous lemma is that minimal grammars generating a uniform language—different from the language $\{\varepsilon\}$ —do not contain any ε -productions.

Proposition 4.1.8. *Let $X \in \Delta$ and $G = (N, \Sigma, P, S)$ be a minimal X -grammar generating a finite uniform language satisfying $L(G) \neq \{\varepsilon\}$. Then G is ε -free.*

PROOF. The proof is by contradiction. Assume to the contrary that G is not ε -free, i.e., there is some production $A \rightarrow \varepsilon \in P$. Then, from Lemma 4.1.7 and since $L(G) \neq \{\varepsilon\}$, it follows that $A \neq S$ and $L_A(G) = \{\varepsilon\}$. We define the following X -grammar $G' = (N, \Sigma, P', S)$, where P' is obtained from P by replacing all occurrences of A by ε on the right-hand side of each production and omitting the production $A \rightarrow \varepsilon$. Then, $L(G') = L(G)$ and $|G'| < |G|$. Contradiction to the minimality of G . \square

Now, we are able to prove a lower bound on the strict regular complexity of the language Σ^ℓ that matches the upper bound obtained in Lemma 4.1.4.

Lemma 4.1.9. *Let Σ be a finite alphabet and $\ell \geq 1$ an integer. Then $\text{SREGC}(\Sigma^\ell) \geq |\Sigma| \cdot \ell$.*

PROOF. Let $\ell \geq 1$ be an integer and $G = (N, \Sigma, P, S)$ be a minimal SREG-grammar with $L(G) = \Sigma^\ell$. We proceed by induction on ℓ :

- **Base case.** Let $\ell = 1$. Then $G = (\{S\}, \Sigma, P, S)$ with $P = \{S \rightarrow a \mid a \in \Sigma\}$ is clearly a minimal SREG-grammar with $L(G) = \Sigma$. Thus, $\text{SREGC}(\Sigma) \geq |\Sigma| \cdot 1$.
- **Induction step.** Towards contradiction assume that $G = (N, \Sigma, P, S)$ is a minimal SREG-grammar with $L(G) = \Sigma^{\ell+1}$ and $|G| < |\Sigma| \cdot (\ell + 1)$. Since $\Sigma^{\ell+1}$ is a uniform language with $\Sigma^{\ell+1} \neq \{\varepsilon\}$, it follows from Proposition 4.1.8 that G does not contain ε -productions. By definition of strict regular grammars, it is obvious that productions that produce the last letter of a word in $\Sigma^{\ell+1}$ are of the form $A \rightarrow a$, for $a \in \Sigma$. Since $\Sigma^{\ell+1}$ is a uniform language, it is not possible that any of these productions is used to derive a letter which is not the last letter of a word in $\Sigma^{\ell+1}$. Moreover, since one can show that in any minimal SREG-grammar that generates a finite language, any nonterminal derives at least two distinct words, it is impossible that there is a nonterminal $A \in N$ such that both $A \rightarrow a$ and $A \rightarrow bB$, for $a, b \in \Sigma$ and $B \in N$, are in P . For otherwise one could produce words over Σ that are longer than $\ell + 1$. Let $P_\Sigma = \{A \rightarrow a \in P \mid a \in \Sigma\}$ and $N_\Sigma = \{A \in N \mid A \rightarrow a \in P_\Sigma\}$. Since every letter in Σ is used to end a word in $\Sigma^{\ell+1}$, the following grammar $G' = (N, \Sigma, P', S)$

with

$$P' = P \setminus (P_\Sigma \cup \{B \rightarrow aA \in P \mid A \in N_\Sigma, B \neq A, \text{ and } a \in \Sigma\}) \\ \cup \{B \rightarrow a \mid B \rightarrow aA \in P, A \in N_\Sigma, B \neq A, \text{ and } a \in \Sigma\}$$

is a strict regular grammar generating Σ^ℓ with

$$|G'| \leq |G| - |\Sigma| < |\Sigma| \cdot (\ell + 1) - |\Sigma| = |\Sigma| \cdot \ell.$$

This, however, means that $\text{SREGc}(\Sigma^\ell) < |\Sigma| \cdot \ell$, contradicting the induction hypothesis. Thus, $\text{SREGc}(\Sigma^{\ell+1}) \geq |\Sigma| \cdot (\ell + 1)$. □

Open Problem 4.1.10. Is it possible to prove a lower bound on the strict linear complexity of Σ^ℓ that matches the upper bound obtained in Lemma 4.1.4? △

By combining Lemmas 4.1.4 and 4.1.9, we obtain the following result:

Proposition 4.1.11. *Let Σ be a finite alphabet and $\ell \geq 1$ an integer. Then $\text{SREGc}(\Sigma^\ell) = |\Sigma| \cdot \ell$.*

In the strict grammar variants, as opposed to the non-strict ones, the number of productions depends on the length of a longest word in the language to be covered. Therefore, we obtain the following lower bounds on the strict regular and the strict linear cover complexity of arbitrary finite languages.

Lemma 4.1.12. *Let $L \in \mathcal{P}_{\text{fin}}(\Sigma^*)$ be a finite language and $\ell = \max\{|w| \mid w \in L\}$. Then*

$$\text{SREGcc}(L) \geq \ell \quad \text{and} \quad \text{SLINcc}(L) \geq \left\lfloor \frac{\ell}{2} + 1 \right\rfloor.$$

PROOF. Since the result for strict regular grammars can be shown using similar arguments, we only give a proof of the result for strict linear grammars. We will first show that in any minimal strict linear grammar $G = (N, \Sigma, P, S)$ the following statement holds:

$$\text{for all } A \in N \text{ and all } w \in \Sigma^*: \text{ if } A \Rightarrow_G^k w, \text{ then } k \geq \left\lfloor \frac{|w|}{2} + 1 \right\rfloor.$$

To prove the above statement, we will proceed by induction on the length of a derivation of w .

- **Base case:** Assume $k = 1$. If $A \Rightarrow_G w$, then, by definition of strict linear grammars, we must have that $w \in \Sigma \cup \{\varepsilon\}$, i.e., $|w| \leq 1$. Thus, it clearly follows that

$$k = 1 \geq \left\lfloor \frac{1}{2} + 1 \right\rfloor = \left\lfloor \frac{|w|}{2} + 1 \right\rfloor.$$

- **Induction step:** Suppose $k \geq 2$ and $A \Rightarrow_G^k w$. We have to distinguish four cases according to the form of the derivation of w :

1. Let

$$A \Rightarrow_G aBb \Rightarrow_G^{k-1} w = aw_1b$$

with $a, b \in \Sigma$, $B \in N$, and $w_1 \in \Sigma^*$. Then, obviously,

$$B \Rightarrow_G^{k-1} w_1.$$

Thus, by induction hypothesis, it follows that

$$k - 1 \geq \left\lfloor \frac{|w_1|}{2} + 1 \right\rfloor.$$

This means that

$$\begin{aligned} k &\geq \left\lfloor \frac{|w_1|}{2} + 1 \right\rfloor + 1 = \left\lfloor \frac{|w_1|}{2} + 1 + 1 \right\rfloor = \left\lfloor \frac{|w_1|}{2} + \frac{2}{2} + 1 \right\rfloor \\ &= \left\lfloor \frac{|w_1| + 2}{2} + 1 \right\rfloor = \left\lfloor \frac{|w|}{2} + 1 \right\rfloor. \end{aligned}$$

2. Next, consider

$$A \Rightarrow_G aB \Rightarrow_G^{k-1} w = aw_1$$

with $a \in \Sigma$, $B \in N$, and $w_1 \in \Sigma^*$. The claim follows from similar arguments as in the first case.

3. Moreover,

$$A \Rightarrow_G Bb \Rightarrow_G^{k-1} w = w_1b$$

with $b \in \Sigma$, $B \in N$, and $w_1 \in \Sigma^*$. The claim follows from similar arguments as in the first case.

4. Finally,

$$A \Rightarrow_G B \Rightarrow_G^{k-1} w$$

with $B \in N$ and $w \in \Sigma^*$. Obviously,

$$B \Rightarrow_G^{k-1} w.$$

By induction hypothesis, we get

$$k - 1 \geq \left\lfloor \frac{|w|}{2} + 1 \right\rfloor,$$

which clearly implies that

$$k \geq \left\lfloor \frac{|w|}{2} + 1 \right\rfloor.$$

This concludes the induction.

Now, let L be a finite language over Σ with $\ell = \max\{|w| \mid w \in L\}$ and $G = (N, \Sigma, P, S)$ be a minimal strict linear grammar with $L(G) \supseteq L$. Then there is a derivation δ of the form

$$S \Rightarrow_G^k w$$

with $w \in \Sigma^\ell$ and $k \geq 1$. By the above statement, we get that

$$k \geq \left\lfloor \frac{|w|}{2} + 1 \right\rfloor = \left\lfloor \frac{\ell}{2} + 1 \right\rfloor.$$

By Lemma 3.3.8, we can assume, without loss of generality, that G is acyclic. Since in an acyclic strict linear grammar all right-hand sides of productions contain at most one nonterminal, no production can occur twice in the derivation δ . As a consequence, the derivation δ uses k distinct productions in order to derive w . Hence,

$$\text{SLINcc}(L) = |G| \geq k \geq \left\lfloor \frac{\ell}{2} + 1 \right\rfloor,$$

by minimality of G . □

Since every grammar that generates a language also covers that language, we immediately get the following corollary from Lemma 4.1.12.

Corollary 4.1.13. *Let $L \in \mathcal{P}_{\text{fin}}(\Sigma^*)$ be a finite language and $\ell = \max\{|w| \mid w \in L\}$. Then*

$$\text{SREGc}(L) \geq \ell \quad \text{and} \quad \text{SLINc}(L) \geq \left\lfloor \frac{\ell}{2} + 1 \right\rfloor.$$

Every grammar that generates a finite language which is a scattered superlanguage of a finite language L clearly contains at least one word which is at least as long as a longest word in L . Therefore, from Lemma 4.1.12, we get the following corollary for the strict regular and strict linear scattered complexity of finite languages.

Corollary 4.1.14. *Let $L \in \mathcal{P}_{\text{fin}}(\Sigma^*)$ be a finite language and $\ell = \max\{|w| \mid w \in L\}$. Then*

$$\text{SREGsc}(L) \geq \ell \quad \text{and} \quad \text{SLINsc}(L) \geq \left\lfloor \frac{\ell}{2} + 1 \right\rfloor.$$

PROOF. Let $X \in \Gamma_s$ and $w \in L$ with $|w| = \ell$ be arbitrary. Then, for every X -grammar G that generates a finite language and satisfies $L(G) \supseteq L$, we have that there is some $w' \in L(G)$ such that $w \leq w'$. But this means that $|w'| \geq |w|$, i.e., $\max\{|v| \mid v \in L(G)\} \geq \ell$. Therefore, by Corollary 4.1.13, it follows that $|G| \geq \ell$ if $X = \text{SREG}$ and $|G| \geq \left\lfloor \frac{\ell}{2} + 1 \right\rfloor$ if $X = \text{SLIN}$. □

Remark. The exact complexity for unary languages has already been settled in [BMCIW81, Theorem 5] in the sense that any finite unary language L can be generated by a context-free grammar having a constant as well as by a regular or linear grammar having a logarithmic (in the length of a longest word in L) number of productions.

4.2 Lower Bounds on Exact Production Complexity

In this section, we consider different finite languages and show that some of them are incompressible w.r.t. certain grammar types, that is, every grammar generating such a language needs at least as many productions as there are words in that language. Some incompressible languages can already be found in the seminal papers [BMCIW81, Buc81] on concise description of finite languages by different types of grammars. The proofs of these results are based on the following lemma, which states some easy facts about minimal context-free grammars generating finite languages.

Lemma 4.2.1 ([BMCIW81], Lemma 2.1). *Let $G = (N, \Sigma, P, S)$ be a minimal context-free grammar generating the finite language L . Then, for every nonterminal $A \in N \setminus \{S\}$,*

- *there are α_1 and α_2 with $\alpha_1, \alpha_2 \in (N \cup \Sigma)^*$ and $\alpha_1 \neq \alpha_2$ such that $A \rightarrow \alpha_1$ and $A \rightarrow \alpha_2$ are in P ,*
- *the set $L_A(G)$ contains at least two words, and*
- *there are words $u_1, u_2, v_1, v_2 \in \Sigma^*$ such that*

$$u_1 A u_2 \neq v_1 A v_2$$

and both

$$S \Rightarrow_G^* u_1 A u_2 \quad \text{and} \quad S \Rightarrow_G^* v_1 A v_2$$

hold.

Moreover, for all $A \in N$, there is no derivation of the form $A \Rightarrow_G^+ \alpha A \beta$ with $\alpha, \beta \in (N \cup \Sigma)^*$.

Remark. For strict regular and strict linear grammars it is fairly easy to construct finite languages with an arbitrarily high production complexity. In particular, for each integer $n \geq 1$, consider the languages $L_1 = \{a^n\}$ and $L_2 = \{a^{2n-1}\}$. Due to Lemma 4.1.12, every SREG-grammar covering L_1 as well as every SLIN-grammar covering L_2 needs at least n productions, i.e., $\text{SREGcc}(L_1) \geq n$ and $\text{SLINcc}(L_2) \geq n$. Therefore, in this section, the emphasis will be on the production complexity w.r.t. the non-strict grammar types.

One of the languages shown to be incompressible in [BMCIW81] is

$$U_n = \{a^k b^k c a^\ell b^\ell d a^m b^m \mid k + \ell + m = n\},$$

which contains $\binom{n+2}{2}$, i.e., quadratically many, words and satisfies $\text{CFC}(U_n) = |U_n|$. Yet another example is the language

$$R_n = \{a^i b^j \mid i + j = n\}$$

with $\text{REGc}(R_n) = |R_n| = n + 1$.

Further examples of incompressible languages can be found in [Buc81]—we only mention the language

$$L_n = \{a^i b^i c^i \mid 1 \leq i \leq n\}$$

with

$$\text{CFc}(L_n) = |L_n| = n \quad (4.2)$$

as well as the language

$$V_n = \{a^i b a^j c a^i \mid 2i + j = n - 2\}$$

with

$$\text{LINc}(V_n) = |V_n|.$$

Note that the cardinality of V_n is linear in n .

A careful inspection of [BMCIW81, Buc81] reveals that the cardinalities of the incompressible languages presented there are polynomial in the length of a longest word, but what about incompressible finite languages with larger cardinalities?

While there are barely any results that show exponential lower bounds on the production complexity of finite languages, it is, in fact, known that some finite languages only admit context-free grammars with high size. Before we take a closer look at some of these languages, we will define the *symbolic X-complexity* (or *minimal X-size*) of a finite language. First, recall that, for a context-free grammar $G = (N, \Sigma, P, S)$, the size of G is defined as $|G|_s = \sum_{A \rightarrow \alpha \in P} (|\alpha| + 2)$.

Definition 4.2.2. Let $L \in \mathcal{P}_{\text{fin}}(\Sigma^*)$ and $X \in \Delta$. Then the *symbolic X-complexity* of L is defined as

$$\text{Xsz}(L) = \min\{|G|_s \mid G \in X \text{ and } L(G) = L\}.$$

Let $X, Y \in \Delta$. Then we define

$$X \leq_{\text{sz}} Y \quad \text{if and only if} \quad \text{Xsz}(L) \leq \text{Ysz}(L), \text{ for all finite languages } L.$$

In the case that $X \leq_{\text{sz}} Y$, we say that X is *more succinct* than Y w.r.t. the measure type sz . By definition, the following relations hold:

$$\text{CF} \leq_{\text{sz}} \text{LIN} \leq_{\text{sz}} \text{REG} \leq_{\text{sz}} \text{SREG} \quad \text{and} \quad \text{CF} \leq_{\text{sz}} \text{LIN} \leq_{\text{sz}} \text{SLIN} \leq_{\text{sz}} \text{SREG}. \quad (4.3)$$

As a first example of a finite language with a high symbolic CF-complexity, consider the so-called (*bounded*) *copy language*

$$C_n = \{w\$w \mid w \in \{a, b\}^n\},$$

where $n \geq 1$ is an integer. Recently, in [Fil11], it was shown that any context-free grammar for C_n has size at least $\Omega(2^{n/4}/\sqrt{2n})$. The technique presented in [Fil11] in order to prove this result is quite involved and generalises a previously known result from [EKSW05] on an exponential lower bound on the size of context-free grammars generating the set of all permutations over a finite alphabet.

We can summarise the technique of [Fil11] for finite uniform languages as follows¹:

First, for a given finite uniform language L , define a reflexive and symmetric relation \sim on a set V_L such that $u \sim v$ if there are words w_1 and w_2 such that

$$w_1 u w_2, w_1 v w_2 \in L,$$

where

$$V_L = \{v \mid v \text{ is a subword of some } w \in L, \text{ and } n/2 \leq |v| < n\}$$

and n is the length of the words occurring in L . Moreover, we also need the notion of a so-called *clique*, i.e., a subset C of V_L such that $u \sim v$ holds for all $u, v \in C$. Then the following result is proved:

Lemma 4.2.3 ([Fil11, Proposition 6]). *Let L be a finite uniform language. Then every context-free grammar G that generates L satisfies*

$$|G|_s = \Omega\left(\sqrt{\frac{|L|}{M}}\right),$$

where M is the maximal number of words in L that have some subword in a clique C .

The proof of this result proceeds by transforming the given context-free grammar into an equivalent context-free grammar G in Chomsky normal form and then showing that it contains at least $\frac{|L|}{M}$ nonterminals. By the well-known subword lemma for grammars in Chomsky normal form², one can associate to each $w \in L$ a subword $v(w) \in V_L$ that is generated by some nonterminal $N(w)$. It is then shown that the sets

$$N^{-1}(A) = \{w \in L \mid N(w) = A\},$$

where A is a nonterminal, form a partition of L into parts of cardinality at most M . Therefore, G must contain at least $\frac{|L|}{M}$ nonterminals and thus be of size at least $\frac{|L|}{M}$. For arbitrary context-free grammars, we therefore obtain a lower bound on the size of

$$\Omega\left(\sqrt{\frac{|L|}{M}}\right),$$

since the transformation into Chomsky normal form may increase the size by a quadratic factor [LL09].

¹Note that the result of [Fil11, Proposition 6] is more general than the one presented here, as it is stated for arbitrary context-free languages. However, for ease of presentation and since the languages C_n and T_n (see the next page for the definition) are both finite and uniform, we formulate it only for finite uniform languages.

²The subword lemma states that if a word w with $|w| \geq 2$ is generated by a context-free grammar G in Chomsky normal form, then, for each positive $\ell \leq |w|$, there is a subword v of w of length $\ell/2 \leq |v| < \ell$ that is generated by a nonterminal of G [Sha08, Fil11].

Note that the above mentioned lower bound for the language C_n is *not* enough to prove that this language is incompressible in our setting, because C_n contains 2^n many words. Even for more complicated languages such as

$$T_n = \{w\$w\#w \mid w \in \{a, b\}^n\},$$

the language of all triples of length $n \geq 1$, the technique of [Fil11] does not suffice for proving incompressibility. Here, the lower bound induced by [Fil11] is $\Omega(2^{n/8}/\sqrt{3n})$.

In order to obtain the above mentioned lower bounds for the languages C_n and T_n , we can apply Lemma 4.2.3 as follows³:

First, it is shown that any clique w.r.t. either of those languages consists just of a single word. Then it is shown that at most $2n \cdot 2^{n/2}$ and $3n \cdot 2^{3n/4}$ many words in C_n and T_n , respectively, contain any given word in V_{C_n} and V_{T_n} , respectively. Since $|C_n| = |T_n| = 2^n$, we have that

$$\frac{|C_n|}{2n \cdot 2^{n/2}} = \frac{2^{n/2}}{2n} \quad \text{and} \quad \frac{|T_n|}{3n \cdot 2^{3n/4}} = \frac{2^{n/4}}{3n}.$$

Finally, by Lemma 4.2.3, we get

$$\Omega\left(\frac{2^{n/4}}{\sqrt{2n}}\right) \quad \text{and} \quad \Omega\left(\frac{2^{n/8}}{\sqrt{3n}}\right)$$

as lower bounds on the context-free size of the languages C_n and T_n , respectively.

Using the facts stated in Lemma 4.2.1, we can show that T_n is generated minimally by a context-free grammar only by simply listing all words. This shows that T_n is incompressible in our sense, and can be derived using the classic technique from [BMCIW81].

Theorem 4.2.4. *Let $X \in \Gamma$ and $n \geq 1$ an integer. Then $Xc(T_n) = |T_n| = 2^n$.*

PROOF. Let $G = (N, \Sigma, P, S)$ be a minimal context-free grammar generating T_n . Moreover, let $A \in N \setminus \{S\}$ be an arbitrary nonterminal. By Lemma 4.2.1, there are derivations

$$S \Rightarrow_G^* u_1 A u_2 \quad \text{and} \quad S \Rightarrow_G^* w_1 A w_2$$

with $u_1, u_2, w_1, w_2 \in \Sigma^*$ and $u_1 A u_2 \neq w_1 A w_2$ as well as

$$x, y \in \Sigma^* \text{ with } x \neq y, A \Rightarrow_G^* x, \text{ and } A \Rightarrow_G^* y.$$

We will first show that it is impossible that $x \in \{a, b\}^*$ or $y \in \{a, b\}^*$, and then that both x and y must contain the symbols $\$$ and $\#$. Note that $|x| = |y|$, for otherwise one could derive words v_1 and v_2 such that $|v_1| \neq |v_2|$, but T_n only contains words that have the same length.

Assume, without loss of generality, that $x \in \{a, b\}^*$ (the case that $y \in \{a, b\}^*$ is symmetric). From $x \neq y$ and $|x| = |y|$ it follows that both $x \neq \varepsilon$ and $y \neq \varepsilon$ must hold. Now, let $w \in \{a, b\}^n$; we distinguish three cases:

³Note that the results in [Fil11] are proved for the variants of C_n and T_n without the separator symbols $\$$ and $\#$.

1. Suppose that $u_1x \in \{a, b\}^*$ and $u_2 \in \{a, b\}^*\{\$\}\{w\}\{\#\}\{w\}$. Then

$$u_1xu_2 = v_1\$w\#w \quad \text{and} \quad u_1yu_2 = v_2\$w\#w,$$

for $v_1, v_2 \in \{a, b\}^n$ and $v_1 \neq v_2$. Thus, either $v_1 \neq w$ or $v_2 \neq w$, i.e., $v_1\$w\#w \notin T_n$ or $v_2\$w\#w \notin T_n$. This is a contradiction.

2. Suppose that $u_1 \in \{w\}\{\$\}\{a, b\}^*$ and $u_2 \in \{a, b\}^*\{\#\}\{w\}$. Then, since $x \neq y$, there are two derivations

$$S \Rightarrow_G^* u_1Au_2 \Rightarrow_G^* v = u_1xu_2 \quad \text{and} \quad S \Rightarrow_G^* u_1Au_2 \Rightarrow_G^* v' = u_1yu_2$$

with $v \neq v'$. In particular,

$$v = w\$u\#w \quad \text{and} \quad v' = w\$u'\#w$$

with $u \neq w$ or $u' \neq w$. But this means that $v \notin T_n$ or $v' \notin T_n$, which is a contradiction.

3. Suppose that $u_1 \in \{w\}\{\$\}\{w\}\{\#\}\{a, b\}^*$ and $xu_2 \in \{a, b\}^*$. Symmetric to case 1. Thus, we obtain a contradiction again.

Similar arguments show that either $w_1xw_2 \notin T_n$ or $w_1yw_2 \notin T_n$. Hence, we have both $x \notin \{a, b\}^*$ and $y \notin \{a, b\}^*$.

Now, suppose that x or y does not contain both $\$$ and $\#$. Assume, without loss of generality, that x contains $\#$ but does not contain $\$$. Let $w \in \{a, b\}^n$; we distinguish two cases:

1. Suppose that y contains $\$$. Then we have

$$S \Rightarrow_G^* u_1Au_2 \Rightarrow_G^* v = u_1xu_2 \quad \text{and} \quad S \Rightarrow_G^* u_1Au_2 \Rightarrow_G^* v' = u_1yu_2,$$

where either v does not contain $\$$ or v' contains at least two occurrences of $\$$. Thus, either $v \notin T_n$ or $v' \notin T_n$.

2. Suppose that y contains $\#$. Then we have

$$S \Rightarrow_G^* u_1Au_2 \Rightarrow_G^* v = u_1xu_2 \quad \text{and} \quad S \Rightarrow_G^* u_1Au_2 \Rightarrow_G^* v' = u_1yu_2.$$

Together with the fact that $x \neq y$ and $|x| = |y|$, it follows that x and y occupy the exact same positions within the words v and v' , respectively. As a consequence, if $v \in T_n$, then $v' \notin T_n$. If, on the other hand, $v' \in T_n$, then we have that $v \notin T_n$.

In both of these cases, we obtain a contradiction. Thus, both words x and y must contain both $\$$ and $\#$. This and $|x| = |y|$ implies that $x = y$. This is a contradiction to our assumption that $x \neq y$. Hence, $N = \{S\}$ and therefore the only way to generate the language T_n is to list all of its words as right-hand sides of productions having the sole nonterminal S on the left-hand side. In light of Equation 3.1, the statement also holds for the other grammar types in Γ . \square

Since the strict grammar types are less succinct than their respective non-strict variants w.r.t. the exact complexity, we get the following result as a simple corollary from Theorem 4.2.4.

Corollary 4.2.5. *Let $X \in \Gamma_s$ and $n \geq 1$ an integer. Then $Xc(T_n) \geq |T_n| = 2^n$.*

Another immediate consequence of the proof of Theorem 4.2.4 is the following exponential lower bound on the symbolic X -complexity of the triple language T_n . Note that our result on the language T_n is more precise than the one obtained from using the lower bound technique defined in [Fil11].

Corollary 4.2.6. *Let $X \in \Delta$ and $n \geq 1$ an integer. Then $Xsz(T_n) \geq 2^n \cdot (3n + 4)$.*

PROOF. A close inspection of the proof of Theorem 4.2.4 reveals that every Y -grammar, for $Y \in \Gamma$, solely consists of trivial productions. As a consequence, since, by Theorem 4.2.4,

$$Yc(T_n) = 2^n,$$

and every such trivial production consists of $(3n + 4)$ symbols, we get that

$$Ysz(T_n) = 2^n \cdot (3n + 4).$$

Since every strict regular and every strict linear grammar is a regular and a linear grammar, respectively, it follows that

$$\text{SREGsz}(T_n) \geq \text{REGsz}(T_n) = 2^n \cdot (3n + 4) \quad \text{and} \quad \text{SLINsz}(T_n) \geq \text{LINsz}(T_n) = 2^n \cdot (3n + 4).$$

This concludes the proof. \square

Remark. Note that, for any integer $n \geq 7$, the language T_n is already X cover-compressible for any type $X \in \Delta$. Since all words in T_n have length $3n + 2$, by Lemma 4.1.4, there exists already a strict regular grammar with $15n + 10 < 2^n = |T_n|$ productions that covers T_n . That is, for any integer $n \geq 7$, we have that $Xcc(T_n) < |T_n|$.

Later, in Chapter 7, we will use the language T_n as one of our basic building blocks for the language used in the proof of the inapproximability result of the exact complexity of finite languages. It is worth mentioning that the question as to whether the copy language C_n can only be generated minimally by a context-free grammar by simply listing all words is still open.

Open Problem 4.2.7. Let $X \in \Gamma$ and $n \geq 1$ an integer. Is the copy language C_n X -incompressible? \triangle

The following result, which has been shown in [BMCIW81, Lemma 2.2], is needed for the proof of the subsequent Corollary 4.2.10.

Lemma 4.2.8 ([BMCIW81, Lemma 2.2]). *Let $X \in \Delta$ and let $G = (N, \Sigma, P, S)$ be an X -grammar generating a finite language. Then there is an X -grammar $G_{\max} = (N_{\max}, \Sigma, P_{\max}, S)$*

such that

$$N_{\max} \subseteq N, P_{\max} \subseteq P, \text{ and } L(G_{\max}) = L_{\max},$$

where L_{\max} is the subset of $L(G)$ consisting of the words of maximal length.

Using similar arguments as in the proof of Theorem 4.2.4, one can show that any (strict) regular grammar generating the language

$$P_n = \{w\$w^R \mid w \in \{a, b\}^{\leq n}\}$$

of all even length palindromes (with middle marker) needs at least an exponential number of productions. This is done by first showing that the sublanguage

$$P'_n = \{w\$w^R \mid w \in \{a, b\}^n\}$$

of P_n is regular incompressible and then applying Lemma 4.2.8.

Theorem 4.2.9. *Let $X \in \{\text{SREG}, \text{REG}\}$ and $n \geq 1$ an integer. Then $Xc(P'_n) = 2^n$.*

PROOF. To this end, assume that $\Sigma = \{a, b, \$\}$ and that $G = (N, \Sigma, P, S)$ is a minimal regular grammar generating P'_n which contains a nonterminal $A \in N \setminus \{S\}$. By Lemma 4.2.1, there are derivations

$$S \Rightarrow_G^* u_1 A \quad \text{and} \quad S \Rightarrow_G^* u_2 A$$

with $u_1, u_2 \in \Sigma^*$ and $u_1 \neq u_2$ as well as

$$v_1, v_2 \in \Sigma^* \text{ with } A \Rightarrow_G^* v_1, A \Rightarrow_G^* v_2, \text{ and } v_1 \neq v_2.$$

Note that we must have both $|u_1| = |u_2|$ and $|v_1| = |v_2|$, for otherwise we would be able to derive words w_1 and w_2 with $|w_1| \neq |w_2|$, but P'_n only contains words of the same length. Since $v_1 \neq v_2$ and $|v_1| = |v_2|$, it follows that both $v_1 \neq \varepsilon$ and $v_2 \neq \varepsilon$. Let $w \in \{a, b\}^n$ be arbitrary. We distinguish the following two cases:

1. Suppose $u_1 \in \{w\$\}\{a, b\}^*$. Then we must have

$$u_1 = w_1 w_2 \$ w_2^R,$$

where $w = w_1 w_2$ and $v_1, v_2 \in \{a, b\}^*$. Assume, without loss of generality, that $v_1 = w_1^R$. Then, since $v_1 \neq v_2$, it follows that $v_2^R \neq w_1$. Thus, $u_1 v_2 \notin P'_n$. Contradiction.

2. Suppose $u_1 \in \{a, b\}^*$. Then we must have $v_1, v_2 \in \{a, b\}^* \{\$w^R\}$. Assume that $w = u_1 w_2$, for some $w_2 \in \{a, b\}^*$ and, without loss of generality, that

$$v_1 = w_2 \$ w_2^R u_1^R.$$

Since $v_1 \neq v_2$, it follows that $v_2 = w'_2 \$ w_2^R u_1^R$ with $w'_2 \neq w_2$. Thus, $u_1 v_2 \notin P'_n$. Contradiction.

Consequently, we have $N = \{S\}$ and so the only way to generate the language P'_n minimally with a regular grammar is to list all of its words using S . The result for strict regular grammars follows from Equation 3.1. \square

In light of Lemma 4.2.8 and since $P'_n \subseteq P_n$, the results of Theorem 4.2.9 imply that

$$\text{SREGc}(P_n) \geq 2^n \quad \text{and} \quad \text{REGc}(P_n) \geq 2^n.$$

We thus get the following corollary.

Corollary 4.2.10. *Let $X \in \{\text{SREG}, \text{REG}\}$ and $n \geq 1$ an integer. Then $Xc(P_n) \geq 2^n$.*

For (strict) linear and context-free grammars, one observes that the exact complexities $\text{SLINc}(P_n)$, $\text{LINc}(P_n)$, and $\text{CFc}(P_n)$ are at most linear, as witnessed by the strict linear context-free grammar $G = (N, \Sigma, P, S_0)$ with $N = \{S_0, S_1, \dots, S_n\}$, $\Sigma = \{a, b, \$\}$, start symbol S_0 , and where P consists of the following productions

$$\begin{aligned} S_i &\rightarrow aS_{i+1}a \mid bS_{i+1}b \mid S_{i+1} \quad \text{for } 0 \leq i \leq n-1, \\ S_n &\rightarrow \$. \end{aligned}$$

Clearly, G satisfies $L(G) = P_n$, for any integer $n \geq 1$. Note that the upper bound on the exact complexity of P_n immediately implies a corresponding upper bound on the cover complexity w.r.t. these grammar types. Consequently, we get the following lemma.

Lemma 4.2.11. *Let $X \in \{\text{SLIN}, \text{LIN}, \text{CF}\}$, $\tau \in \{c, cc\}$, and $n \geq 1$ an integer. Then $X\tau(P_n) \leq 3n + 1$.*

4.3 Lower Bounds on Cover Production Complexity

In this section, we are going to construct a regular cover-incompressible sequence of finite languages that is similar to, yet more general than, the one defined in [EH15a, EH18]. The need for a more general sequence is motivated by the fact that it allows us to show that the bound on the regular cover complexity of union is tight w.r.t. a fixed alphabet (see Section 6.2). Note that it is trivial to construct a cover-incompressible sequence of languages of constant size, e.g., $L_n = \{a\}$, for some letter a . It is also trivial to construct a sequence of cover-incompressible languages in an infinite alphabet, e.g.,

$$L_n = \{a_1, a_2, \dots, a_n\},$$

for letters a_1, a_2, \dots [EH18]. Consequently, in this section, we will construct a regular cover-incompressible sequence of languages of unbounded size over a finite alphabet.

This new sequence consists of so-called *segmented languages*, i.e., languages in which all words are repetitions of a *separator symbol* followed by a so-called *building block*. More formally, based on [EH18, Definition 9], this is defined as follows:

Definition 4.3.1 (Segmented Word/Language). Let Σ be an alphabet not containing the letter s . Then we write Σ_s for $\Sigma \cup \{s\}$. A word $w \in \Sigma_s^*$ such that

$$w = (sv)^k, \quad \text{for some integer } k \geq 1 \text{ and some } v \in \Sigma^+,$$

is called a *segmented word*. The word v and the letter s are called the *building block* and the *separator symbol*, respectively, of w . Occurrences of v in w are called *segments*. A segmented word $(sv)^k$ with $|v| = \ell$ is called a (k, ℓ) -segmented word. A language that solely consists of (k, ℓ) -segmented words is called a (k, ℓ) -segmented language.

Let Σ be an arbitrary alphabet not containing the letter s . For all integers $n \geq 1$, let $a_n \in \mathbb{N}$, let $\ell, k: \mathbb{N} \rightarrow \mathbb{N}$, and let $A_n \subseteq \Sigma^*$ such that

$$\begin{aligned} \ell(n) &\leq \lceil \log(a_n) \rceil, \\ k(n) &\geq \left\lceil \frac{9 \cdot a_n}{\ell(n) + 1} \right\rceil, \text{ and} \\ A_n &\subseteq \Sigma^{\ell(n)} \text{ with } |A_n| = a_n. \end{aligned}$$

Then, for each integer $n \geq 1$, we write $[\ell(n), k(n), A_n]$ for the language

$$\{(sw)^{k(n)} \mid w \in A_n\}.$$

Note that, for every integer $n \geq 1$, we have $|[\ell(n), k(n), A_n]| = |A_n| = a_n$, and all words in the language $[\ell(n), k(n), A_n]$ have the same length $k(n) \cdot (\ell(n) + 1)$, i.e., $[\ell(n), k(n), A_n]$ is a $(k(n), \ell(n))$ -segmented language for all integers $n \geq 1$. The number of segments has been chosen such that $k(n) \cdot (\ell(n) + 1)$ is $9 \cdot a_n$ padded up to the next multiple of $\ell(n) + 1$.

Remark. The above cover-incompressible sequence was obtained from the one constructed in [EH15a, EH18] by relaxing the constraints on $\ell(n)$ and $k(n)$ from “=” to “ \leq ” and “ \geq ”, respectively, and allowing arbitrary words of length $\ell(n)$ as building blocks for the segmented languages in the sequence.

The subsequent example demonstrates how one has to choose the parameters in order to obtain the regular cover-incompressible sequence constructed in [EH15a, EH18] from the above more general sequence.

Example 4.3.2. For an integer $n \geq 1$ and $k \in \{0, 1, \dots, 2^n - 1\}$, we write

$$b_n(k) \in \{0, 1\}^n$$

for the n -bit binary representation of k . Let, for all integers $n \geq 1$,

$$\begin{aligned} a_n &= n, \\ \ell(n) &= \lceil \log(a_n) \rceil, \\ k(n) &= \left\lceil \frac{9 \cdot a_n}{\ell(n) + 1} \right\rceil, \text{ and} \\ A_n &= \{b_{\ell(n)}(i) \mid 0 \leq i \leq n - 1\}. \end{aligned}$$

Note that we have both $|A_n| = a_n = n$ and $A_n \subseteq \{0, 1\}^{\ell(n)}$. As a consequence,

$$[\ell(n), k(n), A_n] = \{(sw)^{k(n)} \mid w \in A_n\} = \{(sb_{\ell(n)}(i))^{k(n)} \mid 0 \leq i \leq n-1\},$$

which is equal to the language L_n constructed in [EH18, Definition 14].

For instance, if $a_n = n = 5$, then we have $\ell(5) = 3$, $k(5) = 12$, $A_5 = \{000, 001, 010, 011, 100\}$, and $L_5 = \{(s000)^{12}, (s001)^{12}, (s010)^{12}, (s011)^{12}, (s100)^{12}\}$.

In the following definition, we introduce a notation for the set of nonterminals that is involved in a derivation of a specific segment of a given (k, ℓ) -segmented word.

Definition 4.3.3 ([EH18, Definition 10]). Let $G = (N, \Sigma, P, S)$ be a regular grammar and let $w \in L(G)$ be a (k, ℓ) -segmented word with building block v , i.e., $w = (sv)^k$. Moreover, let $i \in \{1, 2, \dots, k\}$, $w_0 = (sv)^{i-1}$, and $w_1 = (sv)^{k-i}$. Then $w = w_0svw_1$. Let δ be a derivation of w in G . Then δ is of the form

$$S \Rightarrow_G^* w'_0 A_1 \Rightarrow_G w_0 sv' A_2 \Rightarrow_G \dots \Rightarrow_G w_0 sv'' A_n \Rightarrow_G w_0 sv w'_1 A_{n+1} \Rightarrow_G^* w,$$

for some $A_1, A_2, \dots, A_{n+1} \in N$ with v' and v'' being prefixes of v , w'_0 being a prefix of w_0 , and w'_1 being a prefix of w_1 . Finally, we write $\text{nonterms}(w, i, \delta)$ for the set of nonterminals which is involved in the derivation of the i -th segment of w in δ , i.e.,

$$\text{nonterms}(w, i, \delta) = \{A_j \mid 1 \leq j \leq n\}.$$

Towards defining the notion of reduced grammar, recall that G_t denotes the grammar that is induced by omitting all *non-trivial* productions from a given grammar G .

Definition 4.3.4 (Reduced Regular Grammar, [EH18, Definition 7]). Let L be a finite language and G be a regular grammar covering L . Then G is called *reduced w.r.t. L* if, for every non-trivial production $A \rightarrow wB$ or $A \rightarrow w$ of G with $w \in \Sigma^*$, there are distinct words $u, v \in L \setminus L(G_t)$ such that w is a subword of both u and v .

In order to express that a nonterminal is derivable from another one in a context-free grammar, we introduce the following binary relation on the set of nonterminals.

Definition 4.3.5 ([EH18, Definition 3]). Let $G = (N, \Sigma, P, S)$ be a context-free grammar. The relation $<_G^1$ on the set of nonterminals N is defined as follows:

$$A <_G^1 B \quad \text{if and only if} \quad \text{there is a production } A \rightarrow \alpha \in P \text{ such that } B \text{ occurs in } \alpha.$$

The relation $<_G$ is defined as the transitive closure of $<_G^1$.

In an acyclic regular grammar G , the ordering $<_G$ is, in general, not linear. For technical purposes, it will be useful to fix a linearisation of $<_G$ and a corresponding linear order of the productions of G . To this aim, we introduce the notion of ordered grammars [EH18]. Recall that, for a grammar $G = (N, \Sigma, P, S)$ and $A \in N$, we write P_A for the set of A -productions, that is, $P_A = \{A \rightarrow \alpha \mid A \rightarrow \alpha \in P\}$.

Definition 4.3.6 ([EH18, Definition 11]). An *ordered regular grammar* is a quadruple $G = (N, \Sigma, P, A_1)$, where N is a finite sequence (A_1, A_2, \dots, A_n) of nonterminals and P is a finite sequence (p_1, p_2, \dots, p_m) of productions such that

1. $G' = (\{A_1, A_2, \dots, A_n\}, \Sigma, \{p_1, p_2, \dots, p_m\}, A_1)$ is a regular grammar,
2. if $A_i <_{G'} A_j$, then $i < j$, and
3. the productions p_1, p_2, \dots, p_m are grouped by their left-hand sides:

$$p_1, p_2, \dots, p_m = q_{1,1}, q_{1,2} \dots q_{1,k_1}, q_{2,1}, q_{2,2} \dots q_{2,k_2} \dots, q_{n,1}, q_{n,2} \dots q_{n,k_n},$$

where $\{q_{i,1}, q_{i,2}, \dots, q_{i,k_i}\} = P_{A_i}$, for all $i \in \{1, 2, \dots, n\}$.

We say that an ordered regular grammar cover-compresses a finite language L , is reduced w.r.t. L , etc. if the underlying regular grammar satisfies the respective property [EH18].

Definition 4.3.7 ([EH18, Definition 13]). Let $G = (N, \Sigma, (p_1, p_2, \dots, p_m), A_1)$ be an ordered regular grammar and let $s < m$. For $A \in N$, define

$$\text{pmin}(A) = \min\{j \mid p_j \in P_A\} \quad \text{and} \quad \text{pmax}(A) = \max\{j \mid p_j \in P_A\}.$$

Furthermore, for $j \in \{1, 2, \dots, \lceil \frac{m}{s} \rceil - 1\}$, define

$$N_j = \{A \in N \mid (j-1) \cdot s \leq \text{pmin}(A) \text{ and } \text{pmax}(A) < (j+1) \cdot s\}.$$

We say that $(N_j)_{j=1}^{\lceil \frac{m}{s} \rceil - 1}$ is the s -covering of G .

Remark. Note that N_j and N_{j+1} can overlap, but N_j and N_{j+2} cannot. Furthermore, it holds that

$$|P_{N_j}| \leq 2s,$$

for all $j \in \{1, 2, \dots, \lceil \frac{m}{s} \rceil - 1\}$ [EH18].

The following results are key ingredients for the proof of the regular cover-incompressibility result and their proofs can be found in [EH18].

Lemma 4.3.8 ([EH18, Lemma 5]). Let $(L_n)_{n \geq 1}$ be a regular cover-compressible sequence of finite languages such that L_n is (k_n, ℓ_n) -segmented and $(k_n)_{n \geq 1}$ is unbounded. Then there is a sequence of finite languages $(L'_n)_{n \geq 1}$ such that

1. $L'_n \subseteq L_n$, for all integers $n \geq 1$,
2. $(L'_n)_{n \geq 1}$ is cover-compressible by a reduced acyclic regular grammar without trivial productions, and
3. $(|L'_n|)_{n \geq 1}$ is unbounded.

Lemma 4.3.9 ([EH18, Lemma 6]). *Let L be a finite (k, ℓ) -segmented language that is cover-compressed by a reduced acyclic regular grammar $G = (N, \Sigma, P, S)$ without trivial productions. For each $w \in L$, fix a G -derivation δ_w of w . Let $N_0 \subseteq N$, let $P_0 = P_{N_0}$, and let*

$$S_0 = \{(w, i) \in L \times \{1, 2, \dots, k\} \mid \text{nonterms}(w, i, \delta_w) \subseteq N_0\}.$$

Then we have $|S_0| \leq 2^{|P_0|} \cdot |P_0|$.

Lemma 4.3.10 ([EH18, Lemma 8]). *Let L be a finite (k, ℓ) -segmented language, let $G = (N, \Sigma, P, S)$ be an ordered regular grammar without trivial productions that cover-compresses the language L , and let $|G| > s \geq \frac{4|L|}{k}$. Moreover, let $N_1, N_2, \dots, N_{\lceil \frac{|G|}{s} \rceil - 1}$ be the s -covering of G , let $w \in L$, and let δ be a G -derivation of w . Then, for at least half of the $i \in \{1, 2, \dots, k\}$, there is a $j \in \{1, 2, \dots, \lceil \frac{|G|}{s} \rceil - 1\}$ such that $\text{nonterms}(w, i, \delta) \subseteq N_j$.*

The proof of the following Theorem 4.3.11 can be obtained by a slight modification of the proof of [EH18, Theorem 1], but, nevertheless, we include the full proof for the sake of completeness. We can summarise the proof strategy as follows: both Lemmas 4.3.9 and 4.3.10 assume a segmented language that is cover-compressed by a reduced grammar without trivial productions. While Lemma 4.3.9 states an upper bound on the number of segments covered by a certain part of a reduced cover-compressing grammar without trivial rules, Lemma 4.3.10 shows a lower bound on the number of segments covered by the productions of a single N_j . The following proof will show that these two bounds are contradictory and thus derive the cover-incompressibility of $([\ell(n), k(n), A_n])_{n \geq 1}$.

Theorem 4.3.11. *Any sequence $([\ell(n), k(n), A_n])_{n \geq 1}$ is regular cover-incompressible.*

PROOF. We fix a sequence $([\ell(n), k(n), A_n])_{n \geq 1}$ and abbreviate $[\ell(n), k(n), A_n]$ by L_n in the remainder of the proof. Suppose that $(L_n)_{n \geq 1}$ is a regular cover-compressible sequence of finite languages. Then, by Lemma 4.3.8, there is a sequence $(L'_n)_{n \geq 1}$ which satisfies $L'_n \subseteq L_n$, for all integers $n \geq 1$, and is regular cover-compressed by a sequence $(G_n)_{n \geq 1}$ of reduced acyclic regular grammars without trivial productions. We consider G_n as an ordered regular grammar G'_n by fixing an arbitrary linear order satisfying Definition 4.3.6. Let us fix, for every integer $n \geq 1$ and every $w \in L'_n$, a derivation $\delta_{n,w}$ of w w.r.t. G'_n .

First, note that, for all integers $n \geq 1$, we have

$$k(n) \geq \left\lceil \frac{9 \cdot a_n}{\ell(n) + 1} \right\rceil \geq \left\lceil \frac{9 \cdot a_n}{\lceil \log(a_n) \rceil + 1} \right\rceil = \left\lceil \frac{9 \cdot |A_n|}{\lceil \log(|A_n|) \rceil + 1} \right\rceil \geq \frac{9 \cdot |A_n|}{\log(|A_n|) + 2},$$

and since $|A_n| = |L_n| \geq |L'_n|$, we have

$$k(n) \geq \frac{9 \cdot |L'_n|}{\log(|L'_n|) + 2}. \quad (4.4)$$

Therefore, $s_n := \frac{4}{9} \cdot (\log(|L'_n|) + 2) \geq \frac{4|L'_n|}{k(n)}$. Let $N_1, N_2, \dots, N_{\lceil \frac{|G'_n|}{s_n} \rceil - 1}$ be the s_n -covering of G'_n and define

$$U_n := |\{(w, i) \in L'_n \times \{1, 2, \dots, k(n)\} \mid \text{there exists a } j \text{ such that } \text{nonterms}(w, i, \delta_{n,w}) \subseteq N_j\}|.$$

Note that there is some integer n_0 such that for all integers $n \geq n_0$,

$$|G'_n| > s_n,$$

which can be seen as follows: by Lemma 3.3.1, we have that

$$|L(G'_n)| \leq 2^{|G'_n|-1} \leq 2^{|G'_n|}$$

and thus

$$|L'_n| \leq |L(G'_n)| \leq 2^{|G'_n|}.$$

Consequently,

$$4 \cdot \log(|L'_n|) \leq 4 \cdot |G'_n|.$$

This implies that there is some integer n_0 such that for all integers $n \geq n_0$, we have that

$$4 \cdot \log(|L'_n|) + 8 < 9 \cdot |G'_n|,$$

which is equivalent to the existence of some integer n_0 such that for all integers $n \geq n_0$, we have that

$$s_n = \frac{4}{9} \cdot (\log(|L'_n|) + 2) < |G'_n|.$$

By Lemma 4.3.10, we have $U_n \geq \frac{|L'_n| \cdot k(n)}{2}$, which, together with (4.4), entails

$$U_n \geq \frac{9 \cdot |L'_n|^2}{2 \cdot (\log(|L'_n|) + 2)}. \quad (4.5)$$

On the other hand, applying Lemma 4.3.9 to all N_j , for $j = 1, 2, \dots, \left\lceil \frac{|G'_n|}{s_n} \right\rceil - 1$, and summing up yields

$$U_n \leq \sum_{j=1}^{\left\lceil \frac{|G'_n|}{s_n} \right\rceil - 1} 2^{|P_{N_j}|} \cdot |P_{N_j}| \leq \left(\left\lceil \frac{|G'_n|}{s_n} \right\rceil - 1 \right) \cdot 2^{2s_n} \cdot 2s_n,$$

since $|P_{N_j}| \leq 2s_n$, for all $j \in \{1, 2, \dots, \left\lceil \frac{|G'_n|}{s_n} \right\rceil - 1\}$. We have that

$$2^{2s_n} \cdot 2s_n \leq C \cdot |L'_n|^{\frac{8}{9}} \cdot (\log(|L'_n|) + 2), \text{ for some } C \in \mathbb{N}$$

and

$$\left\lceil \frac{|G'_n|}{s_n} \right\rceil - 1 \leq \frac{|L'_n|}{s_n} = \frac{9 \cdot |L'_n|}{4 \cdot (\log(|L'_n|) + 2)}$$

and therefore

$$U_n \leq D \cdot |L'_n|^{\frac{17}{9}}, \text{ for some } D \in \mathbb{N}. \quad (4.6)$$

Putting (4.5) and (4.6) together, we obtain

$$\frac{9 \cdot |L'_n|^2}{2 \cdot (\log(|L'_n|) + 2)} \leq U_n \leq D \cdot |L'_n|^{\frac{17}{9}}.$$

Therefore,

$$|L'_n|^2 \leq E \cdot |L'_n|^{\frac{17}{9}} \cdot (\log(|L'_n|) + 2), \text{ for some } E \in \mathbb{N}. \quad (4.7)$$

But, by Lemma 4.3.8, the function $n \mapsto |L'_n|$ is unbounded. Hence, there is an $M \in \mathbb{N}$ such that for all $n \geq M$, the inequality (4.7) is not satisfied. Contradiction. Therefore, the sequence $(L_n)_{n \geq 1}$ is regular cover-incompressible. This finishes the proof. \square

Relating Finite and Infinite Complexity Measures



WE will consider several different production complexity measures for finite languages w.r.t. different interpretations of approximation (i.e., equivalence, cover, and scattered cover), where the underlying grammar either generates a finite or an infinite language. In the case that the generated language is infinite, the intersection with all words up to a certain length has to be considered in order to obtain the finite language under consideration. These measures will then be related according to a group of relations that are inspired by the taxonomy w.r.t. nonterminal complexity which was introduced in [DP89]. By the very nature of these relations, one can distinguish two main categories:

1. the first category fixes a measure type τ and then compares the different grammar types in Δ with each other w.r.t. the measure type τ .
2. the second category swaps the roles of measure and grammar type, that is, some grammar type $X \in \Delta$ is fixed and then the different measure types under consideration are compared with each other w.r.t. the grammar type X .

The above mentioned relative succinctness classification w.r.t. nonterminal complexity measures, which was introduced in the book *Regulated Rewriting in Formal Language Theory* by Dassow and Păun [DP89], is based on the following four relations defined on a set consisting of the classes¹ of context-free grammars as well as of the classes of matrix, programmed, and random context-free grammars:

Let X and Y be two distinct types of grammars, let \mathcal{L} be the intersection of the classes of languages generated by X - and Y -grammars, and let

$$X_N(L) = \min\{|N| \mid G = (N, \Sigma, P, S) \in X \text{ and } L(G) = L\}.$$

¹Note that here we do not mean class of context-free grammars in the sense of Definition 3.3.3.

Then we write

- $X \leq_N Y$ if and only if there is a constant c such that

$$|X_N(L) - Y_N(L)| \leq c,$$

for all languages $L \in \mathcal{L}$;

- $X \leq_N^1 Y$ if and only if there is a constant c such that

$$X_N(L) \leq Y_N(L) + c,$$

for all languages $L \in \mathcal{L}$, and there is a sequence $(L_i)_{i \geq 0}$ of languages in \mathcal{L} such that

$$Y_N(L_i) - X_N(L_i) \geq i;$$

- $X \leq_N^2 Y$ if and only if there is a constant c such that

$$X_N(L) \leq Y_N(L) + c,$$

for all languages $L \in \mathcal{L}$, and there is a sequence $(L_i)_{i \geq 0}$ of languages in \mathcal{L} such that

$$\lim_{i \rightarrow \infty} \frac{X_N(L_i)}{Y_N(L_i)} = 0;$$

- $X \leq_N^3 Y$ if and only if there is a constant c such that

$$X_N(L) \leq Y_N(L) + c,$$

for all languages $L \in \mathcal{L}$, and there is no function $f: \mathbb{N} \rightarrow \mathbb{N}$ such that

$$Y_N(L) \leq f(X_N(L)),$$

for all languages $L \in \mathcal{L}$.

Note that the subscript “ N ” in X_N refers to the fact that X_N is a nonterminal complexity measure.

It was then shown that both matrix and programmed grammars with context-free core productions (with or without ε -productions) and with or without appearance checking are more succinct than context-free grammars w.r.t. the relation \leq_N^3 . Moreover, random context-free grammars with context-free core productions (with or without ε -productions) with appearance checking are more succinct than context-free grammars w.r.t. the relation \leq_N^3 , but if appearance checking is dropped, then they are only more succinct w.r.t. the relation \leq_N^2 . Both matrix and programmed grammars with context-free core productions are more succinct than random context-free grammars with context-free core productions w.r.t. the relation \leq_N^3 . Another result is that matrix and programmed grammars with context-free core productions are equally succinct, that is, the relation \leq_N holds in both directions. Finally, it was shown that random context-free grammars with context-free core productions and appearance checking are more succinct than the variant without ε -productions w.r.t. the relation \leq_N^2 .

Some of the results in this chapter have been published in [HW18b].

5.1 Infinite Complexity Measures

In this section, we will introduce the infinite versions of the exact, cover, and scattered cover complexity measures X_c , X_{cc} , and X_{sc} , for $X \in \Delta$, and prove some upper bounds on these infinite production complexity measures.

One of the inspirations for the subsequent definitions of the infinite production complexity measures is the way the accepted language of a finite cover automaton is defined.² The fact that finite languages can be represented by both ordinary (deterministic) finite automata and cover finite automata, gives rise to two different complexity measures for finite languages: given a finite language $L \subseteq \Sigma^{\leq \ell}$ with $\ell = \max\{|w| \mid w \in L\}$, one can define the DFA- and the DFCA-state complexity of L as

$$\text{DFAC}(L) = \min\{|\mathcal{A}| \mid \mathcal{A} \text{ is a DFA and } L = L(\mathcal{A})\}$$

and

$$\text{DFAC}_{\infty}(L) = \min\{|\mathcal{A}| \mid \mathcal{A} \text{ is a DFA and } L = L(\mathcal{A}) \cap \Sigma^{\leq \ell}\},$$

respectively, where $|\mathcal{A}|$ denotes the number of states of the automaton \mathcal{A} . Obviously, it holds that

$$\text{DFAC}_{\infty}(L) \leq \text{DFAC}(L), \quad (5.1)$$

for every finite language L . It is worth mentioning that although the definitions of these measures are pretty similar, their values can possibly differ tremendously when applied to the same language. For instance, there is a finite language L , namely $\Sigma^{\leq n}$, over an arbitrary alphabet Σ , which satisfies, for every integer $n \geq 1$, that

$$\text{DFAC}_{\infty}(L) = 1 \quad \text{and} \quad \text{DFAC}(L) = n + 1,$$

since

$$\Sigma^{\leq n} = \Sigma^* \cap \Sigma^{\leq n}.$$

See Figures 51 and 52 for finite automata \mathcal{A}_1 and \mathcal{A}_2 satisfying

$$\Sigma^{\leq n} = L(\mathcal{A}_1) \cap \Sigma^{\leq n} \quad \text{and} \quad L(\mathcal{A}_2) = \Sigma^{\leq n},$$

respectively. Hence, the gap between these two measures can be arbitrarily large.

We will now adapt the definition of the DFCA-state complexity in order to introduce the infinite exact complexity for grammars of type $X \in \Delta$ and, in addition, introduce the infinite X cover as well as the infinite X scattered complexity measures for grammars of type $X \in \Delta$.

Definition 5.1.1 (Infinite X -Complexity Measures). Let $L \subseteq \Sigma^{\leq \ell}$ be a finite language and $X \in \Delta$. Then the *infinite X -complexity* of L is defined as

$$Xc_{\infty}(L) = \min\{|G| \mid G \in X \text{ and } L = L(G) \cap \Sigma^{\leq \ell}\}.$$

²Recall that a finite cover automaton for a finite language L whose longest word is of length ℓ is a finite automaton \mathcal{A} satisfying $L = L(\mathcal{A}) \cap \Sigma^{\leq \ell}$ (see, e.g., [CSY01] or Definitions 2.2.4 and 2.2.5).

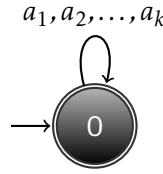


Figure 51: DFCA \mathcal{A}_1 for the finite language $\{a_1, a_2, \dots, a_k\}^{\leq n}$.

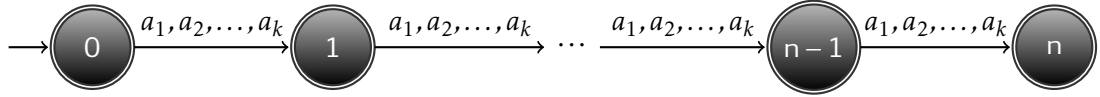


Figure 52: Minimal DFA \mathcal{A}_2 accepting the finite language $\{a_1, a_2, \dots, a_k\}^{\leq n}$.

Similarly, the *infinite X cover complexity* and the *infinite X scattered complexity* of L are defined as

$$X_{cc_\infty}(L) = \min\{|G| \mid G \in X \text{ and } L \subseteq L(G) \cap \Sigma^{\leq \ell}\}$$

and

$$X_{sc_\infty}(L) = \min\{|G| \mid G \in X \text{ and } L \leq L(G) \cap \Sigma^{\leq \ell}\},$$

respectively.

Remark. Note that in the definitions of X_{c_∞} , X_{cc_∞} , and X_{sc_∞} , the grammar G is allowed to generate an *infinite* language. Therefore, they are called infinite complexity measures.

For $X \in \Delta$ and $G \in X$, we say that G is a minimal X -grammar infinitely generating a finite language L if $L = L(G) \cap \Sigma^{\leq \ell}$ and $|G| = X_{c_\infty}(L)$. The notions of minimal X -grammar infinitely covering or infinite scatteredly covering a finite language are defined analogously.

It is worth mentioning that, for $X \in \Delta$, the relation between the measures X_c and X_{c_∞} is analogous to that between DFA_c and DFA_{c_∞} in the sense that, similarly to Equation 5.1, it holds that

$$X_{c_\infty}(L) \leq X_c(L), \quad (5.2)$$

for all finite languages L . This can be shown as follows: for some $\ell \geq 0$, let $L \subseteq \Sigma^{\leq \ell}$ and assume that the X -grammar G is a witness for $X_c(L)$, i.e., G generates the finite language L and $X_c(L) = |G|$. But then we also have that

$$L = L \cap \Sigma^{\leq \ell} = L(G) \cap \Sigma^{\leq \ell},$$

which implies $X_{c_\infty}(L) \leq X_c(L)$.

Moreover, the gap between X_c and X_{c_∞} can also be arbitrarily large, which can be seen as follows: for $X = CF$, consider the languages

$$R_{2^n} = \{a^k b a^m \mid 0 \leq k + m \leq 2^n - 1\} \quad \text{and} \quad R = \{a^k b a^m \mid k, m \geq 0\}.$$

In the proof of [BMCIW81, Theorem 1], it was shown that

$$\text{CFc}(R_{2^n}) \geq n + 1.$$

Moreover, the infinite language R can be generated with the following context-free grammar

$$G_R = (\{S\}, \{a, b\}, \{S \rightarrow aS, S \rightarrow bA, A \rightarrow aA, A \rightarrow \varepsilon\}, S).$$

Thus,

$$R_{2^n} = L(G_R) \cap \{a, b\}^{\leq 2^n},$$

and so

$$\text{CFc}_\infty(R_{2^n}) \leq 4.$$

For $X \in \Delta \setminus \{\text{CF}\}$ and any integer $n \geq 1$, consider the language

$$K_n = \{a, b\}^{\leq n}.$$

Also, in the proof of [BMCIW81, Theorem 1], it was shown that

$$Xc(K_n) \geq n + 1.$$

Moreover, the language K_n can be generated infinitely with the following strict regular grammar

$$G = (\{S\}, \{a, b\}, \{S \rightarrow aS, S \rightarrow bS, S \rightarrow \varepsilon\}, S),$$

since

$$K_n = \{a, b\}^* \cap \{a, b\}^{\leq n}.$$

As a consequence,

$$Xc_\infty(K_n) \leq 3.$$

Hence, for each $X \in \Delta$ and each $n \geq 1$, there is a finite language L_n such that we have that

$$Xc(L_n) \geq n + 1 \quad \text{and} \quad Xc_\infty(L_n) \leq 4, \quad (5.3)$$

that is, there can be an arbitrarily large gap between Xc and Xc_∞ .

Next, we introduce the sets \mathcal{M}_{fin} and \mathcal{M}_∞ of finite and infinite measure types, respectively, i.e.,

$$\mathcal{M}_{\text{fin}} = \{c, cc, sc\} \quad \text{and} \quad \mathcal{M}_\infty = \{c_\infty, cc_\infty, sc_\infty\}.$$

Moreover, we write \mathcal{M} for the set of (all) measure types, that is,

$$\mathcal{M} = \mathcal{M}_{\text{fin}} \cup \mathcal{M}_\infty.$$

Let $\tau \in \mathcal{M}$ and $X, Y \in \Delta$. Then we define

$$X \leq_\tau Y \quad \text{if and only if} \quad X\tau(L) \leq Y\tau(L), \text{ for all finite languages } L.$$

In the case that $X \leq_\tau Y$, we say that X is *more succinct* than Y w.r.t. the measure type τ . By definition, the following relations hold for each $\tau \in \mathcal{M}$:

$$CF \leq_\tau LIN \leq_\tau REG \leq_\tau SREG \quad \text{and} \quad CF \leq_\tau LIN \leq_\tau SLIN \leq_\tau SREG. \quad (5.4)$$

Before we delve into comparing different complexity measures with each other, let us first prove the following basic lemma which states some upper bounds on the newly introduced infinite complexity measures. These results will be used frequently throughout this chapter.

Lemma 5.1.2. *Let $L \subseteq \Sigma^{\leq \ell}$ be a finite language over $\Sigma = \{a_1, a_2, \dots, a_n\}$ and $X \in \Delta$. Then*

1. $X_{C_\infty}(\Sigma^{\leq \ell}) \leq |\Sigma| + 1$,
2. $X_{CC_\infty}(L) \leq |\Sigma| + 1$, and
3. $X_{SC_\infty}(L) \leq |\Sigma| + 1$.

PROOF. We first show claim 1. Consider the strict regular grammar $G = (\{S\}, \Sigma, P, S)$ with the following set of productions P :

$$S \rightarrow a_1 S \mid a_2 S \mid \dots \mid a_n S \mid \varepsilon.$$

It is easy to see that G generates the universal language Σ^* , i.e., $L(G) = \Sigma^*$. Since we have

$$\Sigma^* \cap \Sigma^{\leq \ell} = \Sigma^{\leq \ell},$$

it follows that G infinitely generates $\Sigma^{\leq \ell}$ with $|\Sigma| + 1$ productions, i.e.,

$$SREG_{C_\infty}(\Sigma^{\leq \ell}) \leq |G| = |\Sigma| + 1.$$

From Equation 5.4, it follows that

$$Y_{C_\infty}(\Sigma^{\leq \ell}) \leq |\Sigma| + 1.$$

also holds for the remaining grammar types $Y \in \Delta \setminus \{SREG\}$. This finishes the proof of claim 1.

Claims 2. and 3. immediately follow from 1., since, for every finite language $L \subseteq \Sigma^{\leq \ell}$, we have that

$$\Sigma^{\leq \ell} = \Sigma^* \cap \Sigma^{\leq \ell} \geq L,$$

for $\geq \in \{\supseteq, \geq\}$. □

Remark. Let $X \in \Delta$. Then, by Lemma 5.1.2, the finite complexity measure X_{SC} as well as the infinite complexity measures X_{CC_∞} and X_{SC_∞} are bounded complexity measures in the sense of Chapter 3. Nevertheless, for the sake of completeness, we will also include these bounded measures in our relative succinctness classification.

5.2 Relating Grammar Types

This section is devoted to relating the grammar types in Δ w.r.t. a fixed measure type from \mathcal{M} based on four different relations that vary in strength. In this way, we can classify the difference between different grammar types w.r.t. to a fixed complexity measure type. As already mentioned in the previous section, these relations are similar to those introduced in [DP89] for nonterminal complexity measures. The main difference—apart from replacing the number of nonterminals by the number of productions—is that we consider measures for finite languages only. As a byproduct of this classification, we also show that any regular grammar that infinitely generates the language

$$P_n = \{ w\$w^R \mid w \in \{a, b\}^{\leq n} \}$$

requires $\Omega(2^n)$ many productions.

We start by formally defining the above mentioned relations on grammar types in Δ .

Definition 5.2.1. Let $X, Y \in \Delta$ and $\tau \in \mathcal{M}$. Then we write

- $X \leq_\tau Y$ if and only if

$$X\tau(L) \leq Y\tau(L),$$

for all finite languages L ;

- $X \leq_\tau^1 Y$ if and only if there is a constant c such that

$$X\tau(L) \leq Y\tau(L) + c,$$

for all finite languages L , and there is a sequence of finite languages $(L_i)_{i \geq 0}$ such that

$$Y\tau(L_i) - X\tau(L_i) \geq i;$$

- $X \leq_\tau^2 Y$ if and only if there is a constant c such that

$$X\tau(L) \leq Y\tau(L) + c,$$

for all finite languages L , and there is a sequence of finite languages $(L_i)_{i \geq 0}$ such that

$$\lim_{i \rightarrow \infty} \frac{X\tau(L_i)}{Y\tau(L_i)} = 0;$$

- $X \leq_\tau^3 Y$ if and only if there is a constant c such that

$$X\tau(L) \leq Y\tau(L) + c,$$

for all finite languages L , and there is no function $f: \mathbb{N} \rightarrow \mathbb{N}$ such that

$$Y\tau(L) \leq f(X\tau(L)),$$

for all finite languages L .

We write $X =_\tau Y$ if both $X \leq_\tau Y$ and $Y \leq_\tau X$ hold. Moreover, for ease of presentation, we sometimes denote the relation \leq_τ by \leq_τ^0 . Let $i \in \{0, 1, 2, 3\}$ and $\tau \in \mathcal{M}$, then the relation \leq_τ^i is said to be of *order* i .

Let $\Pi_1, \Pi_2 \subseteq \Delta$, $\mathcal{N} \subseteq \mathcal{M}$, and $i \in \{0, 1, 2, 3\}$. Then we write

$$\Pi_1 \leq_{\mathcal{N}}^i \Pi_2$$

if, for all $X \in \Pi_1$, all $Y \in \Pi_2$, and all $\tau \in \mathcal{N}$, it holds that $X \leq_\tau^i Y$. Similarly, we write

$$\Pi_1 \not\leq_{\mathcal{N}}^i \Pi_2$$

if, for all $X \in \Pi_1$, all $Y \in \Pi_2$, and all $\tau \in \mathcal{N}$, it holds that $X \not\leq_\tau^i Y$. In the case that Π_1 , Π_2 , or \mathcal{N} is a singleton, we omit braces, e.g., we write $X \leq_\tau^i Y$ instead of $\{X\} \leq_{\{\tau\}}^i \{Y\}$.

Remark. Note that, for $i \in \{0, 1, 2, 3\}$,

$$\Pi_1 \not\leq_{\mathcal{N}}^i \Pi_2$$

being satisfied does not coincide with the fact that

$$\Pi_1 \leq_{\mathcal{N}}^i \Pi_2$$

does not hold. The latter does not hold as soon as we have that

$$X \not\leq_\tau^i Y$$

for at least one $X \in \Pi_1$, at least one $Y \in \Pi_2$, or at least one $\tau \in \mathcal{N}$. On the other hand,

$$\Pi_1 \not\leq_{\mathcal{N}}^i \Pi_2$$

is only satisfied if

$$X \not\leq_\tau^i Y$$

holds for all $X \in \Pi_1$, all $Y \in \Pi_2$, and all $\tau \in \mathcal{N}$.

The following result from [BMCIW81] will be used frequently throughout this chapter.

Lemma 5.2.2 ([BMCIW81, Corollary 2.1]). *Let G be a linear grammar with $n \geq 1$ productions generating a finite language. Then $n \geq \log(|L(G)|) + 1$.*

In order to demonstrate that—for distinct grammar types—the production complexity of the same finite language w.r.t. the same measure type can vary quite immensely, consider, for instance, the language

$$L_n = \{a, b\}^{\leq n},$$

where $n \geq 1$ is an integer. On the one hand, Theorem 4.1.5 shows that

$$\text{CFc}(L_n) \leq 4, \quad \text{for each } n \geq 1,$$

but, on the other hand, from Lemma 5.2.2, it follows that

$$\text{LINc}(L_n) \geq \log(2^{n+1} - 1) + 1 \geq n + 1.$$

In conjunction with Equation 5.4, this clearly implies that we have a gap of highest order (in the sense of Definition 5.2.1) between context-free and linear grammars w.r.t. the exact complexity, i.e.,

$$\text{CF} \leq_c^3 \text{LIN}$$

holds.

If one takes a closer look at the definitions of the relations \leq_τ and \leq_N , one observes that the definition of the former is a bit more restrictive than the latter, as we have dropped the constant c . The reason for this is that the definition without the constant is compatible with the notion of a grammar type being more succinct than another grammar type w.r.t. a measure type (see Equation 5.4).

Remark. Clearly, $X \leq_\tau^3 Y$ implies $X \leq_\tau^2 Y$, which, in turn, implies $X \leq_\tau^1 Y$. Moreover, the inequality $X \leq_\tau^3 Y$ holds if the first condition of its definition is satisfied and there is a sequence of finite languages $(L_i)_{i \geq 0}$ such that $X\tau(L_i) \leq k$, for some constant k , and $Y\tau(L_i) \geq i$.

Let $X, Y \in \Delta$ and $\tau \in \mathcal{M}$. Then we say that X and Y are *incomparable* w.r.t. the relation \leq_τ^i , for $i \in \{0, 1, 2, 3\}$, if we have both $X \not\leq_\tau^i Y$ and $Y \not\leq_\tau^i X$. Moreover, we say that X and Y are *incomparable* w.r.t. the complexity measure τ if we have both $X \not\leq_\tau^i Y$ and $Y \not\leq_\tau^i X$, for each $i \in \{0, 1, 2, 3\}$.

Figure 53 depicts which kinds of relations between the grammar types in Δ hold w.r.t. the measure types in \mathcal{M} . Let $X, Y \in \Delta$ and $i_c, i_{cc}, i_{sc}, i_{c_\infty}, i_{cc_\infty}, i_{sc_\infty}, j_c, j_{cc}, j_{sc}, j_{c_\infty}, j_{cc_\infty}, j_{sc_\infty} \in \{0, 1, 2, 3\}$. Then a directed edge with label $(i_c/j_c, i_{cc}/j_{cc}, i_{sc}/j_{sc}, i_{c_\infty}/j_{c_\infty}, i_{cc_\infty}/j_{cc_\infty}, i_{sc_\infty}/j_{sc_\infty})$ from X to Y , i.e.,

$$X \xrightarrow{(i_c/j_c, i_{cc}/j_{cc}, i_{sc}/j_{sc}, i_{c_\infty}/j_{c_\infty}, i_{cc_\infty}/j_{cc_\infty}, i_{sc_\infty}/j_{sc_\infty})} Y$$

expresses that all of the relations

$$X \leq_c^{i_c} Y, \quad X \leq_{cc}^{i_{cc}} Y, \quad X \leq_{sc}^{i_{sc}} Y, \quad X \leq_{c_\infty}^{i_{c_\infty}} Y, \quad X \leq_{cc_\infty}^{i_{cc_\infty}} Y, \quad X \leq_{sc_\infty}^{i_{sc_\infty}} Y$$

and

$$X \not\leq_c^{i'_c} Y, \quad X \not\leq_{cc}^{i'_{cc}} Y, \quad X \not\leq_{sc}^{i'_{sc}} Y, \quad X \not\leq_{c_\infty}^{i'_{c_\infty}} Y, \quad X \not\leq_{cc_\infty}^{i'_{cc_\infty}} Y, \quad X \not\leq_{sc_\infty}^{i'_{sc_\infty}} Y$$

as well as

$$Y \leq_c^{j_c} X, \quad Y \leq_{cc}^{j_{cc}} X, \quad Y \leq_{sc}^{j_{sc}} X, \quad Y \leq_{c_\infty}^{j_{c_\infty}} X, \quad Y \leq_{cc_\infty}^{j_{cc_\infty}} X, \quad \text{and} \quad Y \leq_{sc_\infty}^{j_{sc_\infty}} X$$

and

$$Y \not\leq_c^{j'_c} X, \quad Y \not\leq_{cc}^{j'_{cc}} X, \quad Y \not\leq_{sc}^{j'_{sc}} X, \quad Y \not\leq_{c_\infty}^{j'_{c_\infty}} X, \quad Y \not\leq_{cc_\infty}^{j'_{cc_\infty}} X, \quad \text{and} \quad Y \not\leq_{sc_\infty}^{j'_{sc_\infty}} X,$$

for all $i'_\tau \in \{i_\tau + 1, i_\tau + 2, \dots, 3\}$ and all $j'_\tau \in \{j_\tau + 1, j_\tau + 2, \dots, 3\}$ with $\tau \in \mathcal{M}$, hold. Let $\tau \in \mathcal{M}$. Then the entry “-” at the position of i_τ (or j_τ) just expresses that the relation $X \leq_\tau^i Y$

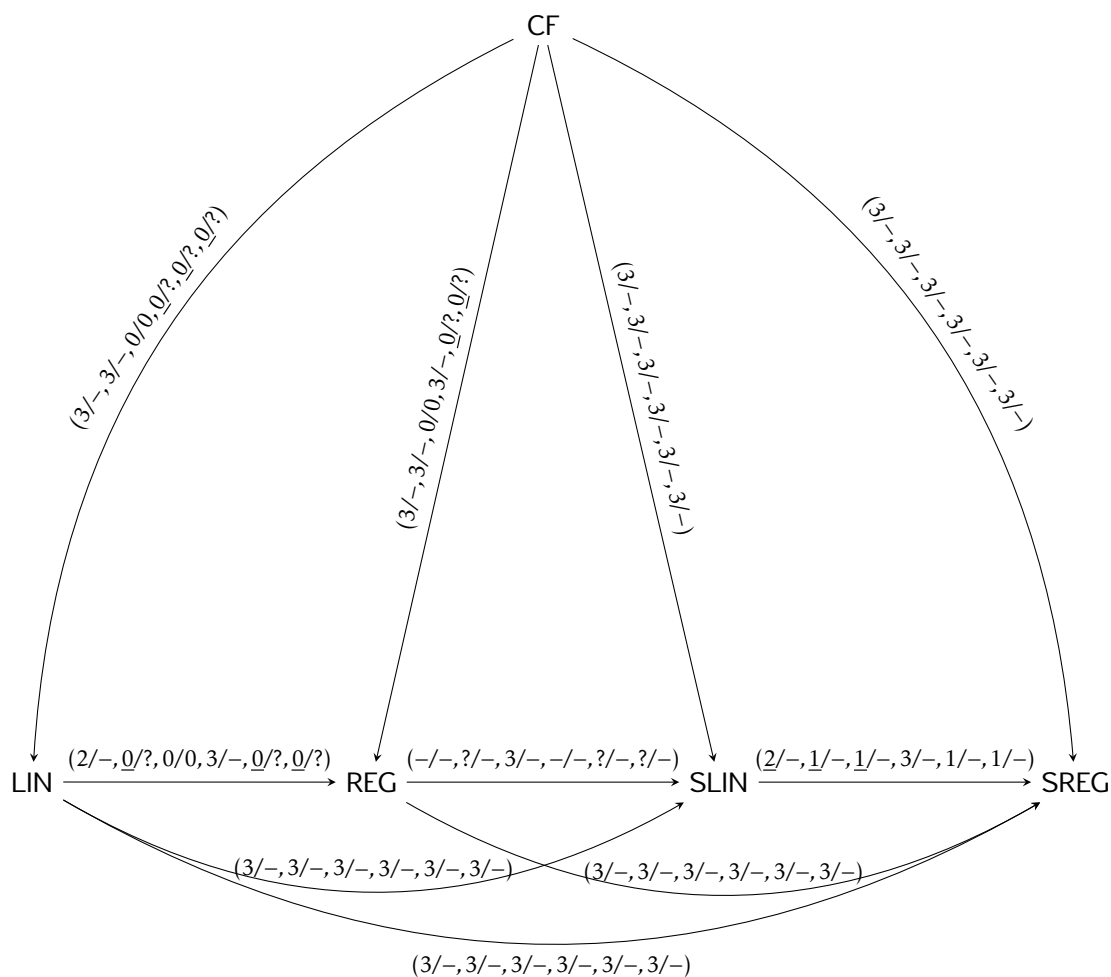


Figure 53: Relations between the grammar types w.r.t. a fixed measure type.

(or $Y \leq_{\tau}^i X$, respectively) does not hold for any $i \in \{0, 1, 2, 3\}$, i.e., $X \not\leq_{\tau}^i Y$ (or $Y \not\leq_{\tau}^i X$, respectively), for all $i \in \{0, 1, 2, 3\}$. Moreover, the entry “?” expresses that we do not know whether or not any relation between the involved grammar types holds. If any number $i_{\tau} \in \{0, 1, 2, 3\}$ (or $j_{\tau} \in \{0, 1, 2, 3\}$), for $\tau \in \mathcal{M}$ is underlined, then this means that the relation $X \leq_{\tau}^{i_{\tau}} Y$ (or $Y \leq_{\tau}^{j_{\tau}} X$, respectively), holds, but we do not know whether or not any of the stronger relations $X \leq_{\tau}^{i'_{\tau}} Y$ (or $Y \leq_{\tau}^{j'_{\tau}} X$, respectively), for $i'_{\tau} \in \{i_{\tau} + 1, i_{\tau} + 2, \dots, 3\}$ and $j'_{\tau} \in \{j_{\tau} + 1, j_{\tau} + 2, \dots, 3\}$ holds. For instance,

$$\text{LIN} \xrightarrow{(2/-,\underline{0}/?,0/0,3/-,\underline{0}/?,\underline{0}/?) } \text{REG}$$

expresses that we have that $\text{LIN} \leq_{\mathcal{C}}^i \text{REG}$ and $\text{LIN} \not\leq_{\mathcal{C}}^3 \text{REG}$ as well as $\text{LIN} \leq_{\mathcal{C}_{\infty}}^j \text{REG}$, for all $i \in \{0, 1, 2\}$ and all $j \in \{0, 1, 2, 3\}$. Moreover, it also expresses that we have $\text{LIN} \leq_{\{\text{CC}, \text{SC}, \text{CC}_{\infty}, \text{SC}_{\infty}\}}^0 \text{REG}$, but $\text{LIN} \not\leq_{\text{SC}}^i \text{REG}$, for all $i \in \{1, 2, 3\}$. It also expresses that we do not know whether or not $\text{LIN} \leq_{\sigma}^i \text{REG}$, for $\sigma \in \{\text{CC}, \text{CC}_{\infty}, \text{SC}_{\infty}\}$ and $i \in \{1, 2, 3\}$ holds. In the other direction, it expresses that it holds that $\text{REG} \not\leq_{\{\mathcal{C}, \mathcal{C}_{\infty}\}}^i \text{LIN}$, for all $i \in \{0, 1, 2, 3\}$, as well as $\text{REG} \leq_{\text{SC}}^0 \text{LIN}$ and $\text{REG} \not\leq_{\text{SC}}^j \text{LIN}$, for all $j \in \{1, 2, 3\}$. Furthermore, it expresses that we do not know whether or not $\text{REG} \leq_{\tau}^i \text{LIN}$ holds for $\tau \in \{\text{CC}, \text{CC}_{\infty}, \text{SC}_{\infty}\}$ and $i \in \{0, 1, 2, 3\}$.

Now, we are going to relate the different grammar types in Δ w.r.t. the finite complexity measure types under investigation. As already mentioned earlier, the relations

$$\text{CF} \leq_{\mathcal{M}} \text{LIN} \leq_{\mathcal{M}} \text{REG} \leq_{\mathcal{M}} \text{SREG} \quad \text{and} \quad \text{CF} \leq_{\mathcal{M}} \text{LIN} \leq_{\mathcal{M}} \text{SLIN} \leq_{\mathcal{M}} \text{SREG} \quad (5.5)$$

hold by definition.

The following result was already shown in [BMCIW81]:

Theorem 5.2.3 ([BMCIW81, Theorem 1]). *It holds that*

$$\text{CF} \leq_{\mathcal{C}}^2 \text{LIN} \leq_{\mathcal{C}}^2 \text{REG} \leq_{\mathcal{C}}^2 \text{SREG} \quad \text{and} \quad \text{CF} \leq_{\mathcal{C}}^2 \text{LIN} \leq_{\mathcal{C}}^2 \text{SLIN}.$$

At this point, one may wonder whether it also holds that $\text{SLIN} \leq_{\mathcal{C}}^2 \text{SREG}$? With the help of the language

$$P_n = \{w\$w^R \mid w \in \{a, b\}^{\leq n}\},$$

the answer to this question is revealed in the subsequent lemma.

Theorem 5.2.4. *It holds that $\text{SLIN} \leq_{\mathcal{C}}^2 \text{SREG}$.*

PROOF. It is obvious that $\text{SLINC}(L) \leq \text{SREGC}(L)$, for all finite languages L , since every strict regular grammar is also strict linear. Now, consider the sequence $(P_n)_{n \geq 0}$. Then, by Corollary 4.2.10 and Lemma 4.2.11, we have

$$\text{SREGC}(P_n) \geq 2^n \quad \text{and} \quad \text{SLINC}(P_n) \leq 3n + 1,$$

respectively. As a consequence,

$$\lim_{n \rightarrow \infty} \frac{\text{SLINC}(P_n)}{\text{SREGC}(P_n)} = 0.$$

□

Next, we show whether or not, and if so, how the grammar types in Δ relate to each other w.r.t. to the finite exact and cover complexity. The attentive reader may have noticed that Equation 5.5 does not contain any information about the relation between strict linear and regular grammars. In the following theorem, we investigate this relation and, moreover, prove all results of Figure 53 pertaining to the finite exact and cover complexity.

Theorem 5.2.5. *For each $i \in \{0, 1, 2, 3\}$ and each $j \in \{0, 1\}$, it holds that*

1. $\{\text{SREG}, \text{REG}\} \not\leq_{\text{c}}^i \{\text{SLIN}, \text{LIN}\},$
2. $\text{CF} \leq_{\{\text{c}, \text{cc}\}}^i \Delta \setminus \{\text{CF}\},$ but $\Delta \setminus \{\text{CF}\} \not\leq_{\{\text{c}, \text{cc}\}}^i \text{CF},$
3. $\text{LIN} \not\leq_{\{\text{c}, \text{cc}\}}^3 \text{REG}$ and $\text{REG} \not\leq_{\{\text{c}, \text{cc}\}}^3 \text{LIN},$
4. $\text{SLIN} \leq_{\text{cc}}^j \text{SREG},$ but $\text{SREG} \not\leq_{\text{cc}}^i \text{SLIN},$
5. $\Gamma_s \not\leq_{\{\text{c}, \text{cc}\}}^i \Gamma,$
6. $\text{LIN} \leq_{\{\text{c}, \text{cc}\}}^i \Gamma_s$ and $\text{REG} \leq_{\{\text{c}, \text{cc}\}}^i \text{SREG}.$

PROOF. For showing claim 1., let $X \in \{\text{SREG}, \text{REG}\}$ and $n \geq 5$ an integer. Then, from Corollary 4.2.10 and Lemma 4.2.11, we know that

$$Xc(P_n) \geq 2^n, \quad \text{LINc}(P_n) \leq 3n + 1, \quad \text{and} \quad \text{SLINc}(P_n) \leq 3n + 1,$$

respectively. Thus, since, for $n \geq 5$,

$$Xc(P_n) - \text{LINc}(P_n) \geq 2^n - 3n - 1 \geq n \quad \text{and} \quad Xc(P_n) - \text{SLINc}(P_n) \geq 2^n - 3n - 1 \geq n,$$

we have that $\{\text{SREG}, \text{REG}\} \not\leq_{\text{c}}^i \{\text{SLIN}, \text{LIN}\}$ and, moreover, that the first conditions of the relations $\{\text{SREG}, \text{REG}\} \leq_{\text{c}}^i \{\text{SLIN}, \text{LIN}\}$ are not satisfied for any $i \in \{1, 2, 3\}$, i.e., $\{\text{SREG}, \text{REG}\} \not\leq_{\text{c}}^i \{\text{SLIN}, \text{LIN}\}$, for all $i \in \{1, 2, 3\}$. Therefore, it holds that

$$\{\text{SREG}, \text{REG}\} \not\leq_{\text{c}}^i \{\text{SLIN}, \text{LIN}\},$$

for each $i \in \{0, 1, 2, 3\}$.

In order to prove claim 2., consider, for each integer $n \geq 4$, the language

$$L_n = \{a, b\}^{\leq n}.$$

From Lemmas 4.1.5 and 5.2.2, it follows that

$$\text{CFc}(L_n) \leq 4 \quad \text{and} \quad \text{LINc}(L_n) \geq \log(2^{n+1} - 1) + 1 \geq n + 1.$$

Moreover, Lemma 5.2.2 also implies that

$$\text{LINcc}(L_n) \geq n + 1.$$

Furthermore, it holds that

$$\text{CF}_{\text{cc}}(L_n) \leq \text{CF}_c(L_n) \leq 4,$$

since every grammar generating L_n is also a grammar covering L_n . Note that, by Equation 5.4, we have that

$$\text{CF} \leq_{\{c, \text{cc}\}} \text{LIN} \leq_{\{c, \text{cc}\}} \text{REG} \leq_{\{c, \text{cc}\}} \text{SREG} \quad \text{and} \quad \text{CF} \leq_{\{c, \text{cc}\}} \text{LIN} \leq_{\{c, \text{cc}\}} \text{SLIN} \leq_{\{c, \text{cc}\}} \text{SREG}.$$

Therefore, for each $\tau \in \{c, \text{cc}\}$, we have that

$$\text{SREG}\tau(L_n) \geq \text{REG}\tau(L_n) \geq \text{LIN}\tau(L_n) \geq n+1 \quad \text{and} \quad \text{SLIN}\tau(L_n) \geq \text{LIN}\tau(L_n) \geq n+1.$$

Thus, we immediately get

$$\text{CF} \leq_{\{c, \text{cc}\}}^i \Delta \setminus \{\text{CF}\},$$

for each $i \in \{0, 1, 2, 3\}$. Furthermore, from the fact that

$$X\tau(L_n) - \text{CF}\tau(L_n) \geq n-3,$$

for $X \in \Delta \setminus \{\text{CF}\}$, it follows that $\Delta \setminus \{\text{CF}\} \not\leq_{\{c, \text{cc}\}} \text{CF}$ and, moreover, that the first condition of the relation $\Delta \setminus \{\text{CF}\} \leq_{\{c, \text{cc}\}}^i \text{CF}$ is not satisfied for any $i \in \{1, 2, 3\}$, i.e., $\Delta \setminus \{\text{CF}\} \not\leq_{\{c, \text{cc}\}}^i \text{CF}$, for all $i \in \{1, 2, 3\}$. Therefore, it holds that

$$\Delta \setminus \{\text{CF}\} \not\leq_{\{c, \text{cc}\}}^i \text{CF},$$

for each $i \in \{0, 1, 2, 3\}$.

Next, we show claim 3. To this end, let L be an arbitrary finite language and G be a minimal linear grammar generating L , i.e., $L(G) = L$ and $|G| = \text{LINc}(L)$. Clearly,

$$\text{REGc}(L) \leq |L| = |L(G)|.$$

From Lemma 3.3.1, it follows that $|L(G)| \leq 2^{\text{LINc}(L)-1}$. Thus,

$$\text{REGc}(L) \leq |L(G)| \leq 2^{\text{LINc}(L)-1}.$$

The function $f: \mathbb{N} \rightarrow \mathbb{N}$ with $x \mapsto 2^{x-1}$ is a witness for

$$\text{REGc}(L') \leq f(\text{LINc}(L')),$$

for all finite languages L' , i.e., $\text{LIN} \not\leq_c^3 \text{REG}$. Since we have $\text{LINc}(L') \leq \text{REGc}(L')$, setting $f = \text{id}_{\mathbb{N}}$ yields $\text{REG} \not\leq_c^3 \text{LIN}$. The result for the cover complexity can be shown similarly by observing that any minimal linear grammar G' with $L(G') \supseteq L$ and $|G'| = \text{LINcc}(L)$ is a linear grammar generating the finite language $L(G')$. Thus,

$$\text{REGcc}(L) \leq |L| \leq |L(G')|.$$

Again, by Lemma 3.3.1, we have $|L(G')| \leq 2^{\text{LINcc}(L)-1}$, and therefore,

$$\text{REGcc}(L) \leq |L(G')| \leq 2^{\text{LINcc}(L)-1}.$$

The claim then follows by proceeding as for the exact complexity.

We prove claims 4.–6. simultaneously. Let $\tau \in \{c, cc\}$ and, for each integer $n \geq 1$, consider the language

$$L_n = \{a^{2n+1}\}.$$

On the one hand, for $X \in \Gamma$, we clearly have that

$$X\tau(L_n) = 1, \quad (5.6)$$

but, on the other hand, by Lemma 4.1.12 and Corollary 4.1.13, we have that

$$\text{SREG}\tau(L_n) \geq 2n + 1 \quad \text{and} \quad \text{SLIN}\tau(L_n) \geq n + 1. \quad (5.7)$$

Moreover,

$$\text{SLIN}_{cc}(L_n) \leq n + 1,$$

as shown by the strict linear grammar G with the following set of productions

$$\begin{aligned} S &\rightarrow aA_2a \\ A_i &\rightarrow aA_{i+1}a \quad \text{for } 2 \leq i \leq n \\ A_{n+1} &\rightarrow a. \end{aligned}$$

Therefore, since

$$\text{SREG}\tau(L_n) - \text{SLIN}\tau(L_n) \geq 2n + 1 - n - 1 = n, \quad (5.8)$$

we obtain that $\text{SREG} \not\leq_{cc}^i \text{SLIN}$ and, moreover, that the first condition of the relation $\text{SREG} \leq_{cc}^i \text{SLIN}$ is not satisfied, for any $i \in \{1, 2, 3\}$, i.e., $\text{SREG} \not\leq_{cc}^i \text{SLIN}$, for all $i \in \{1, 2, 3\}$. As a consequence, it holds that

$$\text{SREG} \not\leq_{cc}^i \text{SLIN},$$

for each $i \in \{0, 1, 2, 3\}$. From Equations 5.4 and 5.8, it also follows that

$$\text{SLIN} \leq_{cc}^j \text{SREG},$$

for each $j \in \{0, 1\}$.

Furthermore, since, for $X \in \Gamma$, we have that

$$\text{SREG}\tau(L_n) - X\tau(L_n) \geq 2n + 1 - 1 = 2n \quad \text{and} \quad \text{SLIN}\tau(L_n) - X\tau(L_n) \geq n + 1 - 1 = n,$$

we obtain that $\Gamma_s \not\leq_{\{c, cc\}} \Gamma$ and, moreover, that the first condition of the relation $\Gamma_s \leq_{\{c, cc\}}^i \Gamma$ is not satisfied for any $i \in \{1, 2, 3\}$, i.e., $\Gamma_s \not\leq_{\{c, cc\}}^i \Gamma$, for all $i \in \{1, 2, 3\}$. As a consequence, it holds that

$$\Gamma_s \not\leq_{\{c, cc\}}^i \Gamma,$$

for each $i \in \{0, 1, 2, 3\}$.

From Equations 5.4, 5.6, and 5.7, it follows that

$$\text{REG} \leq_{\{c,cc\}} \text{SREG} \quad \text{and} \quad \text{LIN} \leq_{\{c,cc\}} \Gamma_s$$

as well as that the first conditions of $\text{REG} \leq_{\{c,cc\}}^i \text{SREG}$ and those of $\text{LIN} \leq_{\{c,cc\}}^i \Gamma_s$ are satisfied for each $i \in \{1, 2, 3\}$. Thus, we get that

$$\text{REG} \leq_{\{c,cc\}}^i \text{SREG} \quad \text{and} \quad \text{LIN} \leq_{\{c,cc\}}^i \Gamma_s,$$

for each $i \in \{0, 1, 2, 3\}$.

This finishes the proof of the stated claims. \square

For the scattered complexity, we have the following situation:

Theorem 5.2.6. *Let $X, Y \in \Gamma$. Then*

1. $\text{REG} =_{\text{sc}} \text{LIN} =_{\text{sc}} \text{CF}$, but $X \not\leq_{\text{sc}}^i Y$, for each $i \in \{1, 2, 3\}$ and $X \neq Y$,
2. $\Gamma \leq_{\text{sc}}^i \Gamma_s$, but $\Gamma_s \not\leq_{\text{sc}}^i \Gamma$, for each $i \in \{0, 1, 2, 3\}$, and
3. $\text{SLIN} \leq_{\text{sc}}^i \text{SREG}$, for each $i \in \{0, 1\}$, but $\text{SREG} \not\leq_{\text{sc}}^j \text{SLIN}$, for each $j \in \{0, 1, 2, 3\}$.

PROOF. In order to prove claim 1., observe that by Theorem 4.1.6, we have that

$$\text{REG}_{\text{sc}}(L) = \text{LIN}_{\text{sc}}(L) = \text{CF}_{\text{sc}}(L) = \begin{cases} 1 & \text{if } L \neq \emptyset, \\ 0 & \text{if } L = \emptyset, \end{cases}$$

i.e., $X =_{\text{sc}} Y$, for all $X, Y \in \Gamma$ with $X \neq Y$. An immediate consequence of this is that none of the stronger relations $X \leq_{\text{sc}}^i Y$, for $i \in \{1, 2, 3\}$, holds for any two distinct grammar types $X, Y \in \Gamma$.

We prove claims 2. and 3. simultaneously. For any integer $n \geq 0$, let

$$L_n = \{a^{2^{n+1}}\}.$$

Then, by Theorem 4.1.6, we clearly have that

$$\text{REG}_{\text{sc}}(L_n) = \text{LIN}_{\text{sc}}(L_n) = \text{CF}_{\text{sc}}(L_n) = 1. \quad (5.9)$$

On the other hand, by Corollary 4.1.14, we have that

$$\text{SREG}_{\text{sc}}(L_n) \geq 2n + 1 \quad \text{and} \quad \text{SLIN}_{\text{sc}}(L_n) \geq n + 1. \quad (5.10)$$

From Equations 5.9 and 5.10, we clearly get that $\Gamma \leq_{\text{sc}} \Gamma_s$ as well as that the first condition of $\Gamma \leq_{\text{sc}}^i \Gamma_s$ is satisfied for each $i \in \{1, 2, 3\}$. Thus, it follows that

$$\Gamma \leq_{\text{sc}}^i \Gamma_s,$$

holds for all $i \in \{0, 1, 2, 3\}$.

The strict linear grammar defined in the proof of Theorem 5.2.5 (4) shows that $\text{SLINsc}(L_n) \leq n + 1$. This implies that

$$\text{SREGsc}(L_n) - \text{SLINsc}(L_n) \geq 2n + 1 - n - 1 = n.$$

Therefore, it follows that $\text{SREG} \not\leq_{\text{sc}} \text{SLIN}$ and, moreover, that the first condition of $\text{SREG} \leq_{\text{sc}}^i \text{SLIN}$, is not satisfied for any $i \in \{1, 2, 3\}$, i.e., $\text{SREG} \not\leq_{\text{sc}}^i \text{SLIN}$, for each $i \in \{1, 2, 3\}$. As a consequence, it holds that

$$\text{SREG} \not\leq_{\text{sc}}^i \text{SLIN},$$

for all $i \in \{0, 1, 2, 3\}$. Furthermore, since every strict regular grammar is also strict linear, we have that $\text{SLINsc}(L) \leq \text{SREGsc}(L)$, for every finite language L , i.e., $\text{SLIN} \leq_{\text{sc}} \text{SREG}$ holds, and, moreover, that the first condition of $\text{SLIN} \leq_{\text{sc}}^1 \text{SREG}$ is also satisfied. Putting things together, we obtain that

$$\text{SLIN} \leq_{\text{sc}}^i \text{SREG},$$

for each $i \in \{0, 1\}$.

From Equations 5.9 and 5.10, for $X \in \Gamma$, we get that

$$\text{SREGsc}(L_n) - X_{\text{sc}}(L_n) \geq 2n + 1 - 1 = 2n \quad \text{and} \quad \text{SLINsc}(L_n) - X_{\text{sc}}(L_n) \geq n + 1 - 1 = n.$$

As a consequence, we have that $\Gamma_s \not\leq_{\text{sc}} \Gamma$, and, moreover, that the first condition of $\Gamma_s \leq_{\text{sc}}^i \Gamma$ is not satisfied for any $i \in \{1, 2, 3\}$, i.e., $\Gamma_s \not\leq_{\text{sc}}^i \Gamma$, for each $i \in \{1, 2, 3\}$. Therefore,

$$\Gamma_s \not\leq_{\text{sc}}^i \Gamma,$$

for all $i \in \{0, 1, 2, 3\}$.

This finishes the proof of the claims. \square

It remains to relate the different grammar types in Δ w.r.t. the infinite complexity measure types c_∞ , cc_∞ , and sc_∞ . As a prerequisite, we will show that every regular grammar that infinitely generates the language

$$P_n = \{w\$w^R \mid w \in \{a, b\}^{\leq n}\}$$

needs at least a number of productions that is exponential in n . A first step towards this goal is to prove that any regular grammar G with $\varepsilon \notin L(G)$ can be transformed into an equivalent ε -free strict regular grammar whose number of productions is at most the number of productions of the given grammar times a factor of $2 \cdot \ell$, where ℓ is the length of a longest production right-hand side in the given grammar. This is achieved by combining the following two Lemmas 5.2.7 and 5.2.8.

Lemma 5.2.7. *Let $G = (N, \Sigma, P, S)$ be a regular grammar generating a language $L \subseteq \Sigma^*$ with $\ell = \max\{|\alpha| \mid A \rightarrow \alpha \in P\}$. Then there is a strict regular grammar G' such that $L(G') = L(G)$ and $|G'| \leq |G| \cdot \ell$.*

PROOF. In order to construct the strict regular grammar G' from G , we replace each G -production of the form

$$p_i : A \rightarrow a_1 a_2 \dots a_{m_i} \alpha$$

with $i \in \{1, 2, \dots, |G|\}$, $m_i \leq \ell - 1$, $A \in N$, $a_1, a_2, \dots, a_{m_i} \in \Sigma$, and $\alpha \in \Sigma \cup N$ by the productions

$$\begin{aligned} A &\rightarrow a_1 A_1 \\ A_1 &\rightarrow a_2 A_2 \\ &\vdots \\ A_{m_i-1} &\rightarrow a_{m_i} \alpha \end{aligned} \quad \text{if } \alpha \in N$$

or

$$\begin{aligned} A &\rightarrow a_1 A_1 \\ A_1 &\rightarrow a_2 A_2 \\ &\vdots \\ A_{m_i-1} &\rightarrow a_{m_i} A_{m_i} \\ A_{m_i} &\rightarrow \alpha, \end{aligned} \quad \text{if } \alpha \in \Sigma$$

where the A_j , for $j \in \{1, 2, \dots, m_i\}$, are fresh nonterminals not occurring in N . Since $m_i \leq \ell - 1$, for all $i \in \{1, 2, \dots, |G|\}$, we clearly have that $|G'| \leq |G| \cdot \ell$. Moreover, it is easy to see that $L(G') = L(G)$. \square

Lemma 5.2.8. *Let $G = (N, \Sigma, P, S)$ be a regular grammar generating a language $L \subseteq \Sigma^*$ with $\varepsilon \notin L$. Then there is an equivalent ε -free regular grammar $G' = (N', \Sigma, P', S)$ satisfying $|G'| \leq 2 \cdot |G|$.*

PROOF. If G does not contain ε -productions, then set $G' = G$ and we are done. So, assume that there is some production $A \rightarrow \varepsilon \in P$. For each production of the form

$$B \rightarrow wA,$$

that is, in which A occurs on the right-hand side, we add the production

$$B \rightarrow w$$

to P . Next, we omit the production $A \rightarrow \varepsilon$ from G . This procedure is repeated until G contains no ε -productions anymore. We clearly end up with a regular grammar G' without ε -productions that satisfies both $L(G') = L(G)$ and $|G'| \leq 2 \cdot |G|$. \square

Another prerequisite for the lower bound result on the infinite regular complexity of the language P_n is a technique which we call the *triple construction (for regular grammars)*. The subsequent proof is a modified version of the one of [Woo87, Theorem 9.1.3] for arbitrary context-free grammars. Given a regular grammar that generates a language

not containing the empty word, this version of the triple construction produces the intersection of the language generated by the given grammar and the set of all words of length at most n over the alphabet of the language generated by the given grammar. To this end, we are going to construct a regular grammar that generates the above mentioned intersection from the given regular grammar and an NFA that accepts the set of all words of length at most n over the alphabet of the language generated by the given grammar. The name triple construction stems from the fact that the nonterminals in the newly constructed grammar are of the form $[i, A, j]$, where A is a nonterminal occurring in the given grammar and both i and j are states occurring in the NFA.

Theorem 5.2.9. *Let $X \in \{\text{SREG}, \text{REG}\}$, $n \geq 1$ be an integer, $G = (N, \Sigma, P, S)$ be an X -grammar generating a language $L \subseteq \Sigma^*$ with $\varepsilon \notin L$, and $\ell = \max\{|\alpha| \mid A \rightarrow \alpha \in P\}$. Then there is a strict regular grammar G' such that $L(G') = L \cap \Sigma^{\leq n}$ and $|G'| \leq |G| \cdot 2 \cdot \ell \cdot (n+1)^3$.*

PROOF. First, since $\varepsilon \notin L$, we transform G into an equivalent ε -free strict regular grammar $G' = (N', \Sigma, P', S')$. By Lemmas 5.2.7 and 5.2.8, this transformation might increase the number of productions in G by at most a factor of $2 \cdot \ell$, i.e.,

$$|G'| \leq |G| \cdot 2 \cdot \ell.$$

Since G' is strict regular and ε -free, all productions in P' are of the form

$$\begin{aligned} A &\rightarrow a, \\ A &\rightarrow aB, \text{ or} \\ A &\rightarrow B, \end{aligned}$$

for $a \in \Sigma$ and $A, B \in N'$.

Now, assume, without loss of generality, that $\Sigma = \{a_1, a_2, \dots, a_n\}$ and consider the finite language $\Sigma^{\leq n}$. Clearly, the NFA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ depicted in Figure 54 with $Q = \{0, 1, \dots, n\}$, $q_0 = 0$, $F = \{0, n\}$, $\delta(i, \varepsilon) = \{i\}$, and $\delta(i, a) = \{i+1, n\}$, for $0 \leq i < n$ and $a \in \Sigma$, accepts the language $\Sigma^{\leq n}$, i.e.,

$$L(\mathcal{A}) = \Sigma^{\leq n}.$$

We are now going to construct a strict regular grammar G'' from G' and \mathcal{A} such that

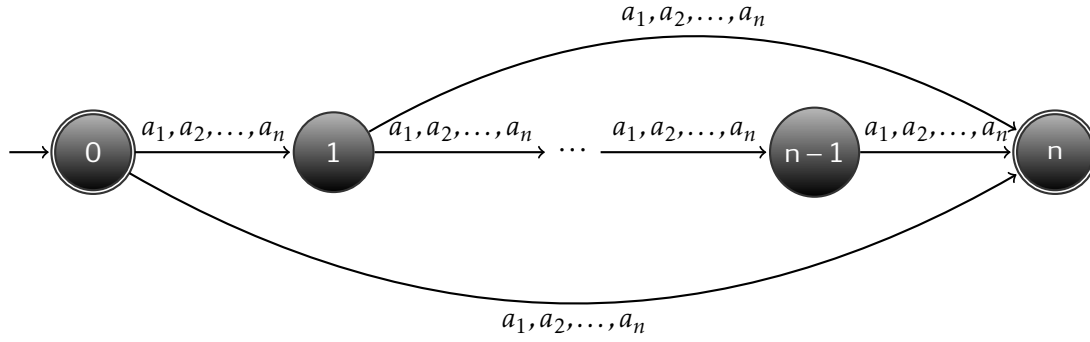
$$L(G'') = L(G') \cap L(\mathcal{A}) = L \cap \Sigma^{\leq n}.$$

To this end, let $G'' = (N'', \Sigma, P'', S'')$, where $S'' \notin N'$ is a new nonterminal,

$$N'' = \{[p, A, q] \mid p, q \in Q \text{ and } A \in N'\},$$

and

$$\begin{aligned} P'' = & \{[p, A, q] \rightarrow a \mid q \in \delta(p, a), A \rightarrow a \in P', \text{ and } a \in \Sigma\} \\ & \cup \{[p, A, q] \rightarrow a[r, B, q] \mid r \in \delta(p, a), q \in Q, A \rightarrow aB \in P', \text{ and } a \in \Sigma\} \\ & \cup \{[p, A, q] \rightarrow [p, B, q] \mid p, q \in Q \text{ and } A \rightarrow B \in P'\} \\ & \cup \{S'' \rightarrow [0, S', 0], S'' \rightarrow [0, S', n]\}. \end{aligned}$$

Figure 54: NFA \mathcal{A} for the finite language $\Sigma^{\leq n}$.

Since \mathcal{A} consists of $n + 1$ states, i.e., $|Q| = n + 1$, and the productions of the form

$$[p, A, q] \rightarrow a[r, B, q]$$

make up the greatest portion of P'' , it follows that

$$|G''| \leq |G'| \cdot (n + 1)^3 \leq |G| \cdot 2 \cdot \ell \cdot (n + 1)^3.$$

The intention behind this so-called *triple construction* is that a nonterminal $[p, A, q]$ generates the words that are derivable from A in G' which also cause the automaton \mathcal{A} to move from state p to state q . Thus, $[0, S', f]$, for $f \in \{0, n\}$, generates words in $L(G') \cap L(\mathcal{A})$. This can be proved formally by means of the following claim.

Claim. Let $w \in \Sigma^*$. Then, for all $A \in N'$ and all $p, q \in Q$, we have that

$$[p, A, q] \Rightarrow_{G''}^* w \quad \text{if and only if} \quad A \Rightarrow_{G'}^* w \text{ and } pw \Big|_{\mathcal{A}}^* q.$$

Note that it is straightforward to prove, for all $A, B \in N'$ and all $p, q \in Q$, that

$$[p, A, q] \Rightarrow_{G''}^* [p, B, q] \quad \text{iff} \quad A \Rightarrow_{G'}^* B \text{ and } p \in \delta(p, \varepsilon). \quad (5.11)$$

by induction on the length of the derivation.

In order to prove the claim, we proceed by induction on $|w|$, i.e., the length of w . Since $\varepsilon \notin L(G')$ and both G' and G'' are ε -free, we can assume that $|w| \geq 1$.

- **Base case:** Assume $|w| = 1$. By definition and since G'' is ε -free, we have that

$$[p, A, q] \Rightarrow_{G''}^* w \quad \text{iff} \quad [p, A, q] \Rightarrow_{G''} w \text{ or } [p, A, q] \Rightarrow_{G''}^* [p, B, q] \Rightarrow_{G''} w.$$

Thus, we distinguish the following two cases:

In the first case, we have that

$$\begin{aligned}
 [p, A, q] \Rightarrow_{G''}^* w & \text{ iff } [p, A, q] \Rightarrow_{G''} w \\
 & \text{ iff } [p, A, q] \rightarrow w \in P'' \\
 & \text{ iff } A \rightarrow w \in P' \text{ and } q \in \delta(p, w) \\
 & \text{ iff } A \Rightarrow_{G'} w \text{ and } pw \Big|_{\mathcal{A}}^* q.
 \end{aligned}$$

This concludes the first case.

In the second case, we have that

$$\begin{aligned}
 [p, A, q] \Rightarrow_{G''}^* w & \text{ iff } [p, A, q] \Rightarrow_{G''}^* [p, B, q] \Rightarrow_{G''} w \\
 & \text{ iff } A \Rightarrow_{G'}^* B, p \in \delta(p, \varepsilon), \text{ and } [p, B, q] \rightarrow w \in P'' \\
 & \text{ iff } A \Rightarrow_{G'}^* B, p \in \delta(p, \varepsilon), B \rightarrow w \in P', \text{ and } q \in \delta(p, w) \\
 & \text{ iff } A \Rightarrow_{G'}^* w \text{ and } pw \Big|_{\mathcal{A}}^* q.
 \end{aligned}$$

This finishes the proof of the base case.

- **Induction step:** We prove both directions separately.

For the left-to-right direction, let $w \in \Sigma^*$ be a word with $|w| \geq 2$, i.e.,

$$w = aw',$$

for some $a \in \Sigma$ and $w' \in \Sigma^+$. Assume that

$$[p, A, q] \Rightarrow_{G''}^* w.$$

Consequently, we have to distinguish two cases:

In the first case, we have the derivation

$$[p, A, q] \Rightarrow_{G''} a[r, B, q] \Rightarrow_{G''}^* aw',$$

where $r \in \delta(p, a)$ and $A \rightarrow aB \in P'$. Therefore, we have that

$$[r, B, q] \Rightarrow_{G''}^* w'.$$

Thus, by induction hypothesis, we have that

$$B \Rightarrow_{G'}^* w' \text{ and } rw' \Big|_{\mathcal{A}}^* q.$$

Putting things together, we obtain that

$$A \Rightarrow_{G'}^* w \text{ and } pw \Big|_{\mathcal{A}}^* q.$$

This concludes the first case.

In the second case, we have the derivation

$$[p, A, q] \Rightarrow_{G''} [p, B, q] \Rightarrow_{G''}^* aw',$$

where $A \rightarrow B \in P'$. In particular, we must have that

$$[p, A, q] \Rightarrow_{G''} [p, B, q] \Rightarrow_{G''}^* [p, C, q] \Rightarrow_{G''} a[r, D, q] \Rightarrow_{G''}^* aw',$$

where $r \in \delta(p, a)$ and $C \rightarrow aD \in P'$. Therefore, by Equation 5.11, it follows that

$$A \Rightarrow_{G'} B \Rightarrow_{G'}^* C \quad \text{and} \quad p \in \delta(p, \varepsilon).$$

Furthermore, by induction hypothesis, we have that

$$D \Rightarrow_{G'}^* w' \quad \text{and} \quad rw' \mid_{\mathcal{A}}^* q.$$

Putting things together, we obtain that

$$A \Rightarrow_{G'}^* w \quad \text{and} \quad pw \mid_{\mathcal{A}}^* q.$$

This finishes the proof of the left-to-right direction.

For the right-to-left direction, let $w \in \Sigma^*$ be a word with $|w| \geq 2$, i.e.,

$$w = aw',$$

for some $a \in \Sigma$ and $w' \in \Sigma^*$. Assume that

$$A \Rightarrow_{G'}^* w \quad \text{and} \quad pw \mid_{\mathcal{A}}^* q.$$

Consequently, we have to distinguish two cases:

In the first case, we have that

$$A \Rightarrow_{G'} aB \Rightarrow_{G'}^* aw', \quad pa \mid_{\mathcal{A}} r, \quad \text{and} \quad rw' \mid_{\mathcal{A}}^* q.$$

By induction hypothesis and definition of G'' , we thus have that

$$[p, A, q] \rightarrow a[r, B, q] \in P'' \quad \text{and} \quad [r, B, q] \Rightarrow_{G''}^* w',$$

since $A \rightarrow aB \in P'$, $p, q, r \in Q$, $a \in \Sigma$, and $r \in \delta(p, a)$. Putting things together, we obtain that

$$[p, A, q] \Rightarrow_{G''} a[r, B, q] \Rightarrow_{G''}^* aw',$$

i.e.,

$$[p, A, q] \Rightarrow_{G''}^* w.$$

This concludes the first case.

In the second case, we have that

$$A \Rightarrow_{G'} B \Rightarrow_{G'}^* C \Rightarrow_{G'} aD \Rightarrow_{G'}^* aw', \quad p \in \delta(p, \varepsilon), \quad pa \mid_{\mathcal{A}} r, \quad \text{and} \quad rw' \mid_{\mathcal{A}}^* q.$$

By induction hypothesis, definition of G'' , and Equation 5.11, we thus have that

$$[p, A, q] \rightarrow [p, B, q] \in P'', \quad [p, B, q] \Rightarrow_{G''}^* [p, C, q], \quad [p, C, q] \rightarrow a[r, D, q] \in P''$$

as well as

$$[r, D, q] \Rightarrow_{G''}^* w'.$$

since $A \rightarrow B \in P'$, $B \Rightarrow_{G'}^* C$, $C \rightarrow aD \in P'$, $p, q, r \in Q$, $a \in \Sigma$, $p \in \delta(p, \varepsilon)$, $r \in \delta(p, a)$, and $rw' \vdash_A^* q$. Putting things together, we obtain that

$$[p, A, q] \Rightarrow_{G''} [p, B, q] \Rightarrow_{G''}^* [p, C, q] \Rightarrow_{G''} a[r, D, q] \Rightarrow_{G''}^* aw',$$

i.e.,

$$[p, A, q] \Rightarrow_{G''}^* w.$$

This finishes the proof of the right-to-left direction and in further consequence also the proof of the claim.

To complete the proof of the theorem, observe that by the above claim, for $w \in \Sigma^*$, it holds that

$$\begin{aligned} w \in L(G'') & \text{ iff } S'' \Rightarrow_{G''}^* w \\ & \text{ iff } [0, S', f] \Rightarrow_{G''}^* w, \text{ for some } f \in \{0, n\} \\ & \text{ iff } w \in L(G') \text{ and } w \in L(\mathcal{A}) \\ & \text{ iff } w \in L(G') \cap L(\mathcal{A}) = L \cap \Sigma^{\leq n}, \end{aligned}$$

that is,

$$L(G'') = L \cap \Sigma^{\leq n}.$$

This finishes the proof of the theorem. \square

With the help of the triple construction for regular grammars, we can now prove a lower bound on the infinite SREG- and REG-complexity of the language

$$P_n = \{w\$w^R \mid w \in \{a, b\}^{\leq n}\}.$$

The proof idea is as follows: let $X \in \{\text{SREG}, \text{REG}\}$ and $G = (N, \Sigma, P, S)$ be an X -grammar that is a witness for $X_{c_\infty}(P_n)$. Then we construct an X -grammar generating the finite set

$$L(G) \cap \Sigma^{\leq 2n+1}.$$

Since this language is equal to P_n , we can apply Corollary 4.2.10 in order to obtain a lower bound on $|G|$. By Theorem 5.2.9, there is a strict regular grammar $G' = (N', \Sigma, P', S')$ that generates the finite language $L(G) \cap \Sigma^{\leq 2n+1}$ and satisfies

$$|G'| \leq |G| \cdot 2 \cdot (2n+2) \cdot (2n+2)^3 = |G| \cdot 2 \cdot (2n+2)^4.$$

Theorem 5.2.10. *Let $X \in \{\text{SREG}, \text{REG}\}$ and $n \geq 1$ an integer. Then $X_{C_\infty}(P_n) = \Omega(2^n)$.*

PROOF. Assume to the contrary that there is an X -grammar G with

$$|G| = X_{C_\infty}(P_n) = o(2^n),$$

that is,

$$P_n = L(G) \cap \{a, b, \$\}^{\leq 2n+1}.$$

We can safely assume that the right-hand side of every production in G is of length at most $2n + 2$, because productions with a longer right-hand side can only generate words that are longer than $2n + 1$ and then G would not be minimal. Moreover, since $\varepsilon \notin P_n$, we can also, without loss of generality, assume that $\varepsilon \notin L(G)$, because by following the steps of the proof of Lemma 5.2.8, we can transform G into a regular grammar G^* with $L(G^*) = L(G) \setminus \{\varepsilon\}$ and $|G^*| \leq 2 \cdot |G| = 2 \cdot o(2^n) = o(2^n)$. Therefore, it also holds that $P_n = L(G^*) \cap \{a, b, \$\}^{\leq 2n+1}$. Next, we apply Theorem 5.2.9, i.e., the triple construction for regular grammars. Let G' be the result of the triple construction, i.e.,

$$L(G') = L(G) \cap \{a, b, \$\}^{\leq 2n+1} = P_n.$$

Then

$$|G'| = o(2^n),$$

because

$$o(2^n) \cdot (2 \cdot (2n + 2)^4) = o(2^n).$$

Since the strict regular grammar G' with $|G'| = o(2^n)$ generates P_n , we get a contradiction to the fact that

$$X_C(P_n) \geq 2^n$$

as shown in Corollary 4.2.10. Thus, we obtain

$$X_{C_\infty}(P_n) = \Omega(2^n).$$

This finishes the proof of the claim. \square

The situation changes drastically when we consider strict linear, linear, and context-free grammars. For these grammar types, we get an upper bound on the infinite complexity of the language P_n that is constant in n .

Lemma 5.2.11. *Let $X \in \{\text{SLIN}, \text{LIN}, \text{CF}\}$ and $n \geq 1$ an integer. Then $X_{C_\infty}(P_n) \leq 3$.*

PROOF. Let $X \in \{\text{SLIN}, \text{LIN}, \text{CF}\}$ and consider the strict linear grammar $G = (N, \{a, b, \$\}, P, S)$, where P contains the following productions:

$$S \rightarrow aSa \mid bSb \mid \$.$$

Clearly, G generates the language of all palindromes over the alphabet $\{a, b\}$ with middle marker $\$$, i.e.,

$$L(G) = \{w\$w^R \mid w \in \{a, b\}^*\}.$$

Since

$$P_n = L(G) \cap \{a, b, \$\}^{\leq 2n+1},$$

it follows that

$$\chi_{c_\infty}(P_n) \leq 3.$$

This finishes the proof of the claim. \square

The subsequent theorem states a taxonomy of the grammar types in Δ w.r.t. the infinite complexity measures.

Theorem 5.2.12. *For all $i \in \{0, 1, 2, 3\}$, all $j \in \{0, 1\}$, and all $k \in \{2, 3\}$, it holds that*

1. $CF \leq_{c_\infty}^i REG$, $LIN \leq_{c_\infty}^i REG$, and $SLIN \leq_{c_\infty}^i SREG$,
2. $\{SREG, REG\} \not\leq_{c_\infty}^i \{SLIN, LIN, CF\}$,
3. $CF \leq_{\mathcal{M}_\infty}^i \Gamma_s$, $LIN \leq_{\mathcal{M}_\infty}^i \Gamma_s$, $REG \leq_{\mathcal{M}_\infty}^i SREG$, and $SLIN \leq_{\{cc_\infty, sc_\infty\}}^j SREG$,
4. $SLIN \not\leq_{\mathcal{M}_\infty}^i \Gamma$ and $SREG \not\leq_{\mathcal{M}_\infty}^i \Delta \setminus \{SREG\}$, and
5. $SLIN \not\leq_{\{cc_\infty, sc_\infty\}}^k SREG$.

PROOF. We prove claims 1. and 2. simultaneously. By Theorem 5.2.10, it holds that

$$SREG_{c_\infty}(P_n) = \Omega(2^n) \quad \text{and} \quad REG_{c_\infty}(P_n) = \Omega(2^n).$$

In contrast, from Lemma 5.2.11, it follows that

$$CF_{c_\infty}(P_n) \leq 3, \quad LIN_{c_\infty}(P_n) \leq 3, \quad \text{and} \quad SLIN_{c_\infty}(P_n) \leq 3.$$

From Equation 5.4, we get that $CF \leq_{c_\infty} REG$, $LIN \leq_{c_\infty} REG$, and $SLIN \leq_{c_\infty} SREG$ as well as that the first conditions of the relations $CF \leq_{c_\infty}^i REG$, $LIN \leq_{c_\infty}^i REG$, and $SLIN \leq_{c_\infty}^i SREG$ are satisfied for any $i \in \{1, 2, 3\}$. Putting things together, we get that

$$CF \leq_{c_\infty}^i REG, \quad LIN \leq_{c_\infty}^i REG, \quad \text{and} \quad SLIN \leq_{c_\infty}^i SREG$$

hold for each $i \in \{0, 1, 2, 3\}$. Moreover, the above infinite complexity bounds on the language P_n also demonstrate that neither $REG \leq_{c_\infty} \{SLIN, LIN, CF\}$ and $SREG \leq_{c_\infty} \{SLIN, LIN, CF\}$ nor the first conditions of $REG \leq_{c_\infty}^i \{SLIN, LIN, CF\}$ and $SREG \leq_{c_\infty}^i \{SLIN, LIN, CF\}$ are satisfied for any $i \in \{1, 2, 3\}$. Therefore,

$$REG \not\leq_{c_\infty}^i \{SLIN, LIN, CF\} \quad \text{and} \quad SREG \not\leq_{c_\infty}^i \{SLIN, LIN, CF\},$$

for each $i \in \{0, 1, 2, 3\}$.

In order to prove claims 3. and 4., consider, for each integer $n \geq 2$, the language

$$L_n = \{a_1 a_2 \cdots a_{2n+1}\},$$

where the $a_i \in \Sigma$, for $1 \leq i \leq 2n+1$, are pairwise distinct. Moreover, let $X \in \Gamma$ and $\tau \in \mathcal{M}_\infty$. It is easy to see that

$$X\tau(L_n) = 1, \quad (5.12)$$

for each $X \in \Gamma$. However,

$$\text{SREG}\tau(L_n) \geq 2n+1 \quad \text{and} \quad \text{SLIN}\tau(L_n) \geq n+1, \quad (5.13)$$

because, by definition, any strict regular grammar G with

$$L(G) \cap \{a_1, a_2, \dots, a_{2n+1}\}^{\leq 2n+1} \geq L_n,$$

for $\geq \in \{=, \supseteq, \geq\}$, needs at least one production for each letter in the set $\{a_1, a_2, \dots, a_{2n+1}\}$. Similarly, any strict linear grammar G' with

$$L(G') \cap \{a_1, a_2, \dots, a_{2n+1}\}^{\leq 2n+1} \geq L_n,$$

for $\geq \in \{=, \supseteq, \geq\}$, needs at least $n+1$ distinct productions, since, by definition, each strict linear production can contain at most two letters from $\{a_1, a_2, \dots, a_{2n+1}\}$. Observe that the strict linear grammar consisting of the productions

$$\begin{aligned} S &\rightarrow a_1 A_2 a_{2n+1} \\ A_2 &\rightarrow a_2 A_3 a_{2n} \\ &\vdots \\ A_n &\rightarrow a_n A_{n+1} a_{n+2} \\ A_{n+1} &\rightarrow a_{n+1} \end{aligned}$$

shows that $\text{SLIN}\tau(L_n) \leq n+1$. Therefore, it follows that

$$\text{SREG}\tau(L_n) - \text{SLIN}\tau(L_n) \geq 2n+1 - n-1 = n. \quad (5.14)$$

From Equation 5.4, we get that the relations $\text{CF} \leq_{\mathcal{M}_\infty} \Gamma_s$, $\text{LIN} \leq_{\mathcal{M}_\infty} \Gamma_s$, $\text{REG} \leq_{\mathcal{M}_\infty} \text{SREG}$, and $\text{SLIN} \leq_{\{\text{cc}_\infty, \text{sc}_\infty\}} \text{SREG}$ as well as the first conditions of the relations $\text{CF} \leq_{\mathcal{M}_\infty}^i \Gamma_s$, $\text{LIN} \leq_{\mathcal{M}_\infty}^i \Gamma_s$, $\text{REG} \leq_{\mathcal{M}_\infty}^i \text{SREG}$, and $\text{SLIN} \leq_{\{\text{cc}_\infty, \text{sc}_\infty\}}^1 \text{SREG}$ are satisfied for any $i \in \{1, 2, 3\}$. Putting things together, we get that

$$\text{CF} \leq_{\mathcal{M}_\infty}^i \Gamma_s, \quad \text{LIN} \leq_{\mathcal{M}_\infty}^i \Gamma_s, \quad \text{REG} \leq_{\mathcal{M}_\infty}^i \text{SREG}, \quad \text{and} \quad \text{SLIN} \leq_{\{\text{cc}_\infty, \text{sc}_\infty\}}^j \text{SREG}$$

are true for all $i \in \{0, 1, 2, 3\}$ and $j \in \{0, 1\}$. Moreover, Equations 5.12 and 5.13 show that neither $\text{SREG} \leq_{\mathcal{M}_\infty} \Gamma$ and $\text{SLIN} \leq_{\mathcal{M}_\infty} \Gamma$ nor the first conditions of $\text{SREG} \leq_{\mathcal{M}_\infty}^i \Gamma$ and $\text{SLIN} \leq_{\mathcal{M}_\infty}^i \Gamma$ are satisfied for any $i \in \{1, 2, 3\}$. As a consequence,

$$\text{SREG} \not\leq_{\mathcal{M}_\infty}^i \Gamma \quad \text{and} \quad \text{SLIN} \not\leq_{\mathcal{M}_\infty}^i \Gamma,$$

for each $i \in \{0, 1, 2, 3\}$. Finally, by Equation 5.14, it follows that neither $\text{SREG} \leq_{\mathcal{M}_\infty} \text{SLIN}$ nor the first condition of $\text{SREG} \leq_{\mathcal{M}_\infty}^i \text{SLIN}$ is satisfied for any $i \in \{1, 2, 3\}$. Hence,

$$\text{SREG} \not\leq_{\mathcal{M}_\infty}^i \text{SLIN},$$

for each $i \in \{0, 1, 2, 3\}$.

Finally, in order to prove claim 5., let L be a finite language over an alphabet Σ . First, observe that we have both

$$\text{SREG}\tau(L) \leq |\Sigma| + 1 \quad \text{and} \quad \text{SLIN}\tau(L) \leq |\Sigma| + 1,$$

for all $\tau \in \{\text{cc}_\infty, \text{sc}_\infty\}$, as shown in Lemma 5.1.2. In the case of strict linear grammars, we have that

$$\text{SLIN}\tau(L) \geq \left\lfloor \frac{|\Sigma|}{2} + 1 \right\rfloor,$$

for $\tau \in \{\text{cc}_\infty, \text{sc}_\infty\}$, since each strict linear production contains at most two distinct letters from Σ . Consequently, for any sequence of finite languages $(L_n)_{n \geq 0}$, we have that

$$\lim_{n \rightarrow \infty} \frac{\text{SLIN}\tau(L_n)}{\text{SREG}\tau(L_n)} \geq \lim_{n \rightarrow \infty} \frac{\left\lfloor \frac{|\Sigma|}{2} + 1 \right\rfloor}{|\Sigma| + 1} \geq \frac{1}{2} \neq 0,$$

which shows that $\text{SLIN} \not\leq_{\{\text{cc}_\infty, \text{sc}_\infty\}}^2 \text{SREG}$. However, we also have that

$$\text{SLIN} \not\leq_{\{\text{cc}_\infty, \text{sc}_\infty\}}^3 \text{SREG},$$

because $\text{SLIN} \leq_{\{\text{cc}_\infty, \text{sc}_\infty\}}^3 \text{SREG}$ would imply $\text{SLIN} \leq_{\{\text{cc}_\infty, \text{sc}_\infty\}}^2 \text{SREG}$.

This finishes the prove of the theorem. \square

Remark. Note that the relations introduced in Definition 5.2.1 are defined based on (sequences of) finite languages w.r.t. arbitrary alphabets. This means that it is allowed to construct sequences of finite languages with growing alphabets in order to show that one of these relations holds.

If we amend Definition 5.2.1 in such a way that we require that all of the involved (sequences of) finite languages are defined over a fixed finite alphabet Σ , then we get four additional relations $\leq_{\Sigma, \tau}$, $\leq_{\Sigma, \tau}^1$, $\leq_{\Sigma, \tau}^2$, and $\leq_{\Sigma, \tau}^3$ for each measure type $\tau \in \mathcal{M}$. For instance, let $X, Y \in \Delta$ and $\tau \in \mathcal{M}$. Then $X \leq_{\Sigma, \tau}^1 Y$ holds if and only if there is a constant c such that

$$X\tau(L) \leq Y\tau(L) + c,$$

for all finite languages $L \subseteq \Sigma^*$, and there is a sequence of finite languages $(L_i)_{i \geq 0}$ where $L_i \subseteq \Sigma^*$ for each $i \geq 0$, such that

$$Y\tau(L_i) - X\tau(L_i) \geq i.$$

A close inspection of the proofs of Theorem 5.2.12 (1) and (2) reveals that we also have

$$CF \leq_{\{a,b,\$,c_\infty\}}^i \{SREG, REG\}, \quad LIN \leq_{\{a,b,\$,c_\infty\}}^i \{SREG, REG\}, \quad \text{and} \quad SLIN \leq_{\{a,b,\$,c_\infty\}}^i SREG$$

as well as

$$\{SREG, REG\} \not\leq_{\{a,b,\$,c_\infty\}}^i \{SLIN, LIN, CF\},$$

for all $i \in \{0, 1, 2, 3\}$. However, the proof of Theorem 5.2.12 (3) is not applicable in order to show the respective results for the relations defined w.r.t. a fixed alphabet. Therefore, we have to leave the following problems open:

Open Problem 5.2.13. Is there an alphabet Σ such that

$$CF \leq_{\Sigma, c_\infty}^i SLIN, \quad LIN \leq_{\Sigma, c_\infty}^i SLIN, \quad \text{and} \quad REG \leq_{\Sigma, \mathcal{M}_\infty}^i SREG$$

and

$$CF \leq_{\Sigma, \{cc_\infty, sc_\infty\}}^i \Gamma_s, \quad LIN \leq_{\Sigma, \{cc_\infty, sc_\infty\}}^i \Gamma_s, \quad \text{and} \quad SLIN \leq_{\Sigma, \{cc_\infty, sc_\infty\}}^j SREG,$$

for all $i \in \{0, 1, 2, 3\}$ and $j \in \{0, 1\}$? Δ

Furthermore, the proof of Theorem 5.2.12 (4) does not show that the respective results also hold for the relations w.r.t. a fixed alphabet. Thus, we have a further open problem:

Open Problem 5.2.14. Let $X \in \Gamma$ and $Y \in \Delta \setminus \{SREG\}$. Is there an alphabet Σ such that

$$SLIN \not\leq_{\Sigma, \mathcal{M}_\infty}^i X \quad \text{and} \quad SREG \not\leq_{\Sigma, \{cc_\infty, sc_\infty\}}^i Y,$$

for each $i \in \{0, 1, 2, 3\}$? Δ

It has turned out that the stated proof of [HW18b, Theorem 13] is not sufficient to prove the statement of [HW18b, Theorem 13]. More specifically, the arguments used in the proof have turned out to be sufficient only for the more restricted relations that are defined w.r.t. a fixed alphabet. This leads to the following result:

Theorem 5.2.15. Let Σ be a finite alphabet and $X, Y \in \Delta$. Then we have that

$$X \not\leq_{\Sigma, \{cc_\infty, sc_\infty\}}^i Y,$$

for all $i \in \{1, 2, 3\}$.

PROOF. Let L be an arbitrary finite language over the alphabet Σ . Then, from Lemma 5.1.2, we know that

$$X_{cc_\infty}(L) \leq |\Sigma| + 1 \quad \text{and} \quad X_{sc_\infty}(L) \leq |\Sigma| + 1$$

holds for all grammar types in Δ . As a consequence, $X \leq_{\Sigma, \{cc_\infty, sc_\infty\}}^i Y$ does not hold for any $i \in \{1, 2, 3\}$ and any $X, Y \in \Delta$. \square

We have now shown all results depicted in Figure 53 and a quick glance at this figure also reveals the problems that are still open w.r.t. the relations of Definition 5.2.1.

Open Problem 5.2.16. Fill out the unknown relations in Figure 53. Δ

5.3 Relating Measure Types

In this section, we are going to relate the measure types in \mathcal{M} w.r.t. a fixed grammar type from Δ based on four different relations that vary in strength. These relations are similar to those introduced in the previous section, however, now the roles of measure and grammar types are swapped. With the help of these relations, we can classify the difference between different measure types w.r.t. a fixed grammar type. As a byproduct of this classification, we show that any X -grammar, for $X \in \Delta$, that infinitely generates the language

$$T_n = \{w\$w\#w \mid w \in \{a, b\}^n\},$$

requires $\Omega(2^{n/4})$ many productions.

Now, let us formally define the above mentioned relations on measure types in \mathcal{M} .

Definition 5.3.1. Let $X \in \Delta$ and $\tau, \sigma \in \mathcal{M}$. Then we write

- $\tau \leq_X \sigma$ if and only if

$$X\tau(L) \leq X\sigma(L),$$

for all finite languages L ;

- $\tau \leq_X^1 \sigma$ if and only if there is a constant c such that

$$X\tau(L) \leq X\sigma(L) + c,$$

for all finite languages L , and there is a sequence of finite languages $(L_i)_{i \geq 0}$ such that

$$X\sigma(L_i) - X\tau(L_i) \geq i;$$

- $\tau \leq_X^2 \sigma$ if and only if there is a constant c such that

$$X\tau(L) \leq X\sigma(L) + c,$$

for all finite languages L , and there is a sequence of finite languages $(L_i)_{i \geq 0}$ such that

$$\lim_{i \rightarrow \infty} \frac{X\tau(L_i)}{X\sigma(L_i)} = 0;$$

- $\tau \leq_X^3 \sigma$ if and only if there is a constant c such that

$$X\tau(L) \leq X\sigma(L) + c,$$

for all finite languages L , and there is no function $f: \mathbb{N} \rightarrow \mathbb{N}$ such that

$$X\sigma(L) \leq f(X\tau(L)),$$

for all finite languages L .

We write $\tau =_X \sigma$ if both $\tau \leq_X \sigma$ and $\sigma \leq_X \tau$ hold. Moreover, for ease of presentation, we sometimes denote the relation \leq_X by \leq_X^0 . Let $i \in \{0, 1, 2, 3\}$ and $X \in \Delta$, then the relation \leq_X^i is said to be of *order* i .

Let $\mathcal{M}_1, \mathcal{M}_2 \subseteq \mathcal{M}$, $\Pi \subseteq \Delta$, and $i \in \{0, 1, 2, 3\}$. Then we write

$$\mathcal{M}_1 \leq_{\Pi}^i \mathcal{M}_2$$

if, for all $\tau_1 \in \mathcal{M}_1$, all $\tau_2 \in \mathcal{M}_2$, and all $X \in \Pi$, it holds that $\tau_1 \leq_X^i \tau_2$. Similarly, we write

$$\mathcal{M}_1 \not\leq_{\Pi}^i \mathcal{M}_2$$

if, for all $\tau_1 \in \mathcal{M}_1$, all $\tau_2 \in \mathcal{M}_2$, and all $X \in \Pi$, it holds that $\tau_1 \not\leq_X^i \tau_2$. In the case that \mathcal{M}_1 , \mathcal{M}_2 , or Π is a singleton, we omit braces, e.g., we write $\tau_1 \leq_X^i \tau_2$ instead of $\{\tau_1\} \leq_{\{X\}}^i \{\tau_2\}$.

Remark. Note that, for $i \in \{0, 1, 2, 3\}$,

$$\mathcal{M}_1 \not\leq_{\Pi}^i \mathcal{M}_2$$

being satisfied does not coincide with the fact that

$$\mathcal{M}_1 \leq_{\Pi}^i \mathcal{M}_2$$

does not hold. The latter does not hold as soon as we have that

$$\tau_1 \not\leq_X^i \tau_2$$

for at least one $\tau_1 \in \mathcal{M}_1$, at least one $\tau_2 \in \mathcal{M}_2$, or at least one $X \in \Pi$. On the other hand,

$$\mathcal{M}_1 \not\leq_{\Pi}^i \mathcal{M}_2$$

is only satisfied if

$$\tau_1 \not\leq_X^i \tau_2$$

holds for all $\tau_1 \in \mathcal{M}_1$, all $\tau_2 \in \mathcal{M}_2$, and all $X \in \Pi$.

In order to demonstrate that—for distinct measure types—the production complexity of the same finite language w.r.t. the same grammar type can vary quite immensely, consider, for instance, the language

$$L_n = \{a^i b^i c^i \mid 1 \leq i \leq n\},$$

where $n \geq 1$ is an integer. On the one hand, Theorem 4.1.6 shows that

$$\text{CFcc}(L_n) \leq 5, \quad \text{for each } n \geq 1,$$

but, on the other hand, as we have seen in Chapter 4 (see Equation 4.2), we have that

$$\text{CFc}(L_n) = n.$$

Since every grammar that generates a language is also a grammar that covers this language, we have that

$$\text{CFcc}(L) \leq \text{CFC}(L),$$

for all finite languages L . Putting things together, we obtain that there is a gap of highest order (in the sense of Definition 5.3.1) between the exact and the cover complexity w.r.t. context-free grammars, i.e.,

$$\text{cc} \leq_{\text{CF}}^3 \text{c}$$

holds.

Remark. Similarly to the relations between grammar types, we have that $\tau \leq_X^3 \sigma$ implies $\tau \leq_X^2 \sigma$, which, in turn, implies $\tau \leq_X^1 \sigma$. Moreover, the inequality $\tau \leq_X^3 \sigma$ holds if the first condition of its definition is satisfied and there is a sequence of finite languages $(L_i)_{i \geq 0}$ such that $X\tau(L_i) \leq k$, for some constant k , and $X\sigma(L_i) \geq i$.

Let $\tau, \sigma \in \mathcal{M}$ and $X \in \Delta$. Then we say that τ and σ are *incomparable* w.r.t. the relation \leq_X^i , for $i \in \{0, 1, 2, 3\}$, if we have both $\tau \not\leq_X^i \sigma$ and $\sigma \not\leq_X^i \tau$. Moreover, we say that τ and σ are *incomparable* w.r.t. the grammar type X if we have both $\tau \not\leq_X^i \sigma$ and $\sigma \not\leq_X^i \tau$, for each $i \in \{0, 1, 2, 3\}$.

Figure 55 depicts which kinds of relations between the measure types in \mathcal{M} hold w.r.t. the grammar types in Δ . Let $\tau, \sigma \in \mathcal{M}$ and $i_{\text{CF}}, i_{\text{LIN}}, i_{\text{REG}}, i_{\text{SLIN}}, i_{\text{SREG}}, j_{\text{CF}}, j_{\text{LIN}}, j_{\text{REG}}, j_{\text{SLIN}}, j_{\text{SREG}} \in \{0, 1, 2, 3\}$. Then a directed edge with label $(i_{\text{CF}}/j_{\text{CF}}, i_{\text{LIN}}/j_{\text{LIN}}, i_{\text{REG}}/j_{\text{REG}}, i_{\text{SLIN}}/j_{\text{SLIN}}, i_{\text{SREG}}/j_{\text{SREG}})$ from τ to σ , i.e.,

$$\tau \xrightarrow{(i_{\text{CF}}/j_{\text{CF}}, i_{\text{LIN}}/j_{\text{LIN}}, i_{\text{REG}}/j_{\text{REG}}, i_{\text{SLIN}}/j_{\text{SLIN}}, i_{\text{SREG}}/j_{\text{SREG}})} \sigma,$$

expresses that all of the relations

$$\tau \leq_{\text{CF}}^{i_{\text{CF}}} \sigma, \quad \tau \leq_{\text{LIN}}^{i_{\text{LIN}}} \sigma, \quad \tau \leq_{\text{REG}}^{i_{\text{REG}}} \sigma, \quad \tau \leq_{\text{SLIN}}^{i_{\text{SLIN}}} \sigma, \quad \tau \leq_{\text{SREG}}^{i_{\text{SREG}}} \sigma$$

and

$$\tau \not\leq_{\text{CF}}^{i'_{\text{CF}}} \sigma, \quad \tau \not\leq_{\text{LIN}}^{i'_{\text{LIN}}} \sigma, \quad \tau \not\leq_{\text{REG}}^{i'_{\text{REG}}} \sigma, \quad \tau \not\leq_{\text{SLIN}}^{i'_{\text{SLIN}}} \sigma, \quad \tau \not\leq_{\text{SREG}}^{i'_{\text{SREG}}} \sigma$$

as well as

$$\sigma \leq_{\text{CF}}^{j_{\text{CF}}} \tau, \quad \sigma \leq_{\text{LIN}}^{j_{\text{LIN}}} \tau, \quad \sigma \leq_{\text{REG}}^{j_{\text{REG}}} \tau, \quad \sigma \leq_{\text{SLIN}}^{j_{\text{SLIN}}} \tau, \quad \sigma \leq_{\text{SREG}}^{j_{\text{SREG}}} \tau$$

and

$$\sigma \not\leq_{\text{CF}}^{j'_{\text{CF}}} \tau, \quad \sigma \not\leq_{\text{LIN}}^{j'_{\text{LIN}}} \tau, \quad \sigma \not\leq_{\text{REG}}^{j'_{\text{REG}}} \tau, \quad \sigma \not\leq_{\text{SLIN}}^{j'_{\text{SLIN}}} \tau, \quad \sigma \not\leq_{\text{SREG}}^{j'_{\text{SREG}}} \tau,$$

for all $i'_X \in \{i_X + 1, i_X + 2, \dots, 3\}$ and all $j'_X \in \{j_X + 1, j_X + 2, \dots, 3\}$ with $X \in \Delta$, hold. Let $X \in \Delta$. Then the entry “ $-$ ” at the position of i_X (or j_X) just expresses that the relation $\tau \leq_X^i \sigma$ (or $\sigma \leq_X^i \tau$, respectively) does not hold for any $i \in \{0, 1, 2, 3\}$, i.e., $\tau \not\leq_X^i \sigma$ (or $\sigma \not\leq_X^i \tau$, respectively), for all $i \in \{0, 1, 2, 3\}$. Moreover, the entry “?” expresses that we do not know whether or not any relation of order $i \in \{1, 2, 3\}$ holds between the involved measure types. If any number $i_X \in \{0, 1, 2, 3\}$ (or $j_X \in \{0, 1, 2, 3\}$, respectively), for $X \in \Delta$, is underlined, then this means that the relation $\tau \leq_X^{i_X} \sigma$ (or $\sigma \leq_X^{j_X} \tau$, respectively) holds, but we do not

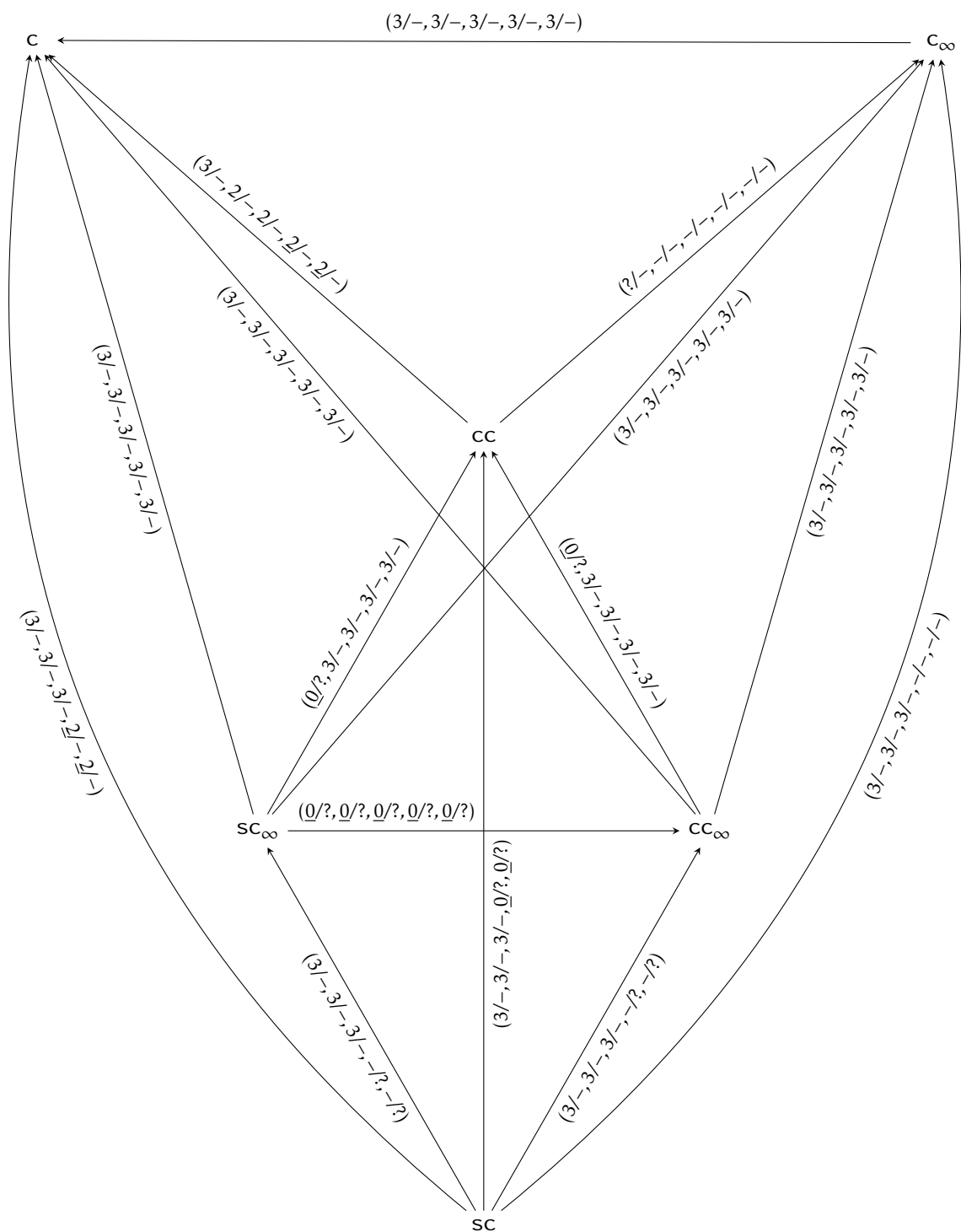


Figure 55: Relations between the measure types w.r.t. a fixed grammar type.

know whether or not any of the stronger relations $\tau \leq_X^{i'_X} \sigma$ (or $\sigma \leq_X^{j'_X} \tau$, respectively), for $i'_X \in \{i_X + 1, i_X + 2, \dots, 3\}$ and $j'_X \in \{j_X + 1, j_X + 2, \dots, 3\}$ holds. For instance,

$$\text{sc} \xrightarrow{(3/-, 3/-, 3/-, 0/? , 0/?)} \text{cc}$$

expresses that, for each $i \in \{0, 1, 2, 3\}$, both $\text{sc} \leq_{\Gamma}^i \text{cc}$ and $\text{cc} \not\leq_{\Gamma}^i \text{sc}$ are satisfied. Moreover, it expresses that the relation $\text{sc} \leq_{\Gamma_s}^0 \text{cc}$ holds, but we do not know whether or not $\text{sc} \leq_{\Gamma_s}^i \text{cc}$ holds for any $i \in \{1, 2, 3\}$. Finally, it also expresses that $\text{cc} \not\leq_{\Gamma_s}^0 \text{sc}$ holds, but we do not know whether or not $\text{cc} \leq_{\Gamma_s}^i \text{sc}$ holds for any $i \in \{1, 2, 3\}$.

We start with comparing the finite with the infinite complexity measures. Except for the scattered complexity, the infinite versions are more succinct than their finite counterparts w.r.t. the grammar types in Δ .

Lemma 5.3.2. *It holds that*

1. $c_{\infty} \leq_{\Delta} c$ and
2. $\text{cc}_{\infty} \leq_{\Delta} \text{cc}$, but we have
3. $\text{sc} \leq_{\Gamma} \text{sc}_{\infty}$.

PROOF. The first claim follows from Equation 5.2. The second claim can be shown as follows: let $L \subseteq \Sigma^{\leq \ell}$ and assume that G is a witness for $X_{\text{cc}}(L)$, i.e., G covers the finite language L and $X_{\text{cc}}(L) = |G|$. But then also

$$L \subseteq L(G) \cap \Sigma^{\leq \ell},$$

which implies

$$X_{\text{cc}_{\infty}}(L) \leq X_{\text{cc}}(L).$$

For the third relation, we argue as follows: in Theorem 4.1.6, it was shown that $Y_{\text{sc}}(L) = 1$ if L is non-empty, and $Y_{\text{sc}}(L) = 0$ if $L = \emptyset$. If $L \neq \emptyset$, we clearly have that $Y_{\text{sc}_{\infty}}(L) \geq 1$, and, on the other hand, if $L = \emptyset$, we have that $Y_{\text{sc}_{\infty}}(L) = 0$. Thus, we conclude that

$$Y_{\text{sc}}(L) \leq Y_{\text{sc}_{\infty}}(L),$$

for every finite language L . □

It is worth mentioning that the argumentation used in the proof of the first two claims of Lemma 5.3.2 does not apply to the third one. This can be seen as follows: consider the finite uniform language

$$L = \{a, b, c\}.$$

Then the regular grammar

$$G = (\{S\}, \{a, b, c\}, \{S \rightarrow abc\}, S)$$

witnesses

$$X_{\text{sc}}(L) = 1,$$

for $X \in \Gamma$, because L is a scattered sublanguage of $\{abc\}$, i.e.,

$$L \leq \{abc\} = L(G),$$

but

$$L(G) \cap \{a, b, c\}^{\leq 1} = \emptyset.$$

Thus, $|G| = 1$ cannot be used as an upper bound for $X_{\text{sc}_\infty}(L)$. Moreover,

$$X_{\text{sc}_\infty}(L) \geq 2,$$

since intersecting the language of any grammar consisting of a single production with the set of all words of length at most one cannot yield a scattered superlanguage of $\{a, b, c\}$. For the case of strict linear grammars, consider the grammar

$$G' = (\{S, A\}, \{a, b, c\}, \{S \rightarrow aAc, A \rightarrow b\}, S),$$

which serves as a witness for

$$\text{SLINsc}(L) = 2.$$

An analogous argument as above in conjunction with the type of restriction in strict linear grammars then shows that

$$\text{SLINsc}_\infty(L) \geq 3.$$

Hence,

$$\text{sc}_\infty \leq_{\Gamma \cup \{\text{SLIN}\}} \text{sc}$$

does not hold in general. Thus, we conclude:

Lemma 5.3.3. *It holds that $\text{sc}_\infty \not\leq_{\Gamma \cup \{\text{SLIN}\}} \text{sc}$.*

However, it is still open whether or not the result of Lemma 5.3.3 also holds for strict regular grammars.

Open Problem 5.3.4. Which one of $\text{sc}_\infty \leq_{\text{SREG}} \text{sc}$ and $\text{sc}_\infty \not\leq_{\text{SREG}} \text{sc}$ does hold? \triangle

Next, we prove—among other results—that the result of Lemma 5.3.2 (3) does neither hold for strict regular nor for strict linear grammars.

Lemma 5.3.5. *It holds that $\text{sc} \not\leq_{\Gamma_s} \mathcal{M}_\infty$.*

PROOF. Let $X \in \Gamma_s$ and, for every integer $n \geq 1$, let

$$L_n = \{a\}^{\leq 2n+1}.$$

Then, by Corollary 4.1.14 and Lemma 5.1.2, it follows that

$$X_{\text{sc}}(L_n) \geq n+1, \quad X_{\text{c}_\infty}(L_n) \leq 2, \quad X_{\text{cc}_\infty}(L_n) \leq 2, \quad \text{and} \quad X_{\text{sc}_\infty}(L_n) \leq 2.$$

Therefore, for any integer $n \geq 2$, we have that,

$$X_{sc}(L_n) > X_{c_\infty}(L_n), \quad X_{sc}(L_n) > X_{cc_\infty}(L_n), \quad \text{and} \quad X_{sc}(L_n) > X_{sc_\infty}(L_n),$$

i.e.,

$$sc \not\leq_{\Gamma_s} \mathcal{M}_\infty.$$

This proves the stated claim. \square

Next, for $X \in \Delta$, we compare the remaining X -complexities. Observe that, for finite languages L_1 and L_2 , $L_1 = L_2$ implies $L_1 \subseteq L_2$, which, in turn, implies $L_1 \leq L_2$. As an easy consequence, we deduce that the (infinite) X scattered complexity is more succinct than the (infinite) X cover complexity, and it is also easy to see that the (infinite) X cover complexity is more succinct than the (infinite) X -complexity. We summarise:

Lemma 5.3.6. *It holds that*

1. $sc \leq_\Delta cc \leq_\Delta c$ and
2. $sc_\infty \leq_\Delta cc_\infty \leq_\Delta c_\infty$.

For the measures cc and c_∞ , we show incomparability w.r.t. the \leq_Δ -relation. Before we can prove this result, we need two prerequisites. The first one is a lower bound on the infinite exact complexity of the language

$$T_n = \{w\$w\#w \mid w \in \{a, b\}^n\}.$$

The proof is similar to the one of Theorem 5.2.10 with the slight modification that we use grammars in 2-Greibach normal form³ instead of strict regular grammars, since we cannot apply the reasoning regarding the absence of ε -productions to context-free grammars in the case of the c_∞ -measure. The switch to 2-Greibach normal form gives rise to another variant of the triple construction that works for all grammar types in Δ .

Theorem 5.3.7. *Let $X \in \Delta$, $n \geq 1$ be an integer, G be an X -grammar generating a language $L \subseteq \Sigma^*$. Then there is an X -grammar G' in 2-Greibach normal form such that $L(G') = L \cap \Sigma^n$ and $|G'| \leq |G|^4 \cdot (n+1)^4$.*

PROOF. We proceed similarly as in the proof of Theorem 5.2.9. Assume that the X -grammar $G = (N, \Sigma, P, S)$ generates L . As a first step, we transform the grammar G into an equivalent X -grammar $G' = (N', \Sigma, P', S')$ in 2-Greibach normal form. According to [BK99], this transformation increases the number of productions by at most a polynomial of fourth degree, that is,

$$|G'| \leq |G|^4.$$

³A context-free grammar $G = (N, \Sigma, P, S)$ is in 2-Greibach normal form if all productions in P are of the form $A \rightarrow a$, $A \rightarrow aB$, $A \rightarrow aBC$, or $S \rightarrow \varepsilon$, where $A \in N$, $a \in \Sigma$, and $B, C \in N \setminus \{S\}$.

If $\varepsilon \in L$, then we can omit the production $S \rightarrow \varepsilon$ from G' , since $\varepsilon \notin L \cap \Sigma^n = (L \setminus \{\varepsilon\}) \cap \Sigma^n$. Therefore, all productions in P' are of the form

$$\begin{aligned} A &\rightarrow a, \\ A &\rightarrow aB, \text{ or} \\ A &\rightarrow aBC, \end{aligned}$$

where $a \in \Sigma$, $A \in N'$, and $B, C \in N \setminus \{S'\}$, i.e., G' is ε -free.

Now, assume that $\Sigma = \{a_1, a_2, \dots, a_n\}$ and consider the finite language Σ^n . Clearly, the DFA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ depicted in Figure 56 with $Q = \{0, 1, \dots, n\}$, initial state $q_0 = 0$, final state set $F = \{n\}$, and the transition function $\delta(i, a) = i + 1$, for $0 \leq i < n$ and $a \in \Sigma$, accepts the language Σ^n , i.e.,

$$L(\mathcal{A}) = \Sigma^n.$$

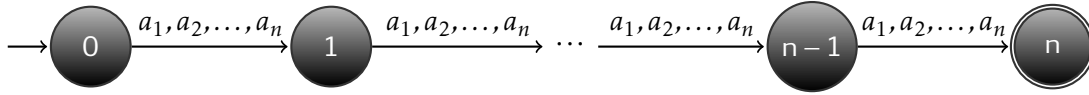


Figure 56: DFA \mathcal{A} for the finite language Σ^n .

We are now going to construct a grammar G'' in 2-Greibach normal form from G' and \mathcal{A} such that

$$L(G'') = L(G) \cap L(\mathcal{A}) = L \cap \Sigma^n.$$

To this end, let $G'' = (N'', \Sigma, P'', S'')$, where $S'' \notin N'$ is a new nonterminal,

$$N'' = \{[p, A, q] \mid p, q \in Q \text{ and } A \in N'\},$$

and

$$\begin{aligned} P'' = & \{[p, A, q] \rightarrow a \mid q = \delta(p, a), A \rightarrow a \in P', \text{ and } a \in \Sigma\} \\ & \cup \{[p, A, q] \rightarrow a[r, B, q] \mid r = \delta(p, a), q \in Q, A \rightarrow aB \in P', \text{ and } a \in \Sigma\} \\ & \cup \{[p, A, q] \rightarrow a[r, B, t][t, C, q] \mid r = \delta(p, a), q, t \in Q, A \rightarrow aBC \in P', \text{ and } a \in \Sigma\} \\ & \cup \{S'' \rightarrow [0, S', n]\}. \end{aligned}$$

Since \mathcal{A} consists of $n + 1$ states, i.e., $|Q| = n + 1$, and the productions of the form

$$[p, A, q] \rightarrow a[r, B, t][t, C, q]$$

make up the greatest portion of P'' , it follows that

$$|G''| \leq |G'| \cdot (n + 1)^4 \leq |G|^4 \cdot (n + 1)^4.$$

Note that $[0, S', n]$ generates words in $L(G') \cap L(\mathcal{A})$. This can be proved formally in a similar fashion as in the proof of Theorem 5.2.9 by taking also the productions of the form

$$[p, A, q] \rightarrow a[r, B, t][t, C, q]$$

into account (see also the proof of [Woo87, Theorem 9.1.3]).

To complete the proof, observe that for $w \in \Sigma^*$, it holds that

$$\begin{aligned} w \in L(G'') & \text{ iff } S'' \Rightarrow_{G''}^* w \\ & \text{ iff } [0, S', n] \Rightarrow_{G''}^* w \\ & \text{ iff } w \in L(G') \text{ and } w \in L(\mathcal{A}) \\ & \text{ iff } w \in L(G') \cap L(\mathcal{A}) = L \cap \Sigma^n, \end{aligned}$$

that is,

$$L(G'') = L \cap \Sigma^n.$$

This finishes the proof of the theorem. \square

The switch to 2-Greibach normal form in the above triple construction induces the fourth root in the lower bound on the infinite exact complexity of the language T_n .

Theorem 5.3.8. *Let $X \in \Delta$ and $n \geq 1$ an integer. Then $X_{C_\infty}(T_n) = \Omega(2^{n/4})$.*

PROOF. We proceed as in the proof of Theorem 5.2.10. From the CF-grammar $G = (N, \Sigma, P, S)$ that is a witness for $CF_{C_\infty}(T_n)$, i.e.,

$$T_n = L(G) \cap \Sigma^{\leq 3n+2} \quad \text{and} \quad |G| = CF_{C_\infty}(T_n),$$

we construct a context-free grammar for the language

$$L(G) \cap \Sigma^{3n+2}.$$

Now, assume to the contrary that

$$|G| = CF_{C_\infty}(T_n) = o(2^{n/4}),$$

for the CF-grammar G . Moreover, since $L(G) \cap \Sigma^{3n+2}$ is equal to T_n , we can apply Theorem 4.2.4 in order to obtain a lower bound on $|G|$. By Theorem 5.3.7, there is a context-free grammar $G' = (N', \Sigma, P', S')$ in 2-Greibach normal form satisfying

$$L(G') = L(G) \cap \Sigma^{3n+2} \quad \text{and} \quad |G'| \leq |G|^4 \cdot (3n+3)^4.$$

Then the context-free grammar G' generates T_n and satisfies

$$|G'| = o(2^{n/4}),$$

since

$$o(2^{n/4})^4 \cdot (3n+3)^4 = o(2^n).$$

This, however, is a contradiction to Theorem 4.2.4, which shows that

$$\text{CFc}(T_n) = \Omega(2^n),$$

and thus implies

$$\text{CFc}_\infty(T_n) = \Omega(2^{n/4}).$$

Therefore,

$$\text{X}_{\text{C}_\infty}(T_n) = \Omega(2^{n/4}),$$

for $\text{X} \in \Delta$, since context-free grammars are more succinct than all other considered grammar types w.r.t. the infinite X -complexity. \square

The second prerequisite consists of the finite context-free exact and cover complexity bounds of $|\Sigma| + 2$ for the language $\Sigma^{\leq \ell}$ as stated in Lemma 4.1.5.

Now, we are ready for the incomparability results:

Lemma 5.3.9. *It holds that*

1. $\text{c} \not\preceq_\Delta \{\text{cc}, \text{sc}, \text{c}_\infty, \text{cc}_\infty, \text{sc}_\infty\}$,
2. $\text{c}_\infty \not\preceq_\Delta \{\text{cc}, \text{sc}, \text{cc}_\infty, \text{sc}_\infty\}$,
3. $\text{cc} \not\preceq_\Delta \{\text{sc}, \text{c}_\infty, \text{cc}_\infty, \text{sc}_\infty\}$, and
4. $\text{cc}_\infty \not\preceq_\Delta \{\text{sc}, \text{sc}_\infty\}$.

PROOF. Let $\text{X} \in \Delta$. First, we show that both $\text{c} \not\preceq_\Delta \{\text{cc}, \text{sc}, \text{cc}_\infty, \text{sc}_\infty\}$ and $\text{c}_\infty \not\preceq_\Delta \{\text{cc}, \text{sc}, \text{cc}_\infty, \text{sc}_\infty\}$ hold. To this end, consider the language

$$T_n = \{w\$w\#w \mid w \in \{a, b\}^n\},$$

where $n \geq 1$ is an integer. Since $T_n \subseteq \{a, b, \$, \#\}^{3n+2}$, it follows from Theorems 4.1.6, 4.2.4, and 5.3.8, Corollary 4.2.5 as well as Lemmas 4.1.4, 4.1.5, and 5.1.2, that

$$\text{Xc}(T_n) \geq 2^n, \quad \text{Xsc}(T_n) \leq 12n + 8, \quad \text{and} \quad \text{Xc}_\infty(T_n) = \Omega(2^{n/4})$$

as well as

$$\text{Xcc}(T_n) \leq 12n + 8, \quad \text{Xcc}_\infty(T_n) \leq 5, \quad \text{and} \quad \text{Xsc}_\infty(T_n) \leq 5.$$

As a consequence, for a sufficiently large integer $n \geq 1$, we have that

$$\text{Xc}(T_n) > \text{Xcc}(T_n), \quad \text{Xc}(T_n) > \text{Xsc}(T_n), \quad \text{Xc}(T_n) > \text{Xcc}_\infty(T_n), \quad \text{and} \quad \text{Xc}(T_n) > \text{Xsc}_\infty(T_n)$$

hold. Moreover, we also have that

$$\text{Xc}_\infty(T_n) > \text{Xcc}(T_n) \quad \text{and} \quad \text{Xc}_\infty(T_n) > \text{Xsc}(T_n)$$

as well as

$$\text{Xc}_\infty(T_n) > \text{Xcc}_\infty(T_n) \quad \text{and} \quad \text{Xc}_\infty(T_n) > \text{Xsc}_\infty(T_n).$$

hold. Hence,

$$c \not\leq_{\Delta} cc, \quad c \not\leq_{\Delta} sc, \quad c \not\leq_{\Delta} cc_{\infty}, \quad \text{and} \quad c \not\leq_{\Delta} sc_{\infty}$$

as well as

$$c_{\infty} \not\leq_{\Delta} cc, \quad c_{\infty} \not\leq_{\Delta} sc, \quad c_{\infty} \not\leq_{\Delta} cc_{\infty}, \quad \text{and} \quad c_{\infty} \not\leq_{\Delta} sc_{\infty}.$$

In order to show $c \not\leq_{\Delta} c_{\infty}$ and $cc \not\leq_{\Delta} \mathcal{M}_{\infty}$, consider, for any integer $n \geq 2$, the language

$$L_n = \{a, b\}^{\leq n}.$$

On the one hand, by Lemma 5.1.2, we have that

$$X_{c_{\infty}}(L_n) \leq 3, \quad X_{cc_{\infty}}(L_n) \leq 3, \quad \text{and} \quad X_{sc_{\infty}}(L_n) \leq 3,$$

but, on the other hand, by Lemma 5.2.2, we have that

$$Y_c(L_n) \geq n+1 \quad \text{and} \quad Y_{cc}(L_n) \geq n+1,$$

for $Y \in \{\text{SREG}, \text{SLIN}, \text{REG}, \text{LIN}\}$. As a consequence, for any sufficiently large integer $n \geq 2$,

$$Y_{c_{\infty}}(L_n) < Y_c(L_n), \quad Y_{c_{\infty}}(L_n) < Y_{cc}(L_n), \quad Y_{cc_{\infty}}(L_n) < Y_{cc}(L_n), \quad \text{and} \quad Y_{sc_{\infty}}(L_n) < Y_{cc}(L_n).$$

From Lemma 4.1.5, it follows that

$$CF_c(L_n) \geq 4 \quad \text{and} \quad CF_{cc}(L_n) \geq 4,$$

i.e.,

$$CF_{c_{\infty}}(L_n) < CF_c(L_n) \quad \text{and} \quad CF_{c_{\infty}}(L_n) < CF_{cc}(L_n)$$

as well as

$$CF_{cc_{\infty}}(L_n) < CF_{cc}(L_n) \quad \text{and} \quad CF_{sc_{\infty}}(L_n) < CF_{cc}(L_n),$$

for each integer $n \geq 2$. Putting things together, we have that

$$c \not\leq_{\Delta} c_{\infty} \quad \text{and} \quad cc \not\leq_{\Delta} \mathcal{M}_{\infty}.$$

It remains to show $cc \not\leq_{\Delta} sc$ and claim 4. To this end, consider the language

$$L = \{\varepsilon, a\}.$$

On the one hand, for $X \in \Delta$, the X -grammar G with the single production

$$S \rightarrow a$$

clearly satisfies

$$L = \{\varepsilon, a\} \leq \{a\} = L(G) \quad \text{and} \quad L = \{\varepsilon, a\} \leq \{a\} = L(G) \cap \{a\}^{\leq 1}$$

and thus we have that

$$X_{sc}(L) = 1 \quad \text{and} \quad X_{sc_{\infty}}(L) = 1.$$

On the other hand, any X -grammar covering or infinitely covering L needs at least two productions, since there is no production that can generate both the empty word and a non-empty word without the help of another production. As a consequence,

$$X_{cc}(L) \geq 2 \quad \text{and} \quad X_{cc_\infty}(L) \geq 2.$$

Putting things together, we have that

$$cc \not\leq_\Delta sc, \quad cc_\infty \not\leq_\Delta sc, \quad \text{and} \quad cc_\infty \not\leq_\Delta sc_\infty$$

hold.

This finishes the proof of the lemma. \square

In the remainder of this section, we classify the relations between the measures in \mathcal{M} according to the taxonomy of Definition 5.3.1.

Theorem 5.3.10. *We have that*

$$cc \leq_{CF}^i c \quad \text{and} \quad cc \leq_{\Delta \setminus \{CF\}}^j c, \quad \text{but} \quad c \not\leq_\Delta^i cc \quad \text{and} \quad cc \not\leq_{\{REG, LIN\}}^3 c,$$

for all $i \in \{0, 1, 2, 3\}$ and all $j \in \{0, 1, 2\}$.

PROOF. First of all, note that

$$cc \leq_\Delta c \quad \text{and} \quad c \not\leq_\Delta cc \tag{5.15}$$

follow from Lemmas 5.3.6 and 5.3.9, respectively.

Consider, for each integer $n \geq 1$, the language

$$L_n = \{a^k b^k c^k \mid 1 \leq k \leq n\}.$$

From Equation 4.2, we know that

$$CFc(L_n) = n.$$

On the other hand, by Theorem 4.1.6, we have

$$CFcc(L_n) \leq 5.$$

Furthermore, by Equation 5.15, the first condition of $cc \leq_{CF}^i c$ is satisfied for each $i \in \{1, 2, 3\}$. Hence,

$$cc \leq_{CF}^i c,$$

for all $i \in \{1, 2, 3\}$. Note that, for $n \geq 5$, it holds that

$$CFc(L_n) - CFcc(L_n) \geq n - 5.$$

Thus, the first condition of $c \leq_{CF}^i cc$ is not satisfied for any $i \in \{1, 2, 3\}$, i.e.,

$$c \not\leq_{CF}^i cc,$$

for each $i \in \{1, 2, 3\}$.

Let $Z \in \Delta \setminus \{CF\}$ and, for each integer $n \geq 1$, consider the language

$$T_n = \{w\$w\#w \mid w \in \{a, b\}^n\}.$$

On the one hand, by Theorem 4.2.4 and Corollary 4.2.5, we have that

$$Zc(T_n) \geq 2^n.$$

On the other hand, since $\{a, b, \$, \#\}^{3n+2} \supseteq T_n$, we have that

$$Zcc(T_n) \leq 12n + 8$$

by Lemma 4.1.4. Hence,

$$\lim_{n \rightarrow \infty} \frac{Zcc(T_n)}{Zc(T_n)} = 0,$$

and thus, since by Equation 5.15, also the first condition of $cc \leq_{\Delta \setminus \{CF\}}^j c$ is satisfied for each $j \in \{1, 2\}$, we have that

$$cc \leq_{\Delta \setminus \{CF\}}^j c,$$

for all $j \in \{1, 2\}$.

Moreover, note that, for any integer $n \geq 8$, it holds that

$$Zc(T_n) - Zcc(T_n) \geq 2^n - 12n - 8 \geq n.$$

Therefore, the first condition of $c \leq_Z^i cc$ is not satisfied for any $i \in \{1, 2, 3\}$, i.e.,

$$c \not\leq_{\Delta \setminus \{CF\}}^i cc,$$

for each $i \in \{1, 2, 3\}$.

Finally, let $Y \in \{REG, LIN\}$, let L be an arbitrary finite language, and let $f: \mathbb{N} \rightarrow \mathbb{N}$ be a function defined by $x \mapsto 2^{x-1}$. Moreover, let G be a minimal Y -grammar with $L(G) \supseteq L$. By Lemma 3.3.1, we know that

$$|L| \leq |L(G)| \leq 2^{Ycc(L)-1}.$$

Furthermore, by simply listing all words in L with a trivial Y -grammar, it holds that

$$Yc(L) \leq |L|.$$

Thus,

$$Yc(L) \leq |L| \leq |L(G)| \leq 2^{Ycc(L)-1} = f(Ycc(L)).$$

This shows that there exists a function $f: \mathbb{N} \rightarrow \mathbb{N}$ such that

$$Yc(L) \leq f(Ycc(L)),$$

for all finite languages L . Hence,

$$cc \not\leq_{\{REG, LIN\}}^3 c.$$

This concludes the proof of the theorem. \square

Theorem 5.3.11. *We have that*

$$sc \leq_{\Gamma}^i c \quad \text{and} \quad sc \leq_{\Gamma_s}^j c, \quad \text{but} \quad c \not\leq_{\Delta}^i sc,$$

for all $i \in \{0, 1, 2, 3\}$ and all $j \in \{0, 1, 2\}$.

PROOF. First of all, note that

$$sc \leq_{\Delta} c \quad \text{and} \quad c \not\leq_{\Delta} sc \tag{5.16}$$

follow from Lemmas 5.3.6 and 5.3.9, respectively.

Let $X \in \Gamma$, $Z_s \in \Gamma_s$, and, for each integer $n \geq 1$, consider the language

$$T_n = \{w\$w\#w \mid w \in \{a, b\}^n\}.$$

On the one hand, by Theorem 4.2.4 and Corollary 4.2.5, we have that

$$Xc(T_n) = 2^n \quad \text{and} \quad Z_sc(T_n) \geq 2^n.$$

On the other hand, we have that

$$Xsc(T_n) = 1 \quad \text{and} \quad Z_ssc(T_n) \leq 12n + 8$$

by Theorem 4.1.6. Hence,

$$\lim_{n \rightarrow \infty} \frac{Z_ssc(T_n)}{Z_sc(T_n)} = 0,$$

and thus, since by Equation 5.16, also the first condition of $sc \leq_{\Gamma_s}^j c$ is satisfied for each $j \in \{1, 2\}$, we have that

$$sc \leq_{\Gamma_s}^j c,$$

for all $j \in \{1, 2\}$. The bounds

$$Xc(T_n) = 2^n \quad \text{and} \quad Xsc(T_n) = 1$$

together with Equation 5.16 also show that

$$sc \leq_{\Gamma}^i c$$

holds for all $i \in \{1, 2, 3\}$.

Furthermore, note that for $X_\Delta \in \Delta$ and any integer $n \geq 7$, due to Theorems 4.1.6 and 4.2.4 and Corollary 4.2.5, it holds that

$$X_\Delta c(T_n) \geq 2^n \quad \text{and} \quad X_\Delta sc(T_n) \leq 12n + 8.$$

Thus,

$$X_\Delta c(T_n) - X_\Delta sc(T_n) \geq 2^n - 12n - 8 \geq n.$$

Therefore, the first condition of $c \leq_\Delta^i sc$ is not satisfied for any $i \in \{1, 2, 3\}$, i.e.,

$$c \not\leq_\Delta^i sc,$$

for each $i \in \{1, 2, 3\}$. □

Theorem 5.3.12. *We have that*

$$sc \leq_\Gamma^i cc, \quad \text{but} \quad cc \not\leq_\Gamma^i sc,$$

for all $i \in \{0, 1, 2, 3\}$.

PROOF. Note that, from Lemmas 5.3.6 and 5.3.9, it follows that

$$sc \leq_\Gamma cc \quad \text{and} \quad cc \not\leq_\Gamma sc, \tag{5.17}$$

respectively.

Let $Y \in \{\text{REG}, \text{LIN}\}$ and, for each integer $n \geq 1$, consider the language

$$K_n = \{a, b\}^{\leq n}.$$

Then, since $|K_n| = 2^{n+1} - 1$, it follows that

$$Ycc(K_n) \geq n + 1$$

by Lemma 5.2.2. On the other hand,

$$Ysc(K_n) = 1$$

follows from Theorem 4.1.6. Since, by Equation 5.17, the first condition of $sc \leq_{\{\text{REG}, \text{LIN}\}}^i cc$ is satisfied for each $i \in \{1, 2, 3\}$, it follows that

$$sc \leq_{\{\text{REG}, \text{LIN}\}}^i cc,$$

for all $i \in \{1, 2, 3\}$. Moreover, note that, for any integer $n \geq 1$, it holds that

$$Ycc(K_n) - Ysc(K_n) \geq n.$$

Therefore, the first condition of $cc \leq_{\{REG, LIN\}}^i sc$ is not satisfied for any $i \in \{1, 2, 3\}$, i.e.,

$$cc \not\leq_{\{REG, LIN\}}^i sc,$$

for each $i \in \{1, 2, 3\}$.

For context-free grammars, consider, for each integer $n \geq 2$, the language

$$K'_n = \{a_1, a_2, \dots, a_n\}^{\leq n}.$$

Then, from Lemma 4.1.5 and Theorem 4.1.6, we get that

$$CFcc(K'_n) = n + 2 \quad \text{and} \quad CFsc(K'_n) = 1.$$

Thus, since, by Equation 5.17, the first condition of $sc \leq_{CF}^i cc$ is satisfied for each $i \in \{1, 2, 3\}$, we have that

$$sc \leq_{CF}^i cc,$$

for all $i \in \{1, 2, 3\}$. Moreover, note that, for any integer $n \geq 2$, it holds that

$$CFcc(K'_n) - CFsc(K'_n) \geq n + 1.$$

Therefore, the first condition of $cc \leq_{CF}^i sc$ is not satisfied for any $i \in \{1, 2, 3\}$, i.e.,

$$cc \not\leq_{CF}^i sc,$$

for each $i \in \{1, 2, 3\}$. □

Theorem 5.3.13. *We have that*

$$\mathcal{M}_\infty \leq_\Delta^i c, \quad \text{but} \quad c \not\leq_\Delta^i \mathcal{M}_\infty,$$

for all $i \in \{0, 1, 2, 3\}$.

PROOF. First, we prove that $c_\infty \leq_\Delta^i c$, but $c \not\leq_\Delta^i c_\infty$, for all $i \in \{0, 1, 2, 3\}$. From Lemmas 5.3.2 and 5.3.9, it follows that

$$c_\infty \leq_\Delta c \quad \text{and} \quad c \not\leq_\Delta c_\infty, \tag{5.18}$$

respectively.

We show that, for $X_\Delta \in \Delta$, there is a sequence of finite languages $(L_n)_{n \geq 1}$ and a constant c such that

$$X_\Delta c_\infty(L_n) \leq c \quad \text{and} \quad X_\Delta c(L_n) \geq n + 1.$$

For any integer $n \geq 1$, consider the languages

$$R_{2^n} = \{a^k b a^m \mid 0 \leq k + m \leq 2^n - 1\}, \quad R = \{a^k b a^m \mid k, m \geq 0\}, \quad \text{and} \quad K_n = \{a, b\}^{\leq n}.$$

In the proof of Equation 5.3, it was shown that, for each $X \in \Delta \setminus \{CF\}$ and each $n \geq 1$,

$$CF_{c_\infty}(R_{2^n}) \leq 4 \quad \text{and} \quad CF(R_{2^n}) \geq n + 1$$

and

$$X_{c_\infty}(K_n) \leq 3 \quad \text{and} \quad X_c(K_n) \geq n + 1.$$

Moreover, note that, by Equation 5.18, it follows that the first conditions of $c_\infty \leq_\Delta^i c$ are satisfied for each $i \in \{1, 2, 3\}$. Putting things together, we get that

$$c_\infty \leq_\Delta^i c,$$

for all $i \in \{1, 2, 3\}$.

Furthermore, note that, for any integer $n \geq 3$, it holds that

$$CF(R_{2^n}) - CF_{c_\infty}(R_{2^n}) \geq n - 3$$

and

$$X_c(K_n) - X_{c_\infty}(K_n) \geq n - 2.$$

Therefore, the first condition of $c \leq_\Delta^i c_\infty$ is not satisfied for any $i \in \{1, 2, 3\}$, i.e.,

$$c \not\leq_\Delta^i c_\infty,$$

for each $i \in \{1, 2, 3\}$.

Next, we prove that $\{cc_\infty, sc_\infty\} \leq_\Delta^i c$, but $c \not\leq_\Delta^i \{cc_\infty, sc_\infty\}$, for each $i \in \{0, 1, 2, 3\}$. From Lemmas 5.3.2, 5.3.6, and 5.3.9, it follows that

$$\{cc_\infty, sc_\infty\} \leq_\Delta c \quad \text{and} \quad c \not\leq_\Delta \{cc_\infty, sc_\infty\}. \quad (5.19)$$

For each integer $n \geq 1$, consider the language

$$L_n = \{a^k b^k c^k \mid 1 \leq k \leq n\}.$$

From Equation 4.2, we know that

$$X_\Delta c(L_n) \geq n,$$

for $X_\Delta \in \Delta$. Then, by Lemma 5.1.2,

$$X_\Delta cc_\infty(L_n) \leq 4 \quad \text{and} \quad X_\Delta sc_\infty(L_n) \leq 4.$$

Hence, since, by Equation 5.19, the first conditions of $\{cc_\infty, sc_\infty\} \leq_\Delta^i c$ are satisfied for each $i \in \{1, 2, 3\}$, it follows that

$$\{cc_\infty, sc_\infty\} \leq_\Delta^i c$$

holds for all $i \in \{1, 2, 3\}$.

Moreover, note that, for $\tau \in \{cc_\infty, sc_\infty\}$ and any integer $n \geq 4$, it holds that

$$X_\Delta c(L_n) - X_\Delta \tau(L_n) \geq n - 4.$$

Therefore, the first conditions of $c \leq_\Delta^i \{cc_\infty, sc_\infty\}$ are not satisfied for any $i \in \{1, 2, 3\}$, i.e.,

$$c \not\leq_\Delta^i \{cc_\infty, sc_\infty\},$$

for each $i \in \{1, 2, 3\}$. □

Theorem 5.3.14. *We have that*

$$cc_\infty \leq_{\Delta \setminus \{CF\}}^i cc, \quad \text{but} \quad cc \not\leq_{\Delta \setminus \{CF\}}^i cc_\infty,$$

for all $i \in \{0, 1, 2, 3\}$.

PROOF. Note that, from Lemmas 5.3.2 and 5.3.9, it follows that

$$cc_\infty \leq_{\Delta \setminus \{CF\}} cc \quad \text{and} \quad cc \not\leq_{\Delta \setminus \{CF\}} cc_\infty,$$

respectively.

Let $Z \in \Delta \setminus \{CF\}$ and, for any integer $n \geq 1$, consider the language

$$K_n = \{a, b\}^{\leq n}.$$

Since $|K_n| = 2^{n+1} - 1$, it follows that

$$Zcc(K_n) \geq n + 1,$$

by Lemma 5.2.2. Then, by Lemma 5.1.2, we get that

$$Zcc_\infty(K_n) \leq 3.$$

A similar argument as in the proof of Theorem 5.3.13 then shows that

$$cc_\infty \leq_{\Delta \setminus \{CF\}}^i cc$$

and

$$cc \not\leq_{\Delta \setminus \{CF\}}^i cc_\infty,$$

for each $i \in \{1, 2, 3\}$. □

Theorem 5.3.15. *We have that*

$$sc \not\leq_{\Gamma_s}^i c_\infty \quad \text{and} \quad c_\infty \not\leq_{\Gamma_s}^i sc,$$

for all $i \in \{0, 1, 2, 3\}$.

PROOF. Consider, for any integer $n \geq 1$, the unary language

$$D_n = \{a\}^{\leq 2n+1}$$

and let $Z_s \in \Gamma_s$. From Lemma 5.1.2 and Corollary 4.1.14, it follows that

$$Z_s c_\infty(D_n) \leq 2 \quad \text{and} \quad Z_s \text{sc}(D_n) \geq n + 1.$$

Therefore,

$$Z_s \text{sc}(D_n) - Z_s c_\infty(D_n) \geq n - 1,$$

and so neither $\text{sc} \leq_{\Gamma_s} c_\infty$ nor the first conditions of $\text{sc} \leq_{\Gamma_s}^i c_\infty$ are satisfied for any $i \in \{1, 2, 3\}$, i.e.,

$$\text{sc} \not\leq_{\Gamma_s}^j c_\infty,$$

for all $j \in \{0, 1, 2, 3\}$. Next, for each integer $n \geq 1$, consider the language

$$T_n = \{w\$w\#w \mid w \in \{a, b\}^n\}.$$

From Theorems 4.1.6 and 5.3.8, it follows that

$$Z_s \text{sc}(T_n) \leq 12n + 8 \quad \text{and} \quad Z_s c_\infty(T_n) = \Omega(2^{n/4}).$$

Therefore, for a sufficiently large integer $n \geq 1$, it holds that

$$Z_s c_\infty(T_n) > Z_s \text{sc}(T_n),$$

i.e.,

$$c_\infty \not\leq_{\Gamma_s} \text{sc}.$$

Moreover, since, for a sufficiently large integer $n \geq 1$, we have that

$$Z_s c_\infty(T_n) - Z_s \text{sc}(T_n) \geq n,$$

it follows that there is no constant c such that

$$Z_s c_\infty(T_n) \leq Z_s \text{sc}(T_n) + c,$$

for each $n \geq 1$. Hence,

$$c_\infty \not\leq_{\Gamma_s}^i \text{sc},$$

for all $i \in \{1, 2, 3\}$. Putting things together, we get that

$$c_\infty \not\leq_{\Gamma_s}^j \text{sc},$$

for each $j \in \{0, 1, 2, 3\}$. □

Theorem 5.3.16. *We have that*

$$sc \leq_{\Gamma}^i c_{\infty}, \quad \text{but} \quad c_{\infty} \not\leq_{\Gamma}^i sc$$

and

$$\{cc_{\infty}, sc_{\infty}\} \leq_{\Delta}^i c_{\infty}, \quad \text{but} \quad c_{\infty} \not\leq_{\Delta}^i \{cc_{\infty}, sc_{\infty}\},$$

for all $i \in \{0, 1, 2, 3\}$.

PROOF. Due to Lemmas 5.3.2 and 5.3.6, we have that

$$\{cc_{\infty}, sc_{\infty}\} \leq_{\Delta} c_{\infty} \quad \text{and} \quad sc \leq_{\Gamma} c_{\infty},$$

and thus also that the first conditions of

$$\{cc_{\infty}, sc_{\infty}\} \leq_{\Delta}^i c_{\infty} \quad \text{and} \quad sc \leq_{\Gamma}^i c_{\infty}$$

are satisfied for all $i \in \{1, 2, 3\}$. Moreover, Theorem 5.3.8 tells us that, for $X \in \Delta$, we have that

$$X_{c_{\infty}}(T_n) = \Omega(2^{n/4}).$$

On the other hand, by Theorem 4.1.6 and Lemma 5.1.2, for $Y \in \Gamma$ and all integers $n \geq 1$, it holds that

$$Y_{sc}(T_n) = 1, \quad X_{cc_{\infty}}(T_n) \leq 5, \quad \text{and} \quad X_{sc_{\infty}}(T_n) \leq 5.$$

This clearly shows that both

$$\{cc_{\infty}, sc_{\infty}\} \leq_{\Delta}^i c_{\infty} \quad \text{and} \quad sc \leq_{\Gamma}^i c_{\infty}$$

hold for each $i \in \{0, 1, 2, 3\}$.

Moreover, note that, for $\tau \in \{cc_{\infty}, sc_{\infty}\}$ and any sufficiently large integer $n \geq 1$, it holds that

$$X_{c_{\infty}}(T_n) - X_{\tau}(L_n) \geq n$$

and

$$Y_{c_{\infty}}(T_n) - Y_{sc}(T_n) \geq n.$$

Therefore, the first conditions of $c_{\infty} \leq_{\Delta}^i \{cc_{\infty}, sc_{\infty}\}$ and $c_{\infty} \leq_{\Gamma}^i sc$ are not satisfied for any $i \in \{1, 2, 3\}$. Furthermore, from Lemma 5.3.9, it follows that

$$c_{\infty} \not\leq_{\Delta} \{cc_{\infty}, sc_{\infty}\} \quad \text{and} \quad c_{\infty} \not\leq_{\Gamma} sc.$$

Putting things together, we obtain that

$$c_{\infty} \not\leq_{\Delta}^i \{cc_{\infty}, sc_{\infty}\} \quad \text{and} \quad c_{\infty} \not\leq_{\Gamma}^i sc,$$

for each $i \in \{0, 1, 2, 3\}$. □

Theorem 5.3.17. *We have that*

$$sc_{\infty} \leq_{\Delta \setminus \{CF\}}^i cc, \quad \text{but} \quad cc \not\leq_{\Delta \setminus \{CF\}}^i sc_{\infty},$$

for all $i \in \{0, 1, 2, 3\}$.

PROOF. From Lemmas 5.3.2, 5.3.6, and 5.3.9, it follows that

$$sc_{\infty} \leq_{\Delta \setminus \{CF\}} cc \quad \text{and} \quad cc \not\leq_{\Delta \setminus \{CF\}} sc_{\infty}.$$

Let $Z \in \Delta \setminus \{CF\}$ and, for each integer $n \geq 0$, consider the language

$$K_n = \{a, b\}^{\leq n}.$$

Then, on the one hand, since $|K_n| = 2^{n+1} - 1$, we have that

$$Zcc(K_n) \geq n + 1$$

by Lemma 5.2.2. On the other hand, by Lemma 5.1.2, we have that

$$Zsc_{\infty}(K_n) \leq 3.$$

A similar argument as in the proof of Theorem 5.3.13 then shows that

$$sc_{\infty} \leq_{\Delta \setminus \{CF\}}^i cc$$

and

$$cc \not\leq_{\Delta \setminus \{CF\}}^i sc_{\infty},$$

for each $i \in \{1, 2, 3\}$. □

Theorem 5.3.18. *We have that*

$$sc \leq_{\Gamma}^i \{cc_{\infty}, sc_{\infty}\}, \quad \text{but} \quad \{cc_{\infty}, sc_{\infty}\} \not\leq_{\Gamma}^i sc \quad \text{and} \quad sc \not\leq_{\Gamma_s}^i \{cc_{\infty}, sc_{\infty}\},$$

for all $i \in \{0, 1, 2, 3\}$.

PROOF. From Lemmas 5.3.2, 5.3.3, 5.3.5, 5.3.6, and 5.3.9, it follows that

$$sc \leq_{\Gamma} \{cc_{\infty}, sc_{\infty}\}, \quad \{cc_{\infty}, sc_{\infty}\} \not\leq_{\Gamma} sc, \quad \text{and} \quad sc \not\leq_{\Gamma_s} \{cc_{\infty}, sc_{\infty}\}. \quad (5.20)$$

Let $X \in \Gamma$, $Z_s \in \Gamma_s$, and, for each integer $n \geq 1$, consider the language

$$E_n = \{a_1, a_2, \dots, a_n\}.$$

Then, on the one hand, by Theorem 4.1.6, we have that

$$Xsc(E_n) = 1.$$

On the other hand,

$$Xcc_{\infty}(E_n) \geq n \quad \text{and} \quad Xsc_{\infty}(E_n) \geq n,$$

because every X-grammar with $|G| = X_{cc_\infty}(E_n)$ or $|G| = X_{sc_\infty}(E_n)$ has to satisfy

$$E_n \subseteq L(G) \cap \{a_1, a_2, \dots, a_n\}^{\leq 1} \quad \text{or} \quad E_n \subseteq L(G) \cap \{a_1, a_2, \dots, a_n\}^{\leq 1},$$

respectively. Therefore, it must be the case that $L(G) \supseteq \{a_1, a_2, \dots, a_n\}$, for otherwise both

$$E_n \not\subseteq L(G) \cap \{a_1, a_2, \dots, a_n\}^{\leq 1} \quad \text{and} \quad E_n \not\subseteq L(G) \cap \{a_1, a_2, \dots, a_n\}^{\leq 1}.$$

Hence,

$$X_{cc_\infty}(E_n) \geq n \quad \text{and} \quad X_{sc_\infty}(E_n) \geq n,$$

because any X-grammar covering the language $\{a_1, a_2, \dots, a_n\}$ needs at least n productions. As a consequence, since, by Equation 5.20, the first conditions of $sc \leq_\Gamma^i \{cc_\infty, sc_\infty\}$ are satisfied for each $i \in \{1, 2, 3\}$, it follows that

$$sc \leq_\Gamma^i \{cc_\infty, sc_\infty\}$$

holds for all $i \in \{1, 2, 3\}$.

Moreover, note that, for any integer $n \geq 1$, it holds that

$$X_{cc_\infty}(E_n) - X_{sc}(E_n) \geq n - 1 \quad \text{and} \quad X_{sc_\infty}(E_n) - X_{sc}(E_n) \geq n - 1.$$

Therefore, the first conditions of $\{cc_\infty, sc_\infty\} \leq_\Gamma^i sc$ are not satisfied for any $i \in \{1, 2, 3\}$, i.e.,

$$\{cc_\infty, sc_\infty\} \not\leq_\Gamma^i sc,$$

for each $i \in \{1, 2, 3\}$.

Finally, let $Z_s \in \Gamma_s$ and consider, for every integer $n \geq 1$, the language

$$B_n = \{a^{2n+1}\}.$$

From Corollary 4.1.14 and Lemma 5.1.2, it follows that

$$Z_s sc(B_n) \geq n + 1, \quad Z_s cc_\infty(B_n) \leq 2, \quad \text{and} \quad Z_s sc_\infty(B_n) \leq 2.$$

Thus, for any integer $n \geq 1$, it holds that

$$Z_s sc(B_n) - Z_s cc_\infty(B_n) \geq n - 1 \quad \text{and} \quad Z_s sc(B_n) - Z_s sc_\infty(B_n) \geq n - 1.$$

Therefore, the first conditions of $sc \leq_{\Gamma_s}^i \{cc_\infty, sc_\infty\}$ are not satisfied for any $i \in \{1, 2, 3\}$, i.e.,

$$sc \not\leq_{\Gamma_s}^i \{cc_\infty, sc_\infty\}$$

for each $i \in \{1, 2, 3\}$. □

Theorem 5.3.19. *We have that*

$$c_\infty \not\leq_\Delta^i cc \quad \text{and} \quad cc \not\leq_{\Delta \setminus \{CF\}}^i c_\infty,$$

for all $i \in \{0, 1, 2, 3\}$.

PROOF. From Lemma 5.3.9, it follows that

$$c_\infty \not\leq_\Delta cc \quad \text{and} \quad cc \not\leq_{\Delta \setminus \{CF\}} c_\infty.$$

Let $X_\Delta \in \Delta$, $Z \in \Delta \setminus \{CF\}$, and, for any integer $n \geq 3$, consider the language

$$K_n = \{a, b\}^{\leq n}.$$

Then, on the one hand, since $|K_n| = 2^{n+1} - 1$, we have that

$$Zcc(K_n) \geq n + 1$$

by Lemma 5.2.2. On the other hand, by Lemma 5.1.2, we have that

$$Zc_\infty(K_n) \leq 3.$$

A similar argument as in the proof of Theorem 5.3.13 then shows that

$$cc \not\leq_{\Delta \setminus \{CF\}}^i c_\infty,$$

for all $i \in \{1, 2, 3\}$.

Furthermore, recall the finite language

$$T_n = \{w\$w\#w \mid w \in \{a, b\}^n\}$$

and observe that, since $T_n \subseteq \{a, b, \$, \#\}^{3n+2}$, Lemmas 4.1.4 and 4.1.5 and Theorem 5.3.8 show that

$$CFcc(T_n) \leq 5, \quad Zcc(T_n) \leq 12n + 8, \quad \text{and} \quad X_\Delta c_\infty(T_n) = \Omega(2^{n/4}).$$

Hence, since, for any sufficiently large integer $n \geq 1$, we have that

$$X_\Delta c_\infty(T_n) - X_\Delta cc(T_n) \geq n,$$

it follows that the first condition of $c_\infty \leq_\Delta^i cc$ is not satisfied for any $i \in \{1, 2, 3\}$, i.e.,

$$c_\infty \not\leq_\Delta^i cc,$$

for all $i \in \{1, 2, 3\}$. □

Remark. Note that the relations introduced in Definition 5.3.1 are defined based on (sequences of) finite languages w.r.t. arbitrary alphabets. This means that it is allowed to construct sequences of finite languages with growing alphabets in order to show that one of these relations holds.

If we amend Definition 5.3.1 in such a way that we require that all of the involved (sequences of) finite languages are defined over a fixed finite alphabet Σ , then we get four additional relations $\leq_{\Sigma, X}$, $\leq_{\Sigma, X}^1$, $\leq_{\Sigma, X}^2$, and $\leq_{\Sigma, X}^3$ for each grammar type $X \in \Delta$. For

instance, let $\tau, \sigma \in \mathcal{M}$ and $X \in \Delta$. Then $\tau \leq_{\Sigma, X}^1 \sigma$ holds if and only if there is a constant c such that

$$X\tau(L) \leq X\sigma(L) + c,$$

for all finite languages $L \subseteq \Sigma^*$, and there is a sequence of finite languages $(L_i)_{i \geq 0}$, where $L_i \subseteq \Sigma^*$ for each $i \geq 0$, such that

$$X\sigma(L_i) - X\tau(L_i) \geq i.$$

A close inspection of the proof of Theorem 5.3.12 reveals that we also have

$$sc \leq_{\{a,b\}, \{REG, LIN\}}^i cc \quad \text{and} \quad cc \not\leq_{\{a,b\}, \{REG, LIN\}}^i sc,$$

for each $i \in \{0, 1, 2, 3\}$. However, the proof of the context-free case of Theorem 5.3.12 is not applicable in order to show the respective results for the relations defined w.r.t. a fixed alphabet. Therefore, we have—among others—the following additional results:

Theorem 5.3.20. *Let Σ be a finite alphabet. Then, for all $i \in \{1, 2, 3\}$, we have that*

1. $cc \not\leq_{\Sigma, CF}^i \{sc, cc_\infty, sc_\infty\}$ and $\{sc, cc_\infty, sc_\infty\} \not\leq_{\Sigma, CF}^i cc$,
2. $cc_\infty \not\leq_{\Sigma, \Delta}^i sc_\infty$ and $sc_\infty \not\leq_{\Sigma, \Delta}^i cc_\infty$, and
3. $sc \not\leq_{\Sigma, \Gamma}^i \{cc_\infty, sc_\infty\}$ and $\{cc_\infty, sc_\infty\} \not\leq_{\Sigma, \Gamma}^i sc$.

PROOF. We start with the proof of claim 1. From Theorem 4.1.6 and Lemma 5.1.2, we get that

$$CFcc(L) \leq |\Sigma| + 2, \quad CFsc(L) = 1, \quad CFcc_\infty(L) \leq |\Sigma| + 1, \quad \text{and} \quad CFsc_\infty(L) \leq |\Sigma| + 1,$$

for all finite languages L over Σ . As a consequence,

$$\tau \not\leq_{\Sigma, CF}^i \sigma,$$

for all $\tau, \sigma \in \{cc, sc, cc_\infty, sc_\infty\}$ and all $i \in \{1, 2, 3\}$. This finishes the proof of the first claim.

Next, we prove claim 2. To this end, let $X \in \Delta$. Then, from Lemma 5.1.2, we know that

$$Xcc_\infty(L) \leq |\Sigma| + 1 \quad \text{and} \quad Xsc_\infty(L) \leq |\Sigma| + 1,$$

for all finite languages L over Σ . As a consequence, we get that $\tau \leq_{\Sigma, \Delta}^i \sigma$ does *not* hold for all $\tau, \sigma \in \{cc_\infty, sc_\infty\}$ and all $i \in \{1, 2, 3\}$. This concludes the proof of the second claim.

Finally, we prove claim 3. From Theorem 4.1.6 and Lemma 5.1.2, we know that

$$Xsc(L) = 1, \quad Xcc_\infty(L) \leq |\Sigma| + 1, \quad \text{and} \quad Xsc_\infty(L) \leq |\Sigma| + 1,$$

for all finite languages L over Σ and $X \in \Gamma$. Thus, we get that $\tau \leq_{\Sigma, \Gamma}^i \sigma$ does *not* hold for all $\tau, \sigma \in \{sc, cc_\infty, sc_\infty\}$ and all $i \in \{1, 2, 3\}$, i.e.,

$$\tau \not\leq_{\Sigma, \Gamma}^i \sigma,$$

for each $\tau, \sigma \in \{sc, cc_\infty, sc_\infty\}$ and each $i \in \{1, 2, 3\}$.

This concludes the proof of the theorem. \square

To conclude this section, we give a list of the remaining open problems.

Open Problem 5.3.21. Let Σ be an alphabet and $X \in \Gamma_s$. Do we have that

$$sc_\infty \leq_{\Sigma, X}^i sc \quad \text{or} \quad sc_\infty \not\leq_{\Sigma, X}^j sc,$$

for $i, j \in \{1, 2, 3\}$? △

Open Problem 5.3.22. Let Σ be an alphabet and $X \in \Gamma_s$. Do we have that

$$cc_\infty \leq_{\Sigma, X}^i sc \quad \text{or} \quad cc_\infty \not\leq_{\Sigma, X}^j sc,$$

for $i, j \in \{1, 2, 3\}$? △

We have now shown all results depicted in Figure 55. Similarly to the previous section, the open problems w.r.t. the relations of Definition 5.3.1 can be extracted from Figure 55.

Open Problem 5.3.23. Fill out the unknown relations in Figure 55. △

6

CHAPTER

Bounds on Language Operations



classic problem in formal language theory is concerned with the closure of classes of languages under certain operations such as intersection, union, concatenation, etc. [HU79]. A natural way to gain a deeper understanding of the descriptive complexity of different descriptive systems is to study the behaviour of their complexities when different language operations are applied. A number of such investigations has been carried out with respect to both the number of states and the number of transitions needed by a deterministic and nondeterministic finite automaton in order to accept a given language [Bir92, YZS94, CSY00, CCISY01, Yu01, HK03, Ell04, GH05, JJS05, Jir05, DS07, HS07, HK09, GMR16]. It is worth mentioning that the operational exact production complexity of strict regular grammars is related to the operational transition complexity of NFAs. More precisely, every strict regular grammar G with p productions can be converted into an NFA \mathcal{A} with p transitions such that $L(\mathcal{A}) = L(G)$ and vice versa.¹ Within the realm of context-free grammars, among others, investigations regarding the exact nonterminal and the exact production complexity of different language operations for arbitrary languages have been carried out in [Geo96, DS08, DH12b, DH12a, Das17]. Particularly, in [DH12b], the authors discussed the range of applying, among others, the operations union and concatenation to two (or a finite number of) possibly infinite languages. For an r -ary operation \oplus under which the family of context-free languages is closed and natural numbers n_1, n_2, \dots, n_r , the authors of [DH12b] defined the range

$$g_{\oplus}(n_1, n_2, \dots, n_r)$$

as the set of all natural numbers k such that there are context-free languages L_i , for $1 \leq i \leq r$, such that

$$\text{Prod}(L_i) = n_i \quad \text{and} \quad \text{Prod}(\oplus(L_1, L_2, \dots, L_r)) = k,$$

¹See, e.g., the proofs of [HU69, Theorems 3.4 and 3.5].

where $\text{Prod}(L) = \min\{|G| \mid G \in \text{CF} \text{ and } L(G) = L\}$.² For unary operations, the ranges were determined completely, whereas the problems for union and concatenation were almost completely and partially solved, respectively. However, in the aforementioned paper, they did not restrict themselves to finite languages, but showed, e.g., that, for $X \in \Gamma$ and two finite ε -free languages L_1 and L_2 defined over disjoint alphabets, the following statement holds:

$$Xc(L_1 \cup L_2) = Xc(L_1) + Xc(L_2).$$

In this chapter, we will prove upper and lower bounds on the production complexity of applying the classic language operations *intersection*, *union*, and *concatenation* to finite languages. However, we will not restrict ourselves to pairs of finite languages which are defined over disjoint alphabets. The results of this section are summarised in Figure 61, where **bold font with gray background** means that we have matching upper and lower bounds w.r.t. a fixed alphabet, and non-bold font means that the bounds are matching only w.r.t. a growing alphabet. In the cases in which we have not yet obtained any bounds for arbitrary finite languages, the figure contains a question mark “?” as entry. Moreover, the entry “-” reflects the fact that the context-free cover complexity is a bounded measure (see Chapter 3) and thus it is of little interest to consider the bounds w.r.t. a fixed alphabet. This stems from the fact that the context-free cover complexity is always constant in the size of the alphabet of the covered language. What we mean by matching upper and lower bounds (or, equivalently, that the upper bound is *tight*) w.r.t. a complexity measure is that we can show the existence of a finite language whose complexity is at least as high as the obtained upper bound. For the remainder of this section, let $\Lambda = \Delta \setminus \{\text{CF}\}$.

Some of the results in this chapter have been published in [GHW18, HW18a, HW19].

	CF	LIN	REG	SLIN	SREG
$Xc(L_1 \cap L_2)$?	?	?	?	?
$Xc(L_1 \cup L_2)$	$c_1 + c_2$	$c_1 + c_2$	$c_1 + c_2$	$c_1 + c_2$	$c_1 + c_2$
$Xc(L_1 L_2)$	$c_1 + c_2$	$\min\{d_1 + c_2, c_1 + d_2\}$	$c_1 + c_2$	$\min\{d_1 + c_2, c_1 + d_2\}$	$c_1 + c_2$
$Xcc(L_1 \cap L_2)$	-	$\min\{c_{c,1}, c_{c,2}\}$	$\min\{c_{c,1}, c_{c,2}\}$	$\min\{c_{c,1}, c_{c,2}\}$	$\min\{c_{c,1}, c_{c,2}\}$
$Xcc(L_1 \cup L_2)$	-	$c_{c,1} + c_{c,2}$	$c_{c,1} + c_{c,2}$	$c_{c,1} + c_{c,2}$	$c_{c,1} + c_{c,2}$
$Xcc(L_1 L_2)$	-	$\min\{d_{c,1} + c_{c,2}, c_{c,1} + d_{c,2}\}$	$c_{c,1} + c_{c,2}$	$\min\{d_{c,1} + c_{c,2}, c_{c,1} + d_{c,2}\}$	$c_{c,1} + c_{c,2}$

Figure 61: Summary of descriptive complexity results for language operations. For $i \in \{1, 2\}$, let $c_i = Xc(L_i)$, $d_i = (\text{S})\text{REG}c(L_i)$, $c_{c,i} = Xcc(L_i)$, and $d_{c,i} = (\text{S})\text{REG}cc(L_i)$.

²As opposed to the definition of $\text{CFC}(L)$, the definition of $\text{Prod}(L)$ does not require that the language L is finite.

6.1 Intersection

The bound on the cover complexity of intersecting two finite languages L_1 and L_2 corresponds to the minimum of the cover complexities of L_1 and L_2 , and this bound is tight as shown in Theorems 6.1.1 and 6.1.2. For the exact complexity of intersection, the upper bound corresponds to the maximum of the exact complexities of L_1 and L_2 , provided that L_1 is a subset of L_2 . However, at this point, we do not know whether this bound is actually tight.

We start with the proof of the upper bound on the cover complexity of intersection.

Theorem 6.1.1. *Let $X \in \Lambda$ and $L_1, L_2 \in \mathcal{P}_{\text{fin}}(\Sigma^*)$ be finite languages. Then*

$$X_{\text{cc}}(L_1 \cap L_2) \leq \min\{X_{\text{cc}}(L_1), X_{\text{cc}}(L_2)\}.$$

PROOF. For $i \in \{1, 2\}$ and $X \in \Lambda$, let G_i be a minimal X -grammar with $L(G_i) \supseteq L_i$, i.e.,

$$|G_i| = X_{\text{cc}}(L_i).$$

Then, clearly,

$$L(G_i) \supseteq L_1 \cap L_2.$$

By choosing a grammar G_i out of G_1 and G_2 with

$$|G_i| = \min\{|G_1|, |G_2|\},$$

we get that

$$X_{\text{cc}}(L_1 \cap L_2) \leq \min\{X_{\text{cc}}(L_1), X_{\text{cc}}(L_2)\}.$$

This concludes the proof of the theorem. \square

In order to show that the bound of Theorem 6.1.1 is tight w.r.t. a fixed alphabet, we can use the fact that X_{cc} , for $X \in \Lambda$, is an unbounded complexity measure (see Corollary 3.3.2).

Theorem 6.1.2. *Let $X \in \Lambda$. Then there is an alphabet Σ such that for all integers $n_1, n_2 \geq 1$, there are finite languages $L_1, L_2 \in \mathcal{P}_{\text{fin}}(\Sigma^*)$ with $X_{\text{cc}}(L_1) \geq n_1$ and $X_{\text{cc}}(L_2) \geq n_2$ such that*

$$X_{\text{cc}}(L_1 \cap L_2) \geq \min\{X_{\text{cc}}(L_1), X_{\text{cc}}(L_2)\}.$$

PROOF. Let Σ be an arbitrary finite alphabet and let $n_1, n_2 \geq 1$ be integers such that, without loss of generality, $n_2 \geq n_1$. From Corollary 3.3.2 and by definition of unboundedness, it follows that there are languages $L_1, L_2 \in \mathcal{P}_{\text{fin}}(\Sigma^*)$ with

$$X_{\text{cc}}(L_1) \geq n_1 \quad \text{and} \quad X_{\text{cc}}(L_2) \geq n_2.$$

Next, we define the language

$$L'_2 = L_1 \cup L_2.$$

Then

$$X_{cc}(L'_2) \geq X_{cc}(L_1) \geq n_1,$$

for otherwise, since $L'_2 \supseteq L_1$, there would be a grammar covering L_1 with less than $X_{cc}(L_1)$ productions. Thus, we clearly have that

$$X_{cc}(L_1 \cap L'_2) = X_{cc}(L_1) = \min\{X_{cc}(L_1), X_{cc}(L'_2)\}.$$

This proves the stated claim. \square

For the exact complexity of the intersection of two finite languages we get a slightly different upper bound. Note that this upper bound only applies to pairs of finite languages where one language is contained in the other.

Theorem 6.1.3. *Let $X \in \Delta$ and $L_1, L_2 \in \mathcal{P}_{\text{fin}}(\Sigma^*)$ be finite languages with $L_1 \subseteq L_2$. Then*

$$X_c(L_1 \cap L_2) \leq \max\{X_c(L_1), X_c(L_2)\}.$$

PROOF. For $i \in \{1, 2\}$ and $X \in \Delta$, let G_i be a minimal X -grammar with $L(G_i) = L_i$, i.e.,

$$|G_i| = X_c(L_i).$$

Then, since

$$L_1 \subseteq L_2,$$

we have that

$$L(G_1) = L_1 = L_1 \cap L_2.$$

As a consequence,

$$X_c(L_1 \cap L_2) = X_c(L_1) \leq \max\{X_c(L_1), X_c(L_2)\}.$$

This finishes the proof of the theorem. \square

Since we do not know whether the bound obtained in the previous theorem is tight, we have to leave open the following problem.

Open Problem 6.1.4. Is the upper bound of Theorem 6.1.3 tight? Δ

It still remains to settle the exact complexity bounds of intersection if neither L_1 is contained in L_2 nor L_2 is contained in L_1 .

Open Problem 6.1.5. Let $X \in \Delta$ and $L_1, L_2 \in \mathcal{P}_{\text{fin}}(\Sigma^*)$ be finite languages with $L_1 \not\subseteq L_2$ and $L_2 \not\subseteq L_1$. Is there a tight upper bound on $X_c(L_1 \cap L_2)$ (w.r.t. a fixed alphabet)? Δ

6.2 Union

The bound on the (exact and cover) complexity of the union of two finite languages L_1 and L_2 corresponds to the sum of the respective complexities of L_1 and L_2 . In the case of the exact complexity, the bound is even tight w.r.t. a fixed alphabet for all grammar types in Δ . This is shown in Theorems 6.2.5 and Corollaries 6.2.2 and 6.2.9. We get a slightly different picture for the state of the tightness results in the case of cover complexity. Here, we have that the bound is only tight w.r.t. a fixed alphabet for strict regular, regular, and strict linear grammars, as shown in Theorems 6.2.1, 6.2.7, and 6.2.8.

We start with the proof of the upper bound on the cover complexity of union.

Theorem 6.2.1. *Let $X \in \Lambda$ and $L_1, L_2 \in \mathcal{P}_{\text{fin}}(\Sigma^*)$ be finite languages. Then*

$$\text{Xcc}(L_1 \cup L_2) \leq \text{Xcc}(L_1) + \text{Xcc}(L_2).$$

PROOF. Let $X \in \Lambda$ and, for $i \in \{1, 2\}$, let $G_i = (N_i, \Sigma_i, P_i, S_i)$ be a minimal X -grammar with $L(G_i) \supseteq L_i$, i.e.,

$$|G_i| = \text{Xcc}(L_i).$$

Furthermore, assume that $N_1 \cap N_2 = \emptyset$. By minimality of G_i and since, by Lemma 3.3.8, we can also assume that G_i is acyclic, S_i does not occur on the right-hand side of a production in P_i . Let $S \notin N_1 \cup N_2$ be a fresh nonterminal; we define $G = (N_1 \cup N_2 \cup S, \Sigma_1 \cup \Sigma_2, P, S)$, where

$$P = \{S \rightarrow \alpha \mid S_1 \rightarrow \alpha \in P_1 \text{ or } S_2 \rightarrow \alpha \in P_2\} \\ \cup \{A \rightarrow \alpha \in P_1 \mid A \neq S_1\} \cup \{A \rightarrow \alpha \in P_2 \mid A \neq S_2\}.$$

Clearly, we have

$$L(G) = L(G_1) \cup L(G_2) \supseteq L_1 \cup L_2 \quad \text{and} \quad |G| = |G_1| + |G_2| = \text{Xcc}(L_1) + \text{Xcc}(L_2),$$

that is,

$$\text{Xcc}(L_1 \cup L_2) \leq \text{Xcc}(L_1) + \text{Xcc}(L_2).$$

Moreover, $G_1, G_2 \in X$ implies $G \in X$. □

The construction defined in the proof of Theorem 6.2.1 can be used without any amendment in order to obtain an analogous upper bound on the exact complexity of finite languages that also holds for context-free grammars. We thus get:

Corollary 6.2.2. *Let $X \in \Delta$ and $L_1, L_2 \in \mathcal{P}_{\text{fin}}(\Sigma^*)$ be finite languages. Then*

$$\text{Xc}(L_1 \cup L_2) \leq \text{Xc}(L_1) + \text{Xc}(L_2).$$

In order to show that the upper bound obtained in Corollary 6.2.2 is tight for the non-strict grammar types, we need to show the following Lemma 6.2.3—which states that if we decompose a finite language that is incompressible w.r.t. a cover complexity measure into a disjoint union of two languages, then these two disjoint languages must be incompressible w.r.t. that cover complexity measure as well.

Lemma 6.2.3. *Let $X \in \{\text{REG}, \text{LIN}\}$ and $L \in \mathcal{P}_{\text{fin}}(\Sigma^*)$ be a finite language with $X_{\text{cc}}(L) = |L|$. Moreover, let $L_1, L_2 \in \mathcal{P}_{\text{fin}}(\Sigma^*)$ be disjoint finite languages such that $L = L_1 \cup L_2$. Then we have that*

$$X_{\text{cc}}(L_1) = |L_1| \quad \text{and} \quad X_{\text{cc}}(L_2) = |L_2|.$$

PROOF. Towards contradiction, assume, without loss of generality, that

$$X_{\text{cc}}(L_1) < |L_1|.$$

Furthermore, since there is a trivial grammar covering L_2 with $|L_2|$ productions, it clearly holds that

$$X_{\text{cc}}(L_2) \leq |L_2|.$$

Thus, from $L_1 \cup L_2 = L$ and Theorem 6.2.1, it follows that

$$X_{\text{cc}}(L) = X_{\text{cc}}(L_1 \cup L_2) \leq X_{\text{cc}}(L_1) + X_{\text{cc}}(L_2) < |L_1| + |L_2| = |L|.$$

But this means that

$$X_{\text{cc}}(L) < |L|,$$

a contradiction. □

Since, for $X \in \Gamma$ and all finite languages L , $X_{\text{cc}}(L) = |L|$ implies $X_{\text{c}}(L) = |L|$, the result of Lemma 6.2.3 also holds for the exact complexity measures w.r.t. the non-strict grammar types.

Corollary 6.2.4. *Let $X \in \Gamma$ and $L \in \mathcal{P}_{\text{fin}}(\Sigma^*)$ be a finite language with $X_{\text{c}}(L) = |L|$. Moreover, let $L_1, L_2 \in \mathcal{P}_{\text{fin}}(\Sigma^*)$ be disjoint finite languages such that $L = L_1 \cup L_2$. Then we have that*

$$X_{\text{c}}(L_1) = |L_1| \quad \text{and} \quad X_{\text{c}}(L_2) = |L_2|.$$

Recall the finite language

$$L_n = \{a^i b^i c^i \mid 1 \leq i \leq n\}$$

with

$$X_{\text{c}}(L_n) = |L_n| = n,$$

for $X \in \Gamma$ (see Equation 4.2). Since L_n can be defined in terms of the union of two disjoint finite languages, we can use it in conjunction with Corollary 6.2.4 in order to show that the upper bound of Corollary 6.2.2 is tight:

Theorem 6.2.5. *Let $X \in \Gamma$. Then there is an alphabet Σ such that for all integers $n_1, n_2 \geq 1$, there exist finite languages $L_1, L_2 \in \mathcal{P}_{\text{fin}}(\Sigma^*)$ with $X_{\text{c}}(L_1) = n_1$ and $X_{\text{c}}(L_2) = n_2$ such that*

$$X_{\text{c}}(L_1 \cup L_2) \geq X_{\text{c}}(L_1) + X_{\text{c}}(L_2).$$

PROOF. Let $X \in \Gamma$ and $\Sigma = \{a, b, c\}$. For all integers $n_1, n_2 \geq 1$, let

$$K = \{a^i b^i c^i \mid 1 \leq i \leq n_1 + n_2\}.$$

Moreover, let

$$K_1 = \{a^i b^i c^i \mid 1 \leq i \leq n_1\} \quad \text{and} \quad K_2 = K \setminus K_1.$$

Clearly, the intersection satisfies both

$$K_1 \cap K_2 = \emptyset \quad \text{and} \quad K_1 \cup K_2 = K.$$

Since, by Equation 4.2, both of the languages K and K_1 are CF-incompressible, we have that

$$X_C(K) = |K| = n_1 + n_2 \quad \text{and} \quad X_C(K_1) = |K_1| = n_1.$$

Thus, by Corollary 6.2.4, it follows that

$$X_C(K_2) = |K_2| = n_2 = |K| - |K_1|.$$

Consequently,

$$X_C(K_1 \cup K_2) = X_C(K) = n_1 + n_2 = X_C(K_1) + X_C(K_2).$$

This finishes the proof of the stated claim. \square

If we consider growing alphabets, then we can show that the above upper bound on the cover complexity of union is tight for all grammar types in Λ .

Theorem 6.2.6. *Let $X \in \Lambda$. Then, for all integers $n_1, n_2 \geq 1$, there exists a finite alphabet Σ and finite languages $L_1, L_2 \in \mathcal{P}_{\text{fin}}(\Sigma^*)$ with $X_{\text{cc}}(L_1) = n_1$ and $X_{\text{cc}}(L_2) = n_2$ such that*

$$X_{\text{cc}}(L_1 \cup L_2) \geq X_{\text{cc}}(L_1) + X_{\text{cc}}(L_2).$$

PROOF. Let $n_1, n_2 \geq 1$ be integers. Then, we define the alphabet

$$\Sigma = \{a_1, a_2, \dots, a_{n_1}, b_1, b_2, \dots, b_{n_2}\}$$

as well as the languages

$$L_1 = \{a_1, a_2, \dots, a_{n_1}\} \quad \text{and} \quad L_2 = \{b_1, b_2, \dots, b_{n_2}\},$$

i.e.,

$$L_1 \cup L_2 = \Sigma.$$

Moreover, we clearly have that

$$X_{\text{cc}}(L_1) = n_1 \quad \text{and} \quad X_{\text{cc}}(L_2) = n_2,$$

and that the language

$$L_1 \cup L_2$$

can only be covered by a trivial grammar. Therefore,

$$X_{\text{cc}}(L_1 \cup L_2) = n_1 + n_2 = X_{\text{cc}}(L_1) + X_{\text{cc}}(L_2).$$

This proves the stated claim. \square

Now, we prove—w.r.t. a fixed alphabet—a lower bound on the strict linear cover complexity of union that matches the upper bound of Theorem 6.2.1. To do so, we use the fact that in the case of strict linear grammars, there is a connection between the number of productions and the length of a longest word in the generated finite language (see Lemma 4.1.12).

Theorem 6.2.7. *There exists a finite alphabet Σ such that for all integers $n_1, n_2 \geq 1$, there are finite languages $L_1, L_2 \in \mathcal{P}_{\text{fin}}(\Sigma^*)$ with $\text{SLINcc}(L_1) = n_1$ and $\text{SLINcc}(L_2) = n_2$ such that*

$$\text{SLINcc}(L_1 \cup L_2) \geq \text{SLINcc}(L_1) + \text{SLINcc}(L_2).$$

PROOF. Let $\Sigma = \{a, b\}$ and, for $n_1, n_2 \geq 1$, we define the finite language

$$L = \{a^{2n_1-1}, b^{2n_2-1}\}.$$

Moreover, let

$$L_1 = \{a^{2n_1-1}\} \quad \text{and} \quad L_2 = \{b^{2n_2-1}\}.$$

Then we clearly have that

$$L = L_1 \cup L_2.$$

Furthermore, from Lemma 4.1.12, we get that

$$\text{SLINcc}(L_1) \geq \left\lfloor \frac{2n_1-1}{2} + 1 \right\rfloor = n_1 \quad \text{and} \quad \text{SLINcc}(L_2) \geq \left\lfloor \frac{2n_2-1}{2} + 1 \right\rfloor = n_2.$$

Next, consider the two strict linear grammars $G_1 = (N_1, \{a\}, P_1, S_1)$ and $G_2 = (N_2, \{b\}, P_2, S_2)$ with the following sets of productions P_1 and P_2 :

$$\begin{array}{ll} S_1 \rightarrow aA_2a & S_2 \rightarrow bB_2b \\ A_2 \rightarrow aA_3a & B_2 \rightarrow bB_3b \\ \vdots & \text{and} \quad \vdots \\ A_{n_1-1} \rightarrow aA_{n_1}a & B_{n_2-1} \rightarrow bB_{n_2}b \\ A_{n_1} \rightarrow a & B_{n_2} \rightarrow b, \end{array}$$

respectively. Clearly,

$$L(G_1) = L_1 \quad \text{and} \quad L(G_2) = L_2,$$

and therefore

$$\text{SLINcc}(L_1) \leq n_1 \quad \text{and} \quad \text{SLINcc}(L_2) \leq n_2.$$

Moreover, since the languages L_1 and L_2 do not share a common letter, there can be no production that is used to derive words from both L_1 and L_2 . Thus, we must have that

$$\text{SLINcc}(L) = \text{SLINcc}(L_1 \cup L_2) \geq \text{SLINcc}(L_1) + \text{SLINcc}(L_2).$$

This proves the stated claim. □

Finally, using segmented languages as defined in Section 4.3 and applying Theorem 4.3.11, we can show a lower bound on the (strict) regular cover complexity of union w.r.t. a fixed alphabet that matches the upper bound obtained in Theorem 6.2.1. At this point, one may ask why the regular cover-incompressible sequence constructed in [EH18] is not suitable for the proof of the following lower bound. The simple answer is that the union of two sequences of this kind does not necessarily result in a sequence of this kind again. The more general cover-incompressible sequence of Section 4.3 allows, however, to define a cover-incompressible sequence that corresponds to the union of two such cover-incompressible sequences.

Theorem 6.2.8. *Let $X \in \{\text{SREG}, \text{REG}\}$. Then there exists an alphabet Σ such that for all integers $n_1, n_2 \geq 1$, there are finite languages $L_1, L_2 \in \mathcal{P}_{\text{fin}}(\Sigma^*)$ with $X_{\text{cc}}(L_1) \geq n_1$ and $X_{\text{cc}}(L_2) \geq n_2$ such that*

$$X_{\text{cc}}(L_1 \cup L_2) \geq X_{\text{cc}}(L_1) + X_{\text{cc}}(L_2).$$

PROOF. Let $\Sigma_1 = \{a, b\}$, $\Sigma_2 = \{c, d\}$, and $\Sigma = \Sigma_1 \cup \Sigma_2$. Moreover, let, for each integer $n \geq 1$,

$$a_{n,1} = a_{n,2} = 2^{\lceil \log(n) \rceil} \quad \text{and} \quad a_n = a_{n,1} + a_{n,2}.$$

We define two sequences of finite languages $(L_{1,n})_{n \geq 1}$ and $(L_{2,n})_{n \geq 1}$ with

$$L_{1,n} = [\ell(n), k(n), \Sigma_1^{\ell(n)}] \quad \text{and} \quad L_{2,n} = [\ell(n), k(n), \Sigma_2^{\ell(n)}],$$

for each integer $n \geq 1$, based on:

$$\ell(n) := \lceil \log(n) \rceil = \lceil \log(a_{n,1}) \rceil = \lceil \log(a_{n,2}) \rceil$$

and

$$k(n) := \left\lceil \frac{9 \cdot a_n}{\ell(n) + 1} \right\rceil \geq \left\lceil \frac{9 \cdot a_{n,1}}{\ell(n) + 1} \right\rceil = \left\lceil \frac{9 \cdot a_{n,2}}{\ell(n) + 1} \right\rceil.$$

Recall that, for $i \in \{1, 2\}$,

$$L_{i,n} = [\ell(n), k(n), \Sigma_i^{\ell(n)}] = \{(sw)^{k(n)} \mid w \in \Sigma_i^{\ell(n)}\}.$$

Thus, clearly, $L_{i,n}$ is a $(k(n), \ell(n))$ -segmented language, for $i \in \{1, 2\}$. Now, let us consider the sequence of finite languages $(L_n)_{n \geq 1}$ with

$$L_n = [\ell(n), k(n), \Sigma_1^{\ell(n)} \cup \Sigma_2^{\ell(n)}].$$

Clearly,

$$\ell(n) \leq \lceil \log(a_n) \rceil,$$

and, moreover,

$$L_n = [\ell(n), k(n), \Sigma_1^{\ell(n)} \cup \Sigma_2^{\ell(n)}] = \{(sw)^{k(n)} \mid w \in \Sigma_1^{\ell(n)} \cup \Sigma_2^{\ell(n)}\}.$$

Thus, clearly, L_n is a $(k(n), \ell(n))$ -segmented language. By Theorem 4.3.11, the sequences $(L_{1,n})_{n \geq 1}$, $(L_{2,n})_{n \geq 1}$, and $(L_n)_{n \geq 1}$ are regular cover-incompressible. Furthermore, we have that

$$L_{1,n} \cap L_{2,n} = \emptyset \quad \text{and} \quad L_n = L_{1,n} \cup L_{2,n},$$

for all integers $n \geq 1$.

By definition of regular cover-incompressibility, without loss of generality, there is an integer M such that for all integers $n \geq M$, we have that

$$\text{REGcc}(L_{1,n}) = |L_{1,n}| \geq n \quad \text{and} \quad \text{REGcc}(L_{2,n}) = |L_{2,n}| \geq n$$

and

$$\text{REGcc}(L_n) = |L_n| = |L_{1,n}| + |L_{2,n}| \geq n + n.$$

Let $n_1, n_2 \geq 1$ and $m \geq M$ be integers such that $m \geq n_1, n_2$. Then

$$\text{REGcc}(L_{1,m}) \geq m \geq n_1 \quad \text{and} \quad \text{REGcc}(L_{2,m}) \geq m \geq n_2.$$

Consequently,

$$\text{REGcc}(L_m) = \text{REGcc}(L_{1,m} \cup L_{2,m}) = |L_m| = |L_{1,m}| + |L_{2,m}| = \text{REGcc}(L_{1,m}) + \text{REGcc}(L_{2,m}).$$

For the SREG-case, let $\Sigma = \{a, b\}$ and, for all integers $n_1, n_2 \geq 1$, we define the finite languages

$$L_1 = \{a^{n_1}\} \quad \text{and} \quad L_2 = \{b^{n_2}\}.$$

Moreover, let

$$L = L_1 \cup L_2.$$

Then, from Lemma 4.1.12, we get that

$$\text{SREGcc}(L_1) \geq n_1 \quad \text{and} \quad \text{SREGcc}(L_2) \geq n_2.$$

Since the words in L_1 and L_2 do not share a common letter, there can be no production that is used to derive words from both L_1 and L_2 . Thus, we must have that

$$\text{SREGcc}(L) = \text{SREGcc}(L_1 \cup L_2) \geq \text{SREGcc}(L_1) + \text{SREGcc}(L_2).$$

This proves the stated claim. □

Since every grammar that generates a given language also covers that language, we immediately get the following result from the proofs of Theorems 6.2.7 and 6.2.8.

Corollary 6.2.9. *Let $X \in \Gamma_s$. Then there exists an alphabet Σ such that for all integers $n_1, n_2 \geq 1$, there exist finite languages $L_1, L_2 \in \mathcal{P}_{\text{fin}}(\Sigma)$ with $Xc(L_1) \geq n_1$ and $Xc(L_2) \geq n_2$ such that*

$$Xc(L_1 \cup L_2) \geq Xc(L_1) + Xc(L_2).$$

6.3 Concatenation

In contrast to union, there is no uniform upper bound on the exact and cover complexity of concatenating two finite languages for all grammar types under consideration. The reason for this is that the method used to combine two given regular grammars into a new regular grammar which generates (or covers) the concatenation of their generated (or covered, respectively) languages does not necessarily give us a linear grammar again if we are given two linear grammars.

As a prerequisite for the proof of the upper bounds on the (strict) linear exact and cover complexity of the concatenation of two finite languages, we need to show that every left-linear grammar can be transformed into an equivalent right-linear grammar (and vice versa) without increasing the number of productions.

Proposition 6.3.1. *Let G be a left-linear grammar generating a finite language. Then there is a right-linear grammar G' with $L(G') = L(G)$ and $|G'| \leq |G|$.*

PROOF. Let $G = (N, \Sigma, P, S)$ be an arbitrary left-linear grammar generating a finite language. Since G generates a finite language, we can assume, in light of Lemma 3.3.8, that G is acyclic, i.e., S does not occur on the right-hand side of any production in G .

Now, we construct the grammar $G' = (N', \Sigma, P', S)$ from G as follows:

- if $S \rightarrow w \in P$, then $S \rightarrow w \in P'$,
- if $A \rightarrow w \in P$, then $S \rightarrow wA \in P'$,
- if $B \rightarrow Aw \in P$, then $A \rightarrow wB \in P'$, and
- if $S \rightarrow Aw \in P$, then $A \rightarrow w \in P'$,

where $A, B \in N$ and $w \in \Sigma^*$. Clearly, the grammar G' is right-linear and satisfies

$$|G'| \leq |G|.$$

Moreover,

$$L(G') = L(G)$$

follows from the proof of [Rév91, Theorem 3.7]. □

By inverting the steps of the procedure used in the proof of Proposition 6.3.1, it follows that we can also transform each right-linear into an equivalent left-linear grammar without increasing the number of productions:

Corollary 6.3.2. *Let G be a right-linear grammar generating a finite language. Then there is a left-linear grammar G' with $L(G') = L(G)$ and $|G'| \leq |G|$.*

Now, we are ready for the proof of the upper bounds on the exact and cover complexity of concatenation.

Theorem 6.3.3. *Let $X \in \{\text{SREG}, \text{REG}\}$ and $L_1, L_2 \in \mathcal{P}_{\text{fin}}(\Sigma^*)$ be finite languages. Then*

1. $X_{\text{cc}}(L_1 L_2) \leq X_{\text{cc}}(L_1) + X_{\text{cc}}(L_2)$,
2. $\text{LIN}_{\text{cc}}(L_1 L_2) \leq \min\{\text{REG}_{\text{cc}}(L_1) + \text{LIN}_{\text{cc}}(L_2), \text{LIN}_{\text{cc}}(L_1) + \text{REG}_{\text{cc}}(L_2)\}$,
3. $\text{SLIN}_{\text{cc}}(L_1 L_2) \leq \min\{\text{SREG}_{\text{cc}}(L_1) + \text{SLIN}_{\text{cc}}(L_2), \text{SLIN}_{\text{cc}}(L_1) + \text{SREG}_{\text{cc}}(L_2)\}$.

PROOF. For $i \in \{1, 2\}$, let $G_i = (N_i, \Sigma_i, P_i, S_i)$ be a minimal X -grammar with $L(G_i) \supseteq L_i$, i.e.,

$$|G_i| = X_{\text{cc}}(L_i),$$

and assume, without loss of generality, that $N_1 \cap N_2 = \emptyset$.

In order to show 1., let $X \in \{\text{SREG}, \text{REG}\}$ and define the X -grammar $G_X = (N_1 \cup N_2, \Sigma_1 \cup \Sigma_2, P_X, S_1)$, where

$$P_X = \{A \rightarrow wS_2 \mid A \rightarrow w \in P_1 \text{ and } w \in \Sigma^*\} \cup \{A \rightarrow \alpha \in P_1 \mid \alpha \notin \Sigma^*\} \cup P_2.$$

Note that in the strict regular case, the above construction of G_{SREG} preserves strict regularity because in a strict regular grammar the right-hand sides of productions without nonterminals are of length at most 1, and thus appending a single nonterminal to the right-hand side of such a production results again in a strict regular production. As a consequence, the above construction also works for strict regular grammars. Thus,

$$L(G_X) = L(G_1)L(G_2) \supseteq L_1 L_2 \quad \text{and} \quad |G_X| = |G_1| + |G_2| = X_{\text{cc}}(L_1) + X_{\text{cc}}(L_2),$$

which shows that

$$X_{\text{cc}}(L_1 L_2) \leq X_{\text{cc}}(L_1) + X_{\text{cc}}(L_2),$$

for all $L_1, L_2 \in \mathcal{P}_{\text{fin}}(\Sigma^*)$.

In order to show 2. and 3., let $G_{(\text{S})\text{REG},i} = (N_{(\text{S})\text{REG},i}, \Sigma_i, P_{(\text{S})\text{REG},i}, S_{(\text{S})\text{REG},i})$ and $G_{(\text{S})\text{LIN},i} = (N_{(\text{S})\text{LIN},i}, \Sigma_i, P_{(\text{S})\text{LIN},i}, S_{(\text{S})\text{LIN},i})$ be minimal (S)REG- and (S)LIN-grammars covering L_i , respectively, for $i \in \{1, 2\}$. That is,

$$L(G_{(\text{S})\text{REG},i}) \supseteq L_i \quad \text{and} \quad L(G_{(\text{S})\text{LIN},i}) \supseteq L_i$$

as well as

$$|G_{(\text{S})\text{REG},i}| = (\text{S})\text{REG}_{\text{cc}}(L_i) \quad \text{and} \quad |G_{(\text{S})\text{LIN},i}| = (\text{S})\text{LIN}_{\text{cc}}(L_i).$$

It remains to show that there are (strict) linear grammars G_1 and G_2 such that

$$L(G_1) \supseteq L_1 L_2 \quad \text{and} \quad L(G_2) \supseteq L_1 L_2,$$

and

$$|G_1| \leq (\text{S})\text{REG}_{\text{cc}}(L_1) + (\text{S})\text{LIN}_{\text{cc}}(L_2) \quad \text{and} \quad |G_2| \leq (\text{S})\text{LIN}_{\text{cc}}(L_1) + (\text{S})\text{REG}_{\text{cc}}(L_2).$$

To this end, assume, without loss of generality, that $N_{(S)REG,i} \cap N_{(S)LIN,j} = \emptyset$, for $i \neq j$. Furthermore, we assume that $G_{(S)REG,1}$ is a right-linear and $G_{(S)REG,2}$ is a left-linear grammar. This assumption is needed for the following definition of the grammars G_1 and G_2 , but it constitutes no restriction since, by Proposition 6.3.1 and Corollary 6.3.2, every right-linear grammar can be transformed into a left-linear one (and vice versa) without increasing the number of productions. Next, we define two (strict) linear grammars

$$G_1 = (N_{(S)REG,1} \cup N_{(S)LIN,2}, \Sigma_1 \cup \Sigma_2, P_1, S_{(S)REG,1})$$

and

$$G_2 = (N_{(S)LIN,1} \cup N_{(S)REG,2}, \Sigma_1 \cup \Sigma_2, P_2, S_{(S)REG,2}),$$

where

$$P_1 = \{A \rightarrow wS_{(S)LIN,2} \mid A \rightarrow w \in P_{(S)REG,1} \text{ and } w \in \Sigma^*\} \\ \cup \{A \rightarrow \alpha \in P_{(S)REG,1} \mid \alpha \notin \Sigma^*\} \cup P_{(S)LIN,2}$$

and

$$P_2 = \{A \rightarrow S_{(S)LIN,1}w \mid A \rightarrow w \in P_{(S)REG,2} \text{ and } w \in \Sigma^*\} \\ \cup \{A \rightarrow \alpha \in P_{(S)REG,2} \mid \alpha \notin \Sigma^*\} \cup P_{(S)LIN,1}.$$

Clearly,

$$|G_1| \leq |G_{(S)REG,1}| + |G_{(S)LIN,2}| \quad \text{and} \quad |G_2| \leq |G_{(S)LIN,1}| + |G_{(S)REG,2}|.$$

Let $w_i \in L_i$. Then, for $i \in \{1, 2\}$, we have that

$$S_{(S)REG,i} \Rightarrow^* w_i \quad \text{and} \quad S_{(S)LIN,i} \Rightarrow^* w_i.$$

Thus, by definition of G_1 and G_2 , we get

$$S_{(S)REG,1} \Rightarrow_{G_1}^* w_1 S_{(S)LIN,2} \Rightarrow_{G_1}^* w_1 w_2$$

and

$$S_{(S)REG,2} \Rightarrow_{G_2}^* S_{(S)LIN,1} w_2 \Rightarrow_{G_2}^* w_1 w_2,$$

that is,

$$w_1 w_2 \in L(G_i),$$

for $i \in \{1, 2\}$. Therefore,

$$L(G_i) \supseteq L_1 L_2,$$

for $i \in \{1, 2\}$. Thus, it follows that

$$|G_1| \leq (S)REGcc(L_1) + (S)LINcc(L_2) \quad \text{and} \quad |G_2| \leq (S)LINcc(L_1) + (S)REGcc(L_2).$$

Finally, since we have both

$$(S)LINcc(L_1 L_2) \leq |G_1| \quad \text{and} \quad (S)LINcc(L_1 L_2) \leq |G_2|,$$

we choose the grammar with the fewest number of productions out of G_1 and G_2 . This shows that both

$$\text{LINcc}(L_1 L_2) \leq \min\{\text{REGcc}(L_1) + \text{LINcc}(L_2), \text{LINcc}(L_1) + \text{REGcc}(L_2)\}$$

and

$$\text{SLINcc}(L_1 L_2) \leq \min\{\text{SREGcc}(L_1) + \text{SLINcc}(L_2), \text{SLINcc}(L_1) + \text{SREGcc}(L_2)\}$$

hold. □

The construction defined in the proof of Theorem 6.3.3 can be used without any amendment in order to obtain an analogous upper bound on the exact complexity of finite languages for the grammar types in Λ . Therefore, we only have to consider the case for context-free grammars in the proof of the following result.

Theorem 6.3.4. *Let $X \in \{\text{SREG}, \text{REG}, \text{CF}\}$ and $L_1, L_2 \in \mathcal{P}_{\text{fin}}(\Sigma^*)$ be finite languages. Then*

1. $Xc(L_1 L_2) \leq Xc(L_1) + Xc(L_2)$,
2. $\text{LINc}(L_1 L_2) \leq \min\{\text{REGc}(L_1) + \text{LINc}(L_2), \text{LINc}(L_1) + \text{REGc}(L_2)\}$,
3. $\text{SLINc}(L_1 L_2) \leq \min\{\text{SREGc}(L_1) + \text{SLINc}(L_2), \text{SLINc}(L_1) + \text{SREGc}(L_2)\}$.

PROOF. For $i \in \{1, 2\}$, let $G_i = (N_i, \Sigma_i, P_i, S_i)$ be a minimal context-free grammar with $L(G_i) = L_i$, i.e.,

$$|G_i| = \text{CFc}(L_i).$$

Assume, without loss of generality, $N_1 \cap N_2 = \emptyset$, and define the grammar $G = (N_1 \cup N_2, \Sigma_1 \cup \Sigma_2, P, S_1)$, where

$$P = \{S_1 \rightarrow \alpha S_2 \mid S_1 \rightarrow \alpha \in P_1\} \cup \{A \rightarrow \alpha \in P_1 \mid A \neq S_1\} \cup P_2.$$

Clearly, G is a context-free grammar with

$$L(G) = L(G_1)L(G_2) = L_1 L_2 \quad \text{and} \quad |G| = |G_1| + |G_2| = \text{CFc}(L_1) + \text{CFc}(L_2).$$

The cases for the remaining grammar types are an immediate consequence of the proof of Theorem 6.3.3. □

Now, we show that a grammar covering the concatenation of two disjoint alphabets (each containing at least two letters) needs at least as many productions as there are elements in their (disjoint) union. This lemma will play an important role in the proof of Theorem 6.3.6.

Lemma 6.3.5. *Let Σ_1 , Σ_2 , and Σ be finite alphabets with $\Sigma = \Sigma_1 \uplus \Sigma_2$ and $|\Sigma_1|, |\Sigma_2| \geq 2$. Then, for all context-free grammars G with $L(G) \supseteq \Sigma_1 \Sigma_2$, we have that $|G| \geq |\Sigma_1| + |\Sigma_2|$.*

PROOF. In light of Lemma 3.3.8, we can assume, without loss of generality, that all grammars in this proof are acyclic. We proceed by induction on $|\Sigma|$.

- **Base case:** Let $|\Sigma| = 4$, i.e., $|\Sigma_1| = |\Sigma_2| = 2$, and assume, without loss of generality, that

$$\Sigma = \{a_1, a_2\} \uplus \{b_1, b_2\}.$$

Then

$$\Sigma_1 \Sigma_2 = \{a_1 b_1, a_1 b_2, a_2 b_1, a_2 b_2\}.$$

Towards contradiction, assume that there is some context-free grammar $G = (N, \Sigma, P, S)$ with

$$L(G) \supseteq \Sigma_1 \Sigma_2 \quad \text{and} \quad |G| \leq 3.$$

Clearly, G cannot be a trivial grammar, for otherwise G could not cover $\Sigma_1 \Sigma_2$. Thus, we can assume that G contains at least two distinct nonterminals S and A . By Lemma 4.2.1, it follows that there are productions

$$A \rightarrow v_1, A \rightarrow v_2 \in P \text{ with } v_1, v_2 \in (N \cup \Sigma)^* \text{ and } v_1 \neq v_2,$$

which means that $|G| \geq 3$. Hence, P must be of the form

$$\{S \rightarrow \alpha, A \rightarrow v_1, A \rightarrow v_2\},$$

where $\alpha \in (N \cup \Sigma)^*$ and $v_1, v_2 \in \Sigma^*$, as G is acyclic. We distinguish the following cases:

If $\alpha = A$, then $|L(G)| = 2$, i.e., G cannot cover $\Sigma_1 \Sigma_2$.

If $\alpha = A^n$, for $n \geq 2$, then we further distinguish the following cases:

1. If $v_1 = av'_1$ and $v_2 = a'v'_2$, for $a, a' \in \Sigma_1$ and $v'_1, v'_2 \in \Sigma^*$. In this case, we cannot derive words of length 2 ending with some $b \in \Sigma_2$ (even if both v'_1 and v'_2 do so).
2. If $v_1 = bv'_1$ and $v_2 = b'v'_2$, for $b, b' \in \Sigma_2$ and $v'_1, v'_2 \in \Sigma^*$. In this case, we can only derive words starting with b or b' , but these kinds of words do not occur in $\Sigma_1 \Sigma_2$.
3. If $v_1 = av'_1$ and $v_2 = bv'_2$, for $a \in \Sigma_1$, $b \in \Sigma_2$, and $v'_1, v'_2 \in \Sigma^*$. In this case, we can only derive words starting with a fixed $a \in \Sigma_1$ or $b \in \Sigma_2$. As a consequence, we cannot derive words in $\Sigma_1 \Sigma_2$ that start with some $a' \in \Sigma_1$ such that $a \neq a'$.
4. If $v_1 = bv'_1$ and $v_2 = av'_2$, for $a \in \Sigma_1$, $b \in \Sigma_2$, and $v'_1, v'_2 \in \Sigma^*$. Symmetric to the previous case.
5. If $v_1 = \varepsilon$ and $v_2 = cv'_2$, for $c \in \Sigma$ and $v'_2 \in \Sigma^*$. In this case, we can only derive words starting with a fixed $c \in \Sigma$. As a consequence, we cannot derive words in $\Sigma_1 \Sigma_2$ that start with some $a \in \Sigma_1$ such that $a \neq c$.
6. If $v_1 = cv'_1$ and $v_2 = \varepsilon$, for $c \in \Sigma$ and $v'_1 \in \Sigma^*$. Symmetric to the previous case.

If α has $w' \in \Sigma^+$ as subword, then G cannot derive all words occurring in $\Sigma_1 \Sigma_2$, because there is no $w' \in \Sigma^+$ which is a subword of all $w \in \Sigma_1 \Sigma_2$. Hence, we have that $|G| \geq 4$.

This concludes the base case.

- **Induction step:** Assume, without loss of generality, that $\Sigma = \Sigma_1 \uplus \Sigma_2$ with

$$\Sigma_2 = \Sigma'_2 \uplus \{b_{n+1}\},$$

where

$$|\Sigma_1| = m \quad \text{and} \quad |\Sigma_2| = n + 1.$$

Towards contradiction, assume that there is some CF-grammar $G = (N, \Sigma, P, S)$ with

$$L(G) \supseteq \Sigma_1 \Sigma_2 \quad \text{and} \quad |G| < m + n + 1.$$

Next, we define the language

$$L' = \Sigma_1 \Sigma'_2$$

and let

$$\Sigma(L') = \Sigma_1 \uplus \Sigma'_2$$

denote the alphabet induced by the words in L' . Clearly,

$$|\Sigma(L')| = m + n,$$

and we can apply the induction hypothesis to obtain that for any CF-grammar G' with $L(G') \supseteq L'$, we have that $|G'| \geq m + n$.

Let $G'' = (N, \Sigma \setminus \{b_{n+1}\}, P'', S)$ be a CF-grammar obtained from G by defining

$$P'' = P \setminus \{A \rightarrow \alpha_1 b_{n+1} \alpha_2 \in P \mid \alpha_1, \alpha_2 \in (N \cup \Sigma)^*\}.$$

Then it follows that $L(G'') \supseteq L'$, since $b_{n+1} \notin \Sigma(L')$, and any production in which b_{n+1} occurs on the right-hand side can only be used to derive words containing b_{n+1} .

Note that any grammar covering $\Sigma_1 \Sigma_2$ must contain at least one production whose right-hand side contains the letter b_{n+1} . Thus,

$$|G''| < m + n,$$

which contradicts the induction hypothesis. The case that $\Sigma = \Sigma_1 \uplus \Sigma_2$ with

$$\Sigma_1 = \Sigma'_1 \uplus \{a_{n+1}\},$$

where

$$|\Sigma_1| = m + 1 \quad \text{and} \quad \Sigma_2 = n$$

can be shown using an analogous argument.

This finishes the proof of the Lemma. \square

If we consider growing alphabets, then we are able to show that the bounds of Theorem 6.3.3 are tight.

Theorem 6.3.6. *Let $X \in \{\text{SREG}, \text{REG}\}$. Then, for all integers $n_1, n_2 \geq 2$, there is a finite alphabet Σ and finite languages $L_1, L_2 \in \mathcal{P}_{\text{fin}}(\Sigma^*)$ with $X_{\text{cc}}(L_1) = n_1$ and $X_{\text{cc}}(L_2) = n_2$ such that*

1. $X_{\text{cc}}(L_1 L_2) \geq X_{\text{cc}}(L_1) + X_{\text{cc}}(L_2)$,
2. $\text{LIN}_{\text{cc}}(L_1 L_2) \geq \min\{\text{REG}_{\text{cc}}(L_1) + \text{LIN}_{\text{cc}}(L_2), \text{LIN}_{\text{cc}}(L_1) + \text{REG}_{\text{cc}}(L_2)\}$,
3. $\text{SLIN}_{\text{cc}}(L_1 L_2) \geq \min\{\text{SREG}_{\text{cc}}(L_1) + \text{SLIN}_{\text{cc}}(L_2), \text{SLIN}_{\text{cc}}(L_1) + \text{SREG}_{\text{cc}}(L_2)\}$.

PROOF. Let $X \in \{\text{SREG}, \text{REG}\}$, $n_1, n_2 \geq 2$ be integers, and define the alphabet

$$\Sigma = \{a_1, a_2, \dots, a_{n_1}, b_1, b_2, \dots, b_{n_2}\}$$

as well as the languages

$$L_1 = \{a_1, a_2, \dots, a_{n_1}\} \quad \text{and} \quad L_2 = \{b_1, b_2, \dots, b_{n_2}\}.$$

Then, clearly, we have that

$$X_{\text{cc}}(L_1) = \text{LIN}_{\text{cc}}(L_1) = \text{SLIN}_{\text{cc}}(L_1) = n_1 \quad \text{and} \quad X_{\text{cc}}(L_2) = \text{LIN}_{\text{cc}}(L_2) = \text{SLIN}_{\text{cc}}(L_2) = n_2.$$

Thus, since every X -grammar is context-free, we have, by Lemma 6.3.5, that

$$X_{\text{cc}}(\Sigma) = X_{\text{cc}}(L_1 L_2) \geq n_1 + n_2 = X_{\text{cc}}(L_1) + X_{\text{cc}}(L_2)$$

as well as

$$\text{LIN}_{\text{cc}}(L_1 L_2) \geq n_1 + n_2 = \min\{\text{REG}_{\text{cc}}(L_1) + \text{LIN}_{\text{cc}}(L_2), \text{LIN}_{\text{cc}}(L_1) + \text{REG}_{\text{cc}}(L_2)\}$$

and

$$\text{SLIN}_{\text{cc}}(L_1 L_2) \geq n_1 + n_2 = \min\{\text{SREG}_{\text{cc}}(L_1) + \text{SLIN}_{\text{cc}}(L_2), \text{SLIN}_{\text{cc}}(L_1) + \text{SREG}_{\text{cc}}(L_2)\}.$$

This proves the stated claim. \square

Since Lemma 6.3.5 can also be applied to the exact X -complexity, for all $X \in \Delta$, we get the following result by following the proof of Theorem 6.3.6.

Corollary 6.3.7. *Let $X \in \{\text{SREG}, \text{REG}, \text{CF}\}$. Then, for all integers $n_1, n_2 \geq 2$, there is a finite alphabet Σ and finite languages $L_1, L_2 \in \mathcal{P}_{\text{fin}}(\Sigma^*)$ with $X_{\text{c}}(L_1) = n_1$ and $X_{\text{c}}(L_2) = n_2$ such that*

1. $X_{\text{c}}(L_1 L_2) \geq X_{\text{c}}(L_1) + X_{\text{c}}(L_2)$,

2. $\text{LINc}(L_1 L_2) \geq \min\{\text{REGc}(L_1) + \text{LINc}(L_2), \text{LINc}(L_1) + \text{REGc}(L_2)\},$
3. $\text{SLINc}(L_1 L_2) \geq \min\{\text{SREGc}(L_1) + \text{SLINc}(L_2), \text{SLINc}(L_1) + \text{SREGc}(L_2)\}.$

However, if we consider fixed alphabets, then, at this point, we are only able to show that the bound of Theorem 6.3.3 is tight for strict regular grammars. Even though we are able to prove a lower bound on the exact and cover complexity of concatenation for strict linear grammars in the following theorem, this lower bound does not match the upper bound obtained in Theorem 6.3.3.

Theorem 6.3.8. *Let $X \in \Gamma_s$. Then there exists a finite alphabet Σ such that for all integers $n_1, n_2 \geq 1$, there exist finite languages $L_1, L_2 \in \mathcal{P}_{\text{fin}}(\Sigma^*)$ with $X_{\text{cc}}(L_1) = n_1$ and $X_{\text{cc}}(L_2) = n_2$ such that*

$$X_{\text{cc}}(L_1 L_2) \geq X_{\text{cc}}(L_1) + X_{\text{cc}}(L_2).$$

PROOF. Let $\Sigma = \{a\}$ and, for all integers $n_1, n_2 \geq 1$, we define the finite languages

$$L = \{a^{n_1+n_2}\} \quad \text{and} \quad L' = \{a^{2n_1+2n_2-2}\}.$$

Moreover, let

$$L_1 = \{a^{n_1}\} \quad \text{and} \quad L_2 = \{a^{n_2}\}.$$

as well as

$$L'_1 = \{a^{2n_1-1}\} \quad \text{and} \quad L'_2 = \{a^{2n_2-1}\}.$$

Clearly,

$$L = L_1 L_2 \quad \text{and} \quad L' = L'_1 L'_2.$$

From Lemma 4.1.12, we thus get that

$$\text{SREGcc}(L_1) \geq n_1, \quad \text{SREGcc}(L_2) \geq n_2, \quad \text{SLINcc}(L'_1) \geq n_1, \quad \text{and} \quad \text{SLINcc}(L'_2) \geq n_2.$$

It is easy to see that also (for the strict linear cases, see the grammars defined in Theorem 6.2.7)

$$\text{SREGcc}(L_1) \leq n_1, \quad \text{SREGcc}(L_2) \leq n_2, \quad \text{SLINcc}(L'_1) \leq n_1, \quad \text{and} \quad \text{SLINcc}(L'_2) \leq n_2.$$

Again, by Lemma 4.1.12, it follows that

$$\text{SREGcc}(L) = \text{SREGcc}(L_1 L_2) \geq n_1 + n_2 = \text{SREGcc}(L_1) + \text{SREGcc}(L_2)$$

and

$$\text{SLINcc}(L') = \text{SLINcc}(L'_1 L'_2) \geq n_1 + n_2 = \text{SLINcc}(L'_1) + \text{SLINcc}(L'_2).$$

This proves the stated claim. □

The arguments used in the proof of Theorem 6.3.8 in conjunction with Corollary 4.1.13 show that an analogous result also holds for the exact complexity.

Corollary 6.3.9. *Let $X \in \Gamma_s$. Then there exists a finite alphabet Σ such that for all integers $n_1, n_2 \geq 1$, there exist finite languages $L_1, L_2 \in \mathcal{P}_{\text{fin}}(\Sigma^*)$ with $Xc(L_1) = n_1$ and $Xc(L_2) = n_2$ such that*

$$Xc(L_1 L_2) \geq Xc(L_1) + Xc(L_2).$$

We will now prove the following lower bound on the exact complexity of the concatenation of finite languages that applies to all grammar types in Δ .

Theorem 6.3.10. *Let $X \in \Delta$. Then there is an alphabet Σ such that for all integers $n_1, n_2 \geq 1$, there exist finite languages $L_1, L_2 \in \mathcal{P}_{\text{fin}}(\Sigma^*)$ with $Xc(L_1) = n_1$ and $Xc(L_2) = n_2$ such that*

$$Xc(L_1 L_2) \geq \max\{Xc(L_1), Xc(L_2)\}.$$

As a first step, we show the following lower bound on the exact complexity of concatenating two finite languages with a fresh middle marker in between them.

Lemma 6.3.11. *Let $X \in \Delta$ and $L_1, L_2 \in \mathcal{P}_{\text{fin}}(\Sigma^*)$ be finite languages. Then*

$$Xc(L_1 \# L_2) \geq \max\{Xc(L_1), Xc(L_2)\},$$

where $\#$ does not occur in Σ .

PROOF. Let L_1 and L_2 be finite languages over Σ and $G = (N, \Sigma \cup \{\#\}, P, S)$ be an X -grammar with

$$L(G) = L_1 \# L_2 \quad \text{and} \quad |G| = Xc(L_1 \# L_2).$$

We will first show that the right-hand side of every production in P contains at most one nonterminal that derives a string containing the letter $\#$. Grammars of some type in Γ_s have this property by definition. So, for grammars of some type in Γ , assume to the contrary that there is a production

$$A \rightarrow \alpha_1 B \alpha_2 C \alpha_3 \quad \text{with} \quad B \Rightarrow_G^* \beta_{1,1} \# \beta_{1,2} \quad \text{and} \quad C \Rightarrow_G^* \beta_{2,1} \# \beta_{2,2},$$

where, for $i \in \{1, 2\}$, $\alpha_1, \alpha_2, \alpha_3, \beta_{i,1}, \beta_{i,2} \in (N \cup \Sigma)^*$. Then, since G is minimal, by Lemma 4.2.1, every nonterminal is used to derive some word in $L(G)$. Thus, there is some derivation

$$S \Rightarrow_G^* \gamma_1 A \gamma_2 \Rightarrow_G \gamma_1 \alpha_1 B \alpha_2 C \alpha_3 \gamma_2 \Rightarrow_G^* \gamma_1 \alpha_1 \beta_{1,1} \# \beta_{1,2} \alpha_2 \beta_{2,1} \# \beta_{2,2} \alpha_3 \gamma_2 \Rightarrow_G^* w_1 \# w_2 \# w_3,$$

for $\gamma_1, \gamma_2 \in (N \cup \Sigma)^*$ and $w_1, w_2, w_3 \in \Sigma^*$. However, this implies that the language $L(G) = L_1 \# L_2$ contains a word with two occurrences of $\#$, which is impossible. Thus, the right-hand side of each production in P contains at most one nonterminal that derives a string containing the letter $\#$.

Next, we will show that there is an X -grammar $G' = (N, \Sigma, P', S)$ satisfying

$$L(G') = (L_1 \#)^{-1} L(G) = L_2 \quad \text{and} \quad |G'| \leq |G|.$$

First, we define the set of nonterminals that derive #:

$$N_{\#} = \{A \in N \mid A \Rightarrow_G^* \alpha_1 \# \alpha_2 \text{ and } \alpha_1, \alpha_2 \in (N \cup \Sigma)^*\}.$$

Now, we are in the position to define the production set of G' :

$$P' = \{A \rightarrow B\alpha_2 \mid A \rightarrow \alpha_1 B\alpha_2 \in P \text{ and } B \in N_{\#}\} \cup \{A \rightarrow \alpha_2 \mid A \rightarrow \alpha_1 \# \alpha_2 \in P\} \\ \cup \{A \rightarrow \alpha \in P \mid \alpha \not\Rightarrow_G^* \alpha_1 \# \alpha_2 \text{ and } \alpha_1, \alpha_2 \in (N \cup \Sigma)^*\}.$$

Clearly, $|G'| \leq |G|$. It remains to show that

$$L(G') = (L_{\#})^{-1}L(G).$$

To this end, we will show the following equivalence by induction on the length of a derivation in G :

$$\text{for all } A \in N_{\#}: \quad A \Rightarrow_G^* \alpha_1 \# \alpha_2 \quad \text{iff} \quad A \Rightarrow_{G'}^* \alpha_2, \quad (6.1)$$

where $\alpha_1, \alpha_2 \in (N \cup \Sigma)^*$. Let $A \in N_{\#}$ be arbitrary; then clearly $A \Rightarrow_G^* \alpha_1 \# \alpha_2$.

- **Base case:** If

$$A \Rightarrow_G \alpha_1 \# \alpha_2,$$

i.e.,

$$A \rightarrow \alpha_1 \# \alpha_2 \in P,$$

then by definition of P' , we have that

$$A \rightarrow \alpha_2 \in P'.$$

As a consequence,

$$A \Rightarrow_{G'} \alpha_2.$$

This concludes the base case.

- **Induction step:** Assume that

$$A \Rightarrow_G^{k+1} \alpha_1 \# \alpha_2,$$

for $k \geq 1$. Then, we clearly have that

$$A \Rightarrow_G^{k+1} \alpha_1 \# \alpha_2 \quad \text{iff} \quad A \Rightarrow_G \beta_1 B \beta_2 \Rightarrow_G^k \beta_1 \gamma_1 \# \gamma_2 \beta_2,$$

where $\alpha_1 = \beta_1 \gamma_1$, $\alpha_2 = \gamma_2 \beta_2$, and $\beta_i, \gamma_i \in (N \cup \Sigma)^*$, for $i \in \{1, 2\}$. This means, in particular, that

$$B \Rightarrow_G^k \gamma_1 \# \gamma_2,$$

and thus, by induction hypothesis, we get that

$$B \Rightarrow_G^k \gamma_1 \# \gamma_2 \quad \text{iff} \quad B \Rightarrow_{G'}^* \gamma_2.$$

Furthermore, since

$$A \rightarrow \beta_1 B \beta_2 \in P$$

and by minimality of G , it follows that

$$B \Rightarrow_G^* \gamma_1 \# \gamma_2 \Rightarrow_G^* w_1 \# w_2,$$

for $w_1, w_2 \in \Sigma^*$, that is, $B \in N_\#$, we have that

$$A \rightarrow B \beta_2 \in P'.$$

Putting things together, we get

$$A \Rightarrow_G^* \alpha_1 \# \alpha_2 \quad \text{iff} \quad A \Rightarrow_{G'} B \beta_2 \Rightarrow_{G'}^* \gamma_2 \beta_2 = \alpha_2.$$

Note that for all $A \in N \setminus N_\#$, it clearly holds that

$$A \Rightarrow_G^* \alpha \quad \text{iff} \quad A \Rightarrow_{G'}^* \alpha,$$

for $\alpha \in (N \cup \Sigma^*)$. Since $S \in N_\#$, for all $w_1 \in L_1$ and all $w_2 \in L_2$, it follows from Equation 6.1 that

$$S \Rightarrow_G^* w_1 \# w_2 \quad \text{iff} \quad S \Rightarrow_{G'}^* w_2,$$

i.e.,

$$L(G') = (L_1 \#)^{-1} L(G) = L_2.$$

Similarly, one can show that there is an X -grammar G'' with

$$L(G'') = L(G)(\#L_2)^{-1} \quad \text{and} \quad |G''| \leq |G|.$$

By definition of X_c , it follows that

$$X_c(L_1) \leq |G''| \leq |G| \quad \text{and} \quad X_c(L_2) \leq |G'| \leq |G|.$$

Putting things together, we get that

$$X_c(L_1 \# L_2) = |G| \geq \max\{X_c(L_1), X_c(L_2)\}.$$

This proves the stated claim. □

Now, we will take a closer look at how the exact complexity of a finite language L changes if we append (or prepend) a fresh symbol to all words in L .

Lemma 6.3.12. *Let $X \in \Gamma$ and $L \in \mathcal{P}_{\text{fin}}(\Sigma^*)$ be a finite language. Then*

$$X_c(L \cdot \#) = X_c(L),$$

where $\#$ is a letter that is not contained in Σ . The statement remains valid if one considers the language $\# \cdot L$ instead of $L \cdot \#$.

PROOF. Let $G = (N, \Sigma, P, S)$ be a minimal X -grammar, for $X \in \{\text{LIN}, \text{CF}\}$, with $L(G) = L$, i.e.,

$$|G| = X_c(L).$$

We define the X -grammar $G' = (N, \Sigma \cup \{\#\}, S)$ with

$$L(G') = L(G) \cdot \# = L \cdot \#$$

using the following set of productions:

$$P' = \{S \rightarrow \alpha\# \mid S \rightarrow \alpha \in P\} \cup \{A \rightarrow \alpha \in P \mid A \neq S\}.$$

For $X = \text{REG}$, we need a slightly different construction for P' :

$$P' = \{A \rightarrow w\# \mid A \rightarrow w \in P \text{ and } w \in \Sigma^*\} \cup \{A \rightarrow \alpha \in P \mid \alpha \notin \Sigma^*\}.$$

Clearly, both of these constructions yield

$$L(G') = L(G) \cdot \# \quad \text{and} \quad |G'| \leq |G|.$$

Thus,

$$X_c(L \cdot \#) \leq X_c(L).$$

But it also holds that

$$X_c(L) \leq X_c(L \cdot \#),$$

for otherwise we could erase the letter $\#$ from all productions of a minimal grammar for $L \cdot \#$ and obtain a grammar that generates L and has fewer productions than $X_c(L)$. When considering $\# \cdot L$ instead of $L \cdot \#$, it suffices to define the production set as follows:

$$P' = \{S \rightarrow \#\alpha \mid S \rightarrow \alpha \in P\} \cup \{A \rightarrow \alpha \in P \mid A \neq S\}.$$

This proves the stated claim. □

By the very nature of the restrictions in strict regular and strict linear grammars, it is not really surprising that we need an additional production in order to be able to append (or prepend) a fresh new symbol to all words in a language.

Lemma 6.3.13. *Let $X \in \Gamma_s$ and $L \in \mathcal{P}_{\text{fin}}(\Sigma^*)$ be a finite language. Then*

$$X_c(L \cdot \#) \leq X_c(L) + 1 \leq X_c(L \cdot \#) + 1,$$

where $\#$ is a letter that is not contained in Σ . The statement remains valid if one considers the language $\# \cdot L$ instead of $L \cdot \#$.

PROOF. For $X \in \Gamma_s$ and $L \in \mathcal{P}_{\text{fin}}(\Sigma^*)$, let $G = (N, \Sigma, P, S)$ be a minimal X -grammar generating L , i.e.,

$$|G| = X_c(L).$$

We define another X -grammar $G' = (N \cup \{C_\#\}, \Sigma \cup \{\#\}, S')$ with

$$L(G') = L(G) \cdot \# = L \cdot \#,$$

where $C_\# \notin N$.

For $X = \text{SREG}$, let $S' := S$ and define the set of productions P' as follows:

$$P' = \{A \rightarrow aC_\# \mid A \rightarrow a \in P \text{ and } a \in \Sigma \cup \{\varepsilon\}\} \cup \{A \rightarrow \alpha \in P \mid \alpha \notin \Sigma \cup \{\varepsilon\}\} \cup \{C_\# \rightarrow \#\}.$$

For $X = \text{SLIN}$, let $S' := S_\#$ such that $S_\# \notin N$, and we construct P' as follows:

$$P' = P \cup \{S_\# \rightarrow S\}.$$

In both of these cases, we clearly have that

$$L(G') = L(G) \cdot \# = L \cdot \# \quad \text{and} \quad |G'| \leq |G| + 1.$$

But it also holds that

$$Xc(L) \leq Xc(L \cdot \#),$$

for otherwise we could erase the letter $\#$ from all productions of a minimal grammar for $L \cdot \#$ and obtain a grammar that generates L and has fewer productions than $Xc(L)$. Putting things together, we get

$$Xc(L \cdot \#) \leq Xc(L) + 1 \leq Xc(L \cdot \#) + 1.$$

To generate $\# \cdot L$ with an X -grammar, we set $S' := S_\#$ and define

$$P' = P \cup \{S_\# \rightarrow \#S\}$$

as the set of productions. This finishes the proof of the stated claim. \square

As already mentioned, thus far—except for strict regular grammars—we have not been able to show that the upper bound on the exact complexity of concatenation is tight, but with the help of the proof of Corollary 6.3.9 as well as the results of Lemmas 6.3.11 and 6.3.12, we are finally able to prove the lower bound expressed in Theorem 6.3.10.

PROOF (OF THEOREM 6.3.10). Let $X \in \Gamma$, and consider, for all integers $n_1, n_2 \geq 1$, the languages

$$K_1 = \{a^i b^i c^i \mid 1 \leq i \leq n_1\} \cdot \{\#\} \quad \text{and} \quad K_2 = \{a^i b^i c^i \mid 1 \leq i \leq n_2\}.$$

From Equation 4.2, it follows that

$$Xc(K_1 \#^{-1}) = n_1 \quad \text{and} \quad Xc(K_2) = n_2.$$

Moreover, from Lemma 6.3.12, we get that

$$Xc(K_1) = Xc(K_1 \#^{-1}) = n_1. \tag{6.2}$$

Thus, we finally get that

$$Xc(K_1 K_2) \geq \max\{Xc(K_1 \#^{-1}), Xc(K_2)\} = \max\{Xc(K_1), Xc(K_2)\}$$

from Lemma 6.3.11 and Equation 6.2.

For $X \in \Gamma_s$, the claim follows from the proof of Corollary 6.3.9. \square

The exact complexity bounds on the operations union and concatenation will play an important role in the proof of the main result of the next chapter.

Complexity of The Smallest Grammar Problem for Finite Languages



UNTIL now, we have focused our attention solely on questions regarding the grammatical complexity of finite languages that fall under the umbrella of descriptive complexity. However, in this chapter, we will shift the focus towards problems that belong to the realm of computational complexity theory.¹ Nevertheless, many of the previously established results will come in handy in order to show some of the results in this chapter. One of the main results is that, for fixed alphabets of cardinality at least 5, given an arbitrary context-free grammar G with p productions that generates a finite language L , the minimal number of productions necessary to generate L with a context-free grammar cannot be approximated within a factor of

$$o(p^{1/6}),$$

unless $P = NP$. Since the size of a context-free grammar depends on the number of productions, the above result also implies that, given an arbitrary context-free grammar G with size s that generates a finite language L , the minimal size of a context-free grammar that generates L cannot be approximated within a factor of

$$o(s^{1/7}),$$

unless $P = NP$. This second result is related to the inapproximability of the smallest grammar problem within a small constant factor, unless $P = NP$ [CLL⁺05]. This classic

¹For a recent survey on modern aspects of computational complexity within formal language theory, see [Fer19].

version of the smallest grammar problem asks for the smallest (in terms of grammar size) context-free grammar that generates exactly a *single* given word. In order to obtain the main result of this chapter, we will reduce from the coNP-complete propositional tautology problem. The reduction features a gadget based on the language

$$T_n = \{w\$w\#w \mid w \in \{a, b\}^n\},$$

where $n \geq 1$ is an integer. For the correctness proof of the reduction, we will use estimates on the grammatical complexity of certain finite languages as well as on the operations union and concatenation of finite languages, which have been shown in Chapters 4 and 6. Furthermore, inspired by the investigation of problems regarding finite automata under the assumption of the so-called *Exponential Time Hypothesis* (ETH) in the paper [FK17], we will also consider the uniform-length universality problem under the assumption of the ETH. Intuitively, the Exponential Time Hypothesis is a conjecture which states that k -SAT, for $k \geq 3$, cannot be decided in subexponential time (see, e.g., [CFK⁺15]) and was introduced in [IP99]. The aforementioned uniform-length universality problem asks, for a given context-free grammar $G = (N, \Sigma, P, S)$ and an integer $\ell \geq 0$, whether $L(G)$ generates all words of length ℓ over Σ , i.e., whether $L(G) = \Sigma^\ell$. In particular, assuming the ETH, we will show that there is no algorithm that decides uniform-length universality in time

$$\mathcal{O}^*\left(2^{o(p^{1/4})}\right),$$

where p is the number of productions of the given grammar G . In addition, we will also show that assuming the ETH, there is no algorithm that decides the uniform-length universality problem in time

$$\mathcal{O}^*\left(2^{o(s^{1/4})}\right),$$

where s is the size of the given grammar G .

Some of the results in this chapter have been published in [GHW18].

7.1 Inapproximability of the Minimal Number of Productions

In this section, we will show that the minimal number of context-free productions necessary to generate the finite language that is generated by a given context-free grammar with p productions cannot be approximated within a factor of

$$o(p^{1/6}),$$

unless $P = NP$. The uniform language

$$T_n = \{w\$w\#w \mid w \in \{a, b\}^n\},$$

as introduced in Section 4.2, will be a basic building block for this endeavour. Our proof strategy is based on a gap-reduction from the coNP-complete *tautology* problem for 3-DNF formulae:

Given a propositional formula φ in 3-DNF, it is coNP-complete to determine whether φ is a tautology—in other words, whether the negation $\neg\varphi$ of φ is *unsatisfiable*.

Then the core idea is to give a suitable representation of the satisfying assignments of φ in the language $\{0, 1\}^n$ for the n variables in the form of a grammar G_φ such that

$$\varphi \text{ is a tautology} \quad \text{if and only if} \quad L(G_\varphi) = \{0, 1\}^n.$$

By construction, there is a one-to-one correspondence between assignments and words from the set $\{0, 1\}^n$. In order to finish our reduction, we embed G_φ into a grammar that generates the finite and uniform language

$$L_\varphi = L(G_\varphi) \cdot \{\&\} \cdot \{a, b, \$, \#\}^{3c \cdot \lceil \log n \rceil + 2} \cup \{0, 1\}^n \cdot \{\&\} \cdot T_{c \cdot \lceil \log n \rceil},$$

for some carefully chosen constant c . It is not hard to see that this reduction is polynomial in n even if we force the grammar that generates L_φ to be strict regular. Then, we distinguish two cases:

1. On the one hand, if φ is a tautology, then we have that

$$L_\varphi = \{0, 1\}^n \cdot \{\&\} \cdot \{a, b, \$, \#\}^{3c \cdot \lceil \log n \rceil + 2},$$

i.e., there is a context-free grammar with a constant number of productions that generates L_φ . Moreover, for the X-grammars with $X \in \{\text{REG}, \text{LIN}\} \cup \Gamma_s$, a linear number of productions suffices, i.e., the number is in $\mathcal{O}(n)$.

2. On the other hand, if φ is not a tautology, then there is an assignment under which φ evaluates to *false*. Hence, there is a word $w \in \{0, 1\}^n$ which corresponds to that assignment and which is *not* a member of $L(G_\varphi)$. But then the left quotient of L_φ w.r.t. the word $w\&$, that is, the language

$$(w\&)^{-1} L_\varphi = \{v \in \{a, b, \$, \#\}^* \mid w\&v \in L_\varphi\},$$

is equal to the language of triples $T_{c \cdot \lceil \log n \rceil}$. From this quotient construction, it will then follow that

$$\text{Xc}(T_{c \cdot \lceil \log n \rceil}) = \mathcal{O}(\text{Xc}(L_\varphi) \cdot n^4).$$

Since we have already seen that

$$\text{Xc}(T_{c \cdot \lceil \log n \rceil}) = \Omega(n^c),$$

we can thus deduce that

$$\text{Xc}(L_\varphi) = \Omega(n^{c-4}).$$

This allows us to prove the main result of this section:

Theorem 7.1.1. *Let $X \in \Delta$. Given an X -grammar with p productions that generates a finite language L , it is impossible to approximate $Xc(L)$ within a factor of $o(p^{1/6})$, unless $P = NP$.*

The remainder of this section is devoted to the proof of Theorem 7.1.1.

Now, we will construct a strict regular grammar that generates the satisfying truth assignments of the given propositional formula φ in 3-DNF with $|\text{var}(\varphi)| = n$. In this way, we will reduce the propositional tautology to the *uniform-length universality problem*:

UNIFORM-LENGTH UNIVERSALITY

INSTANCE: An X -grammar $G = (N, \Sigma, P, S)$, for $X \in \Delta$, and an integer $\ell \geq 0$.

QUESTION: Does it hold that $L(G) = \Sigma^\ell$?

Let φ be a propositional formula in 3-DNF with $\text{var}(\varphi) = \{x_1, x_2, \dots, x_n\}$ consisting of the conjunctive clauses C_1, C_2, \dots, C_m , for $m \geq 1$. We define a strict right-linear grammar $G_\varphi = (N, \{0, 1\}, P, S)$ as follows:

the set of nonterminals is defined as

$$N = \{S\} \cup \{A_{i,j} \mid 1 \leq i \leq m \text{ and } 2 \leq j \leq n\}$$

and the set P consists of the following productions

- $A_{i,j} \rightarrow 1A_{i,j+1}$ if $x_j \in C_i$ and $\neg x_j \notin C_i$, for $1 \leq i \leq m$ and $1 \leq j \leq n-1$,
- $A_{i,j} \rightarrow 0A_{i,j+1}$ if $x_j \notin C_i$ and $\neg x_j \in C_i$, for $1 \leq i \leq m$ and $1 \leq j \leq n-1$,
- $A_{i,j} \rightarrow 1A_{i,j+1}$ if $x_j \notin C_i$ and $\neg x_j \notin C_i$, for $1 \leq i \leq m$ and $1 \leq j \leq n-1$,
- $A_{i,j} \rightarrow 0A_{i,j+1}$ if $x_j \in C_i$ and $\neg x_j \in C_i$, for $1 \leq i \leq m$ and $1 \leq j \leq n-1$,
- $A_{i,n} \rightarrow 1$ if $x_n \in C_i$ and $\neg x_n \notin C_i$, for $1 \leq i \leq m$,
- $A_{i,n} \rightarrow 0$ if $x_n \notin C_i$ and $\neg x_n \in C_i$, for $1 \leq i \leq m$,
- $A_{i,n} \rightarrow 1$ if $x_n \notin C_i$ and $\neg x_n \notin C_i$, for $1 \leq i \leq m$,
- $A_{i,n} \rightarrow 0$ if $x_n \in C_i$ and $\neg x_n \in C_i$, for $1 \leq i \leq m$,

where, for all $i \in \{1, 2, \dots, m\}$, we set $A_{i,1} := S$. Note that the above reduction from 3-DNF formulae to context-free grammars is essentially the same as the reduction to regular expressions presented in [Hun73].

Now, we illustrate—by means of a concrete example—how the reduction from the propositional tautology to the uniform-length universality problem works:

Example 7.1.2. Consider the propositional formula

$$\varphi = \underbrace{(x_1 \wedge x_3)}_{C_1} \vee \underbrace{(x_1 \wedge \neg x_3)}_{C_2} \vee \underbrace{(\neg x_1 \wedge x_2)}_{C_3} \vee \underbrace{(\neg x_1 \wedge \neg x_2)}_{C_4}.$$

Clearly, φ is a tautology in 3-DNF with $\text{var}(\varphi) = \{x_1, x_2, x_3\}$. By the above reduction, the strict regular grammar G_φ consists of the following productions:

$$\begin{array}{ll} A_{1,1} \rightarrow 1A_{1,2} & A_{2,1} \rightarrow 1A_{2,2} \\ A_{1,2} \rightarrow 0A_{1,3} \mid 1A_{1,3} & A_{2,2} \rightarrow 0A_{2,3} \mid 1A_{2,3} \\ A_{1,3} \rightarrow 1 & A_{2,3} \rightarrow 0 \\ \\ A_{3,1} \rightarrow 0A_{3,2} & A_{4,1} \rightarrow 0A_{4,2} \\ A_{3,2} \rightarrow 1A_{3,3} & A_{4,2} \rightarrow 0A_{4,3} \\ A_{3,3} \rightarrow 0 \mid 1 & A_{4,3} \rightarrow 0 \mid 1. \end{array}$$

Note that, for $i \in \{1, 2, 3, 4\}$, the block with start symbol $A_{i,1}$ derives the satisfying truth assignments of the conjunctive clause C_i . Consequently,

$$L(G_\varphi) = \{0, 1\}^3,$$

since each start symbol $A_{i,1}$ derives two distinct satisfying truth assignments.

Now, we slightly modify clause C_4 of φ and obtain another propositional formula φ' of the following form:

$$\varphi' = \underbrace{(x_1 \wedge x_3)}_{C_1} \vee \underbrace{(x_1 \wedge \neg x_3)}_{C_2} \vee \underbrace{(\neg x_1 \wedge x_2)}_{C_3} \vee \underbrace{(\neg x_1 \wedge \neg x_2 \wedge \neg x_3)}_{C'_4}.$$

Then, it is easy to see that φ' is *not* a tautology, since

$$\sigma(x_1, x_2, x_3) = 001 \not\models \varphi'.$$

Moreover, by the above reduction, the corresponding strict regular grammar $G_{\varphi'}$ consists of the following productions:

$$\begin{array}{ll} A_{1,1} \rightarrow 1A_{1,2} & A_{2,1} \rightarrow 1A_{2,2} \\ A_{1,2} \rightarrow 0A_{1,3} \mid 1A_{1,3} & A_{2,2} \rightarrow 0A_{2,3} \mid 1A_{2,3} \\ A_{1,3} \rightarrow 1 & A_{2,3} \rightarrow 0 \\ \\ A_{3,1} \rightarrow 0A_{3,2} & A_{4,1} \rightarrow 0A_{4,2} \\ A_{3,2} \rightarrow 1A_{3,3} & A_{4,2} \rightarrow 0A_{4,3} \\ A_{3,3} \rightarrow 0 \mid 1 & A_{4,3} \rightarrow 0. \end{array}$$

A close inspection of the grammar $G_{\varphi'}$ reveals that

$$L(G_{\varphi'}) \neq \{0, 1\}^3,$$

since the word 001 is not $G_{\varphi'}$ -derivable, i.e., $001 \notin L(G_{\varphi'})$.

Note that since the grammar G_{φ} can be constructed in polynomial time, the above reduction is polynomial-time computable. The following proposition expresses that the number of productions as well as the size of the above constructed grammar G_{φ} is polynomial in the number of variables occurring in φ .

Proposition 7.1.3. *Let φ be a formula in 3-DNF with n variables and m conjunctive clauses and let G_{φ} be the grammar constructed above. Then*

$$|G_{\varphi}| \leq 32 \cdot n^4 \quad \text{and} \quad |G_{\varphi}|_s \leq 128 \cdot n^4.$$

PROOF. Since the propositional formula φ consists of n variables, φ can—under the assumption that no literal occurs more than once in a single conjunctive clause—contain at most

$$m \leq 16 \cdot n^3$$

different conjunctive clauses. This follows because n distinct variables give rise to $2 \cdot n$ distinct literals, and from these $2 \cdot n$ literals, one can obtain at most

$$\sum_{i=0}^3 (2 \cdot n)^i = \frac{(2 \cdot n)^4 - 1}{(2 \cdot n) - 1} \leq \frac{2 \cdot ((2 \cdot n)^4 - 1)}{2 \cdot n} \leq 2 \cdot (2 \cdot n)^3 = 16 \cdot n^3$$

different conjunctive clauses consisting of at most 3 literals. By counting the productions of the strict regular grammar G_{φ} , we get that we have at most two productions for each variable in a single clause. Thus,

$$|G_{\varphi}| \leq 2 \cdot (n \cdot m) \leq 2 \cdot (n \cdot 16 \cdot n^3) = 32 \cdot n^4.$$

Finally,

$$|G_{\varphi}|_s \leq 128 \cdot n^4$$

follows from the fact that each production in G_{φ} consists of at most four symbols, i.e.,

$$|G_{\varphi}|_s \leq 4 \cdot |G_{\varphi}| \leq 4 \cdot 32 \cdot n^4 = 128 \cdot n^4.$$

This concludes the proof of the proposition. \square

The construction of G_{φ} also satisfies the property that a word $w \in \{0, 1\}^n$ is derivable in G_{φ} if and only if w —interpreted as a truth assignment—satisfies the formula φ , i.e.,

$$L(G_{\varphi}) = \{w \in \{0, 1\}^n \mid w \models \varphi\}.$$

An immediate consequence of the subsequent proposition is that G_{φ} generates all words of length n over $\{0, 1\}$ if and only if φ is a tautology. In other words, the reduction from the tautology to the uniform-length universality problem is *correct*.

Proposition 7.1.4. *Let φ be a propositional formula in 3-DNF with n variables. Then, for all words $w \in \{0, 1\}^n$, it holds that $w \in L(G_\varphi)$ if and only if $w \models \varphi$.*

PROOF. Assume, without loss of generality, that φ contains at least one non-empty clause, for otherwise, on the one hand, φ would trivially be unsatisfiable and $L(G_\varphi) = \emptyset$ if φ contains no clause; on the other hand, if φ contains at least one clause, but all of these clauses are empty, then φ is valid and we have that $L(G_\varphi) = \{0, 1\}^n$.

For the left-to-right direction, assume that there is some word $w \in L(G_\varphi)$, i.e., there is some derivation

$$S \Rightarrow_{G_\varphi}^* w.$$

Since, by definition, G_φ is an SREG-grammar without unit productions and $S = A_{i,1}$, for some $i \in \{1, 2, \dots, m\}$, the derivation of the word

$$w = a_{i,1} a_{i,2} \dots a_{i,n}$$

in G_φ must have the following form:

$$A_{i,1} \Rightarrow_{G_\varphi} a_{i,1} A_{i,2} \Rightarrow_{G_\varphi} \dots \Rightarrow_{G_\varphi}^j a_{i,1} a_{i,2} \dots a_{i,j} A_{i,j+1} \Rightarrow_{G_\varphi}^{n-j} a_{i,1} a_{i,2} \dots a_{i,n},$$

where $a_{i,j} \in \{0, 1\}$, for all $j \in \{1, 2, \dots, n\}$. By definition of G_φ , the productions used in the above derivation are constructed based on the structure of the clause C_i in φ . Thus, assume, without loss of generality, that

$$C_i = \ell_{i,1} \wedge \ell_{i,2} \wedge \ell_{i,3},$$

where $\ell_{i,k}$ is a literal, for $k \in \{1, 2, 3\}$. By definition of G_φ , the following cases for the value of $a_{i,j}$ can be distinguished:

1. $a_{i,j} = 1$ if there is some $k \in \{1, 2, 3\}$ such that $\ell_{i,k} = x_j$,
2. $a_{i,j} = 0$ if there is some $k \in \{1, 2, 3\}$ such that $\ell_{i,k} = \neg x_j$, or
3. $a_{i,j} \in \{0, 1\}$ if $x_j \notin \text{var}(C_i)$.

Since C_i contains at least one literal, only the variables that actually occur in C_i are relevant for determining the truth of C_i under a given assignment. Now, we define a truth assignment σ for φ as follows:

$$\sigma(x_j) = a_{i,j}, \quad \text{for all } x_j \in \text{var}(\varphi).$$

The assignment σ clearly satisfies the following properties:

$$\sigma(x_j) = 1 \quad \text{if } \ell_{i,k} = x_j, \text{ for some } k \in \{1, 2, 3\}$$

and

$$\sigma(x_j) = 0 \quad \text{if } \ell_{i,k} = \neg x_j, \text{ for some } k \in \{1, 2, 3\}.$$

Consequently, we have that

$$\sigma(x_1, x_2, \dots, x_n) = \sigma(x_1)\sigma(x_2) \dots \sigma(x_n) = w$$

and

$$\sigma(\ell_{i,k}) = 1,$$

for all $k \in \{1, 2, 3\}$. That is, by semantics of \wedge ,

$$w \models C_i.$$

Finally, by semantics of \vee , it follows that

$$w \models \varphi.$$

This finishes the proof of the left-to-right direction.

For the right-to-left direction, assume that φ is satisfiable, i.e., there is some truth assignment σ with

$$\sigma(x_1, x_2, \dots, x_n) \models \varphi.$$

By semantics of \vee , there is some clause C_i , for $i \in \{1, 2, \dots, m\}$, such that

$$\sigma(x_1, x_2, \dots, x_n) \models C_i.$$

Assume, without loss of generality, that

$$C_i = \ell_{i,1} \wedge \ell_{i,2} \wedge \ell_{i,3},$$

where $\ell_{i,k}$ is a literal, for $k \in \{1, 2, 3\}$. Since σ satisfies C_i , by semantics of \wedge , the following must hold:

$$\sigma(\ell_{i,k}) = 1,$$

for all $k \in \{1, 2, 3\}$. That is, for $k \in \{1, 2, 3\}$, we have that

$$\sigma(x_j) = 1 \quad \text{if } \ell_{i,k} = x_j,$$

and

$$\sigma(x_j) = 0 \quad \text{if } \ell_{i,k} = \neg x_j.$$

Assume, without loss of generality, that

$$\text{var}(C_i) = \{x_{j_1}, x_{j_2}, x_{j_3}\},$$

for $1 \leq j_1, j_2, j_3 \leq n$. By definition of G_φ , there is a derivation

$$A_{i,1} \Rightarrow_{G_\varphi}^n a_{i,1} a_{i,2} \dots a_{i,n}$$

with $a_{i,j} \in \{0, 1\}$, for $j \in \{1, 2, \dots, m\}$ such that for all $s \in \{1, 2, 3\}$, it holds that

$$a_{i,j_s} = 1 = \sigma(x_{j_s}) \text{ if there is some } k \in \{1, 2, 3\} \text{ such that } \ell_{i,k} = x_{j_s}, \text{ or } x_j \notin \text{var}(C_i)$$

and

$$a_{i,j_s} = 0 = \sigma(x_{j_s}) \text{ if there is some } k \in \{1, 2, 3\} \text{ such that } \ell_{i,k} = \neg x_{j_s}, \text{ or } x_j \notin \text{var}(C_i).$$

Moreover, we have that

$$\sigma(x_j) = a_{i,j},$$

for all x_j with $j \in \{1, 2, \dots, n\} \setminus \{j_1, j_2, j_3\}$. Clearly, by definition of G_φ , the nonterminal $A_{i,1}$ derives all words $v \in \{0, 1\}^n$ of the form

$$v = b_1 b_2 \dots b_n,$$

where $b_{j_1} = a_{j_1}$, $b_{j_2} = a_{j_2}$, and $b_{j_3} = a_{j_3}$. In particular, the nonterminal $A_{i,1}$ derives

$$\sigma(x_1)\sigma(x_2)\dots\sigma(x_n) = \sigma(x_1, x_2, \dots, x_n),$$

that is,

$$A_{i,1} \Rightarrow_{G_\varphi}^n a_{i,1} a_{i,2} \dots a_{i,n} = \sigma(x_1)\sigma(x_2)\dots\sigma(x_n) = \sigma(x_1, x_2, \dots, x_n).$$

This finishes the proof of the proposition. \square

A direct consequence of the above polynomial-time reduction is that the uniform-length universality problem is coNP-hard for all grammar types under consideration.

Corollary 7.1.5. *The uniform-length universality problem is coNP-hard for all grammar types in Δ .*

The next step in our proof strategy is the construction of a grammar which generates the left quotient of a word with a finite uniform language different from $\{\varepsilon\}$ and which also has a polynomial number of productions. The word is given as the language of a DFA and the finite language is given as the language of a context-free grammar. For this endeavour, we need two lemmas as prerequisites. The first one states that any finite language whose longest word is of length at least 3, can be generated by a grammar in which the right-hand side of each production is not longer than the length of a longest word in the language.

Lemma 7.1.6. *Let $X \in \Delta$, G be an X -grammar generating a finite language, and $\ell = \max\{|w| \mid w \in L(G)\} \geq 3$. Then there is an X -grammar G' in which all right-hand sides of productions are of length at most ℓ such that $L(G') = L(G)$ and $|G'| \leq |G|$.*

PROOF. Let $X \in \Delta$ and $G = (N, \Sigma, P, S)$ be an X -grammar generating a finite language with

$$\ell := \max\{|w| \mid w \in L(G)\} \geq 3.$$

Note that for strict regular and strict linear grammars the claim holds trivially, since in these kinds of grammars, there is no production with a right-hand side that is longer

than 3. Thus, in the remainder of the proof, we assume that $X \in \Gamma$. Suppose that G contains a production

$$A \rightarrow \alpha \quad \text{with} \quad \alpha \in (N \cup \Sigma)^* \text{ and } |\alpha| > \ell.$$

In addition, we also assume that A is both reachable from S and useful, i.e., there are $\alpha_1, \alpha_2 \in (N \cup \Sigma)^*$ and a $v \in \Sigma^*$ such that

$$S \Rightarrow_G^* \alpha_1 A \alpha_2 \Rightarrow_G^* v.$$

This assumption can be made without loss of generality, since unreachable and useless productions can be removed without changing the generated language and without increasing the number of productions of the resulting grammar with respect to the original one. In particular, we then have that

$$\alpha = \beta_1 B \beta_2,$$

for some $\beta_1, \beta_2 \in (N \cup \Sigma)^*$ and some $B \in N$ with

$$L_B(G) = \{w \in \Sigma^* \mid B \Rightarrow_G^* w\} = \{\varepsilon\}.$$

For otherwise, we could derive a word of length longer than ℓ . The nonterminal B will then be removed from G and all occurrences of B will be replaced by ε . In this way, we will obtain a new grammar G' with $L(G') = L(G)$ and $|G'| \leq |G|$. This step is then repeated for all productions until there is no production left that has a right-hand side of length longer than ℓ . Note that this construction terminates, since, in each step, we have that the new set of nonterminals has fewer elements than the one of the previous grammar. \square

In order to simplify the proofs of some of the subsequent results, we need the notion of binary normal form for context-free grammars.

Definition 7.1.7 (Binary Normal Form). A context-free grammar $G = (N, \Sigma, P, S)$ is said to be in *binary normal form* (2NF) if the right-hand side of all productions in P has length at most two, i.e., for all $A \rightarrow \alpha \in P$ it holds that $|\alpha| \leq 2$.

The next lemma shows that assuming that a grammar is in 2NF does not constitute a major restriction. Note that, by definition, strict regular grammars are already in 2NF.

Lemma 7.1.8. *Let $X \in \Delta$, G be an X -grammar generating a finite language, and $\ell := \max\{|w| \mid w \in L(G)\} \geq 3$. Then there is an X -grammar G' in binary normal form such that $L(G') = L(G)$ and $|G'| \leq |G| \cdot \ell$.*

PROOF. If G is a strict regular grammar, then G is already in binary normal form and we are done. Therefore, let $G = (N, \Sigma, P, S)$ be an X -grammar, for $X \in \Gamma \cup \{\text{SLIN}\}$, generating a finite language and let

$$\ell := \max\{|w| \mid w \in L\} \geq 3.$$

Due to Lemma 7.1.6, we can, without loss of generality, assume that the right-hand side of each production in P has length at most ℓ .

If G is an X -grammar, for $X \in \{\text{REG}, \text{CF}\}$, then we define the following X -grammar $G' = (N \cup N', \Sigma, P', S)$, where P' contains all productions in P whose right-hand side has length at most 2. Additionally, P' contains for each production in P of the form

$$A \rightarrow X_1 X_2 \dots X_m$$

with $3 \leq m \leq \ell$ and $X_i \in N \cup \Sigma$, for $1 \leq i \leq m$, the following productions:

$$\begin{aligned} A &\rightarrow X_1 A_2 \\ A_2 &\rightarrow X_2 A_3 \\ &\vdots \\ A_{m-1} &\rightarrow X_{m-1} X_m. \end{aligned}$$

Note that the set N' is induced by the newly introduced nonterminals.

If G is a (strict) linear grammar, then we define the following strict linear grammar $G' = (N \cup N', \Sigma, P', S)$, where P' contains all productions in P whose right-hand side has length at most 2. Additionally, P' contains for each production in P of the form

$$A \rightarrow a_1 a_2 \dots a_k B a_{k+1} a_{k+2} \dots a_m$$

with $2 \leq m \leq \ell$, $B \in N$, and $a_i \in \Sigma$, for $1 \leq i \leq k$, the following productions:

$$\begin{aligned} A &\rightarrow a_1 A_2 \\ A_2 &\rightarrow a_2 A_3 \\ &\vdots \\ A_k &\rightarrow a_k A_{k+1} \\ A_{k+1} &\rightarrow A_{k+2} a_m \\ A_{k+2} &\rightarrow A_{k+3} a_{m-1} \\ &\vdots \\ A_{m-1} &\rightarrow A_m a_{k+2} \\ A_m &\rightarrow B a_{k+1}. \end{aligned}$$

For productions in P of the form

$$A \rightarrow a_1 a_2 \dots a_m$$

with $3 \leq m \leq \ell$ and $a_i \in \Sigma$, for $1 \leq i \leq \ell$, we proceed as in the CF-case. Note that we also introduce at most $m - 1$ additional nonterminals for each production in P . Thus, the

set N' is induced by these newly introduced nonterminals. Clearly, for each production in P , there are at most $m \leq \ell$ productions in P' and thus

$$|G'| \leq |G| \cdot \ell.$$

It remains to show that $L(G) = L(G')$. Let

$$w = w_1 w_2 \dots w_m$$

with $w_1, w_2, \dots, w_m \in \Sigma^*$, for an integer $m \geq 1$, be an arbitrary word. For all $A \in N$, we will show that

$$A \Rightarrow_G^* w \quad \text{iff} \quad A \Rightarrow_{G'}^* w$$

by induction on the length of a derivation.

- **Base case:** Assume that we have the one step derivation $A \Rightarrow_G w$. We distinguish two cases:

1. $|w| \leq 2$. Then $A \Rightarrow_G w$ iff $A \Rightarrow_{G'} w$ by definition of G' .
2. $|w| > 2$. Then $A \Rightarrow_G w$ iff

$$\begin{aligned} A \Rightarrow_{G'} w_1 A_2 \Rightarrow_{G'} w_1 w_2 A_3 \Rightarrow_{G'} \dots \\ \Rightarrow_{G'} w_1 w_2 \dots w_{m-2} A_{m-1} \Rightarrow_{G'} w_1 w_2 \dots w_{m-1} w_m \end{aligned}$$

by definition of G' .

This finishes the base case.

- **Induction step:** Assume that

$$A \Rightarrow_G^{n+1} w,$$

i.e.,

$$A \Rightarrow_G X_1 X_2 \dots X_k \Rightarrow_G^n w,$$

where $X_i \in N \cup \Sigma$, for $1 \leq i \leq k$. Clearly, we have that

$$X_i \Rightarrow_G^{\leq n} w_i,$$

for $w_i \in \Sigma^*$, $w = w_1 w_2, \dots, w_k$, and $1 \leq i \leq k$.

If $X_i \in \Sigma$, then

$$X_i = w_i \quad \text{and} \quad X_i \Rightarrow_{G'}^* w_i,$$

for $1 \leq i \leq k$.

If $X_i \in N$, then we can apply the induction hypothesis and get that

$$X_i \Rightarrow_G^* w_i \quad \text{iff} \quad X_i \Rightarrow_{G'}^* w_i.$$

Moreover, by definition of G' , we have that

$$\{A \rightarrow X_1 A_2, A_2 \rightarrow X_2 A_3, \dots, A_{k-1} \rightarrow X_{k-1} X_k\} \subseteq P'.$$

Hence,

$$A \Rightarrow_{G'}^* X_1 X_2 \dots X_k \Rightarrow_{G'}^* w_1 w_2 \dots w_k = w.$$

Note that a similar argument shows that the claim also holds for the SLIN- and LIN-case.

This concludes the proof of the lemma. \square

Now that we have collected all necessary ingredients, we can finally prove the result regarding the left quotient of a word with a finite uniform language different from $\{\varepsilon\}$. The proof uses Lemma 7.1.8 and a triple-like construction from [GS63] that is similar to the triple constructions that we have already seen in Chapter 5.

Theorem 7.1.9. *Let $X \in \Delta$ and G be an X -grammar generating a finite uniform language L with $L \neq \{\varepsilon\}$ whose words have length $\ell \geq 3$. Then, for every word $w \in \Sigma^*$, there is an X -grammar G' with $L(G') = w^{-1}L(G)$ and $|G'| = \mathcal{O}(|G| \cdot |w|^3 \cdot \ell)$.*

PROOF. We utilise the quotient construction from the proof of [GS63, Theorem 3.3] for our purpose. Let G be an X -grammar generating a finite uniform language $L \neq \{\varepsilon\}$ whose words have length $\ell \geq 3$. Then there is an equivalent minimal X -grammar G' , i.e.,

$$L(G') = L(G) = L \neq \{\varepsilon\} \quad \text{and} \quad |G'| = \text{Xc}(L) \leq |G|.$$

Since G' is a minimal X -grammar generating a finite uniform language with $L(G') \neq \{\varepsilon\}$, it follows from Proposition 4.1.8 that G' is ε -free, i.e., G' does not contain any ε -productions.

For the given context-free grammar $G' = (N', \Sigma, P', S')$ without ε -productions and a given non-returning deterministic finite automaton²

$$\mathcal{A} = (Q, \Sigma, q_0, \delta, \{f\})$$

with a single final state, a grammar G'' is constructed as follows:

the nonterminals are given by the set

$$N'' = \{[q, X, q'] \mid X \in \Sigma \cup N' \text{ and } q, q' \in Q\},$$

the start symbol is

$$[q_0, S', f],$$

and the productions are given by

1. $[q_0, a, q_0] \rightarrow a$, for each $a \in \Sigma$,
2. $[q, a, q'] \rightarrow \varepsilon$ if $a \in \Sigma$ and $\delta(q, a) = q'$,
3. $[q, A, q'] \rightarrow [q, X_1, q_1][q_1, X_2, q_2] \dots [q_{n-1}, X_n, q']$ if $A \rightarrow X_1 X_2 \dots X_n$ is a production in G' and $q, q_1, q_2, \dots, q_{n-1}, q'$ are states in Q .

²A finite automaton is *non-returning* if there are no transitions entering its start state.

In [GS63], it was shown that the grammar G'' constructed in this way accepts the language

$$L(G') \cdot (L(\mathcal{A}))^{-1} = L(G) \cdot (L(\mathcal{A}))^{-1}.$$

The construction yields the right quotient with a regular language and not the left quotient. We settle this by observing that the reversal operation L^R applied to a language L does not incur any increase in the number of productions necessary (see the proof of [DH12b, Theorem 1]), and that for all pairs of languages L_1 and L_2 it holds that

$$(L_1)^{-1} L_2 = (L_2^R (L_1^R)^{-1})^R.$$

The latter equality can be shown as follows:

$$\begin{aligned} (L_2^R (L_1^R)^{-1})^R &= \{w^R \in \Sigma^* \mid \text{there is a } v^R \in L_1^R \text{ such that } w^R v^R \in L_2^R\}^R \\ &= \{w^R \in \Sigma^* \mid \text{there is a } v \in L_1 \text{ such that } vw \in L_2\}^R \\ &= \{w \in \Sigma^* \mid \text{there is a } v \in L_1 \text{ such that } vw \in L_2\} \\ &= (L_1)^{-1} L_2. \end{aligned}$$

In the worst case, the bulk of the productions resulting from the grammar construction in [GS63] are of course those of the third type. It is essential that the grammar which we feed into the above construction is in 2NF if we want to bound the incurred blowup of the number of productions. Assuming that G' is given in 2NF, the productions of the third type are of the form

$$[q, A, q'] \rightarrow [q, X_1, q_1][q_1, X_2, q'].$$

Thus, the number of productions in G'' is in

$$\mathcal{O}(|Q|^3) \cdot |G'|.$$

Since, in our application, the language $L(\mathcal{A})$ consists of a single word w , we obtain that

$$|G''| = \mathcal{O}(|w|^3) \cdot |G'|.$$

In general, we cannot assume that G' is given in 2NF. By Lemma 7.1.8, the transformation into 2NF blows up the number of productions by another factor of

$$\ell = \max\{|w| \mid w \in L(G')\} \geq 3,$$

so the overall number of productions is in

$$\mathcal{O}(|w|^3 \cdot \ell) \cdot |G'|.$$

Since

$$L(G') = L(G) \quad \text{and} \quad |G'| \leq |G|,$$

it follows that

$$|G''| = \mathcal{O}(|G| \cdot |w|^3 \cdot \ell).$$

This finishes the proof of the theorem. □

In the remainder of this section, we prove our main result regarding the inapproximability of the minimal number of productions needed by a context-free grammar in order to generate a finite language. The following result expresses, for $X \in \Delta$, upper and lower bounds on the X -complexity of the finite language L_φ . In the case of context-free grammars, if φ is a tautology, then a constant number of productions suffices to generate L_φ , however, when we turn to (strict) regular and (strict) linear grammars, the upper bound jumps to a linear number (w.r.t. the number of propositional variables occurring in φ). On the other hand, if φ is not a tautology, then we obtain, for $X \in \Delta$, a lower bound on the X -complexity of L_φ which is polynomial in the number of variables occurring in φ . The value c in the next lemma refers to the constant c used in the construction of L_φ .

Lemma 7.1.10. *Let $X \in \Delta$ and let φ be a propositional formula in 3-DNF over n variables. Then*

$$Xc(L_\varphi) = \begin{cases} O(n) & \text{if } \varphi \text{ is a tautology,} \\ \Omega(n^{c-4}) & \text{otherwise.} \end{cases}$$

PROOF. Assume first that φ is a propositional tautology in 3-DNF over n variables. Then, by definition of the strict regular grammar G_φ and Proposition 7.1.4, we have that

$$L(G_\varphi) = \{0, 1\}^n.$$

This means that

$$\begin{aligned} L_\varphi &= (L(G_\varphi) \cdot \{\&\} \cdot \{a, b, \$, \#\}^{3c \cdot \lceil \log n \rceil + 2}) \cup (\{0, 1\}^n \cdot \{\&\} \cdot T_{c \cdot \lceil \log n \rceil}) \\ &= (\{0, 1\}^n \cdot \{\&\} \cdot \{a, b, \$, \#\}^{3c \cdot \lceil \log n \rceil + 2}) \cup (\{0, 1\}^n \cdot \{\&\} \cdot T_{c \cdot \lceil \log n \rceil}) \\ &= \{0, 1\}^n \cdot \{\&\} \cdot \{a, b, \$, \#\}^{3c \cdot \lceil \log n \rceil + 2}. \end{aligned}$$

From Equation 5.4, we know that

$$\text{LIN} \leq_c \text{REG} \leq_c \text{SREG}$$

and

$$\text{LIN} \leq_c \text{SLIN} \leq_c \text{SREG}.$$

Together with the results of Lemma 4.1.4 and Theorem 6.3.4, it thus follows that, for $X \in \{\text{SREG}, \text{SLIN}, \text{REG}, \text{LIN}\}$, we have that

$$\begin{aligned} Xc(\{0, 1\}^n \cdot \{\&\} \cdot \{a, b, \$, \#\}^{3c \cdot \lceil \log n \rceil + 2}) &\leq 2 \cdot n + 1 + 4 \cdot (3c \cdot \lceil \log n \rceil + 2) \\ &= 2 \cdot n + 12c \cdot \lceil \log n \rceil + 9. \end{aligned}$$

We also note that the context-free grammar $G = (\{S, A\}, \{0, 1, a, b, \$, \#, \&\}, P, S)$ with

$$P = \{S \rightarrow A^n \& B^{3c \cdot \lceil \log n \rceil + 2}, A \rightarrow 0, A \rightarrow 1, B \rightarrow a, B \rightarrow b, B \rightarrow \$, B \rightarrow \#\},$$

shows that $\text{CFC}(L_\varphi) \leq 7$.

Now, assume that φ is *not* a tautology, i.e., there is a truth assignment σ with

$$\sigma(x_1, x_2, \dots, x_n) = \sigma(x_1)\sigma(x_2)\cdots\sigma(x_n) = w,$$

for $w \in \{0, 1\}^n$, such that

$$\sigma(x_1, x_2, \dots, x_n) = w \not\models \varphi.$$

It suffices to consider the case $X = \text{CF}$ for the lower bound, since, by Equation 5.4, the other grammar types are less succinct than CF. Let G be a CF-grammar with

$$L(G) = L_\varphi \quad \text{and} \quad |G| = X_{\text{C}}(L_\varphi).$$

By Proposition 7.1.4, we thus get that

$$w \notin L(G_\varphi) = \{v \in \Sigma^* \mid v \models \varphi\}.$$

This then implies that

$$(w\&)^{-1}L(G) = \emptyset \cup T_{c \cdot \lceil \log n \rceil} = T_{c \cdot \lceil \log n \rceil}.$$

From Theorem 4.2.4 and Corollary 4.2.5, we get that

$$X_{\text{C}}((w\&)^{-1}L(G)) = X_{\text{C}}(T_{c \cdot \lceil \log n \rceil}) = \Omega(2^{c \cdot \log n}) = \Omega(n^c).$$

Since L_φ is a finite uniform language with $L_\varphi \neq \{\varepsilon\}$ and $\max\{|w| \mid w \in L_\varphi\} \geq 3$, an application of Theorem 7.1.9 yields that

$$X_{\text{C}}(T_{c \cdot \lceil \log n \rceil}) = \mathcal{O}(X_{\text{C}}(L_\varphi) \cdot n^4),$$

since

$$\max\{|w| \mid w \in L(G)\} = \mathcal{O}(n).$$

Therefore,

$$X_{\text{C}}(L_\varphi) = \Omega(n^{c-4}),$$

since

$$X_{\text{C}}(T_{c \cdot \lceil \log n \rceil}) = \Omega(n^c).$$

This concludes the proof of the lemma. \square

Now, we are ready to prove the main result of this section:

PROOF (OF THEOREM 7.1.1). We are finally in the position to fix the value of the constant c by choosing $c = 6$. Recall the definition of L_φ as

$$L_\varphi = L(G_\varphi) \cdot \{\&\} \cdot \{a, b, \$, \#\}^{3c \cdot \lceil \log n \rceil + 2} \cup \{0, 1\}^n \cdot \{\&\} \cdot T_{c \cdot \lceil \log n \rceil}.$$

From Proposition 7.1.3, we deduce that for the grammar G_φ it holds that

$$|G_\varphi| = \mathcal{O}(n^4).$$

When we combine the above upper bound with the upper bounds from Lemma 4.1.4, Theorem 4.2.4, and Corollary 4.2.5—using the bounds for union and concatenation (Corollary 6.2.2 and Theorem 6.3.4), we obtain that L_φ admits a strict regular grammar with p productions such that

$$p = \mathcal{O}(n^4) + \mathcal{O}(1) + \mathcal{O}(\log n) + \mathcal{O}(n) + \mathcal{O}(1) + \mathcal{O}(n^c) = \mathcal{O}(n^6).$$

Let $X \in \Delta$. Then, since, in Lemma 7.1.10, we have only obtained asymptotic bounds on the X -complexity of the language L_φ , we cannot literally apply Theorem 2.3.3 in order to obtain the inapproximability result. However, we can still follow the approach used in the proof of that theorem in order to arrive at our desired result: towards contradiction, assume that there is a polynomial-time algorithm A that approximates the minimal number of productions necessary to generate the finite language generated by a given X -grammar consisting of p productions within a factor of

$$o(p^{1/6}).$$

Then A could be used to decide in polynomial time whether φ is a tautology as follows: if φ is a tautology, then, again by Lemma 7.1.10,

$$Xc(L_\varphi) = \mathcal{O}(n),$$

for $X \in \Delta$, and otherwise, that is, if φ is not a tautology, then, by Lemma 7.1.10, we deduce that, for $X \in \Delta$, it holds that

$$Xc(L_\varphi) = \Omega(n^{c-4}) = \Omega(n^2).$$

Consequently, the putative approximation algorithm A returns a number of productions of at most

$$o(p^{1/6}) \cdot \mathcal{O}(n) = o(n) \cdot \mathcal{O}(n) = o(n^2)$$

if and only if φ is a tautology. However, this solves the coNP-hard 3-DNF tautology problem in deterministic polynomial time, which implies $P = NP$. This shows that the X -complexity, for $X \in \Delta$, of a given finite language cannot be approximated within a factor of $o(p^{1/6})$, unless $P = NP$. \square

7.2 The Smallest Grammar Problem for Finite Languages

The classic version of the smallest grammar problem asks for the smallest context-free grammar that generates a *single* given word. Its roots lie in the field of data compression and can be traced back to the papers [LZ76, ZL77, ZL78, SS82]. However, the first

explicit articulation of the smallest grammar problem came only a few years later [NM96, NMW97, YK00, KY00, KYNC00]. Its decision version has been shown to be NP-complete for both unbounded [CLL⁺05] and fixed alphabets of size at least 24 [CFG⁺16], but, recently, in [Fer19], it was stated that—in the journal version based on [CFG⁺16]—the latter result has already been improved to fixed alphabets of size at least 17. In [CLL⁺05], it was also shown that the smallest grammar problem (w.r.t. unbounded alphabets) has an approximation ratio of at least

$$\frac{8569}{8568},$$

unless $P = NP$. We will consider—w.r.t. fixed alphabets of size at least 5—another formulation of the smallest grammar problem that asks for the smallest grammar that generates a given *finite language* instead of just a single word. This formulation is similar to, yet different from the so-called *Generalized Smallest Grammar Problem* of [SG17]. There, the authors ask for the smallest non-recursive³ grammar whose language contains a specific given word (in other words, *covers* a specific given word). Moreover, their notion of grammar size additionally incorporates the cost of uniquely specifying the given word within the grammar. For this extended formulation of the smallest grammar problem, they also provide efficient algorithms that achieve smaller grammars than the state of the art on standard benchmarks.

The authors of [CLL⁺05] defined the size of a context-free grammar $G = (N, \Sigma, P, S)$ as the sum of the lengths of the right-hand sides of all productions. They slightly deviate from the classic definition from [Har78], which we have already introduced in Section 2.1.3 and which will also be used in this thesis:

$$|G|_s = \sum_{A \rightarrow \alpha \in P} (|\alpha| + 2).$$

Recall that, for $X \in \Delta$, the *minimal X-size* of a finite language L is defined as

$$Xsz(L) = \min\{|G|_s \mid G \text{ is an } X\text{-grammar with } L = L(G)\}.$$

The exponential lower bound on the X -complexity of the uniform language

$$T_n = \{w\$w\#w \mid w \in \{a, b\}^n\},$$

for an integer $n \geq 1$ and $X \in \Delta$, as shown in Theorem 4.2.4 and Corollary 4.2.5, immediately implies a lower bound of

$$Xsz(T_n) = \Omega(2^n)$$

on the size of a minimal X -grammar generating T_n , since, by Proposition 4.1.8, we can assume that the grammar is ε -free. Along similar lines as in the proof of Lemma 7.1.10, we get a linear upper bound on the minimal grammar size for L_φ if φ is a tautology.

³A context-free grammar $G = (N, \Sigma, P, S)$ is called *non-recursive* if, for all $A, B \in N$, it holds that if B occurs in a derivation of A , then A does not occur in a derivation of B .

However, the lower bound for the minimal grammar size of L_φ if φ is not a tautology asymptotically coincides with the one obtained for the minimal number of productions. Thus, Lemma 7.1.10, remains valid in the case that X_c is replaced by X_{sz} .

Lemma 7.2.1. *Let $X \in \Delta$ and let φ be a propositional formula in 3-DNF over n variables. Then*

$$X_{sz}(L_\varphi) = \begin{cases} \mathcal{O}(n) & \text{if } \varphi \text{ is a tautology,} \\ \Omega(n^{c-4}) & \text{otherwise.} \end{cases}$$

PROOF. First, assume that φ is a tautology. Then from Lemma 7.1.10, we know that

$$\text{SREG}_c(L_\varphi) = \mathcal{O}(n).$$

Since, in the case that φ is a tautology, we have that

$$L_\varphi = \{0, 1\}^n \cdot \{\&\} \cdot \{a, b, \$, \#\}^{3c \cdot \lceil \log n \rceil + 2},$$

the following strict regular grammar generates L_φ :

$$\begin{array}{ll} S \rightarrow 0A_2 \mid 1A_2 & B_1 \rightarrow aB_2 \mid bB_2 \mid \$B_2 \mid \#B_2 \\ A_2 \rightarrow 0A_3 \mid 1A_3 & B_2 \rightarrow aB_3 \mid bB_3 \mid \$B_3 \mid \#B_3 \\ \vdots & \vdots \\ A_n \rightarrow 0A_{n+1} \mid 1A_{n+1} & B_{3c \cdot \lceil \log n \rceil + 2} \rightarrow a \mid b \mid \$ \mid \#. \\ A_{n+1} \rightarrow \&B_1 & \end{array}$$

Thus, we get that

$$\text{SREG}_{sz}(L_\varphi) = \mathcal{O}(n).$$

Consequently, since strict regular grammars are less succinct than all other grammar types under consideration, and by definition of grammar size, we also have that

$$X_{sz}(L_\varphi) = \mathcal{O}(n),$$

for $X \in \Delta \setminus \{\text{SREG}\}$. In the case that φ is *not* a tautology, we know that

$$X_c(L_\varphi) = \Omega(n^{c-4}),$$

from Lemma 7.1.10. Since L_φ is a uniform language not containing ε , it follows from Proposition 4.1.8 that every minimal grammar generating L_φ is ε -free. Thus, the right-hand side of each production in such a minimal grammar has length at least one. Then, since every production in such a grammar consists of at least three symbols (that is, by counting all symbols on both the left- and right-hand side as well as the arrow symbol “ \rightarrow ”), it follows that

$$X_{sz}(L_\varphi) \geq 3 \cdot X_c(L_\varphi).$$

Thus, in particular,

$$\text{Xsz}(L_\varphi) = \Omega(n^{c-4}).$$

This concludes the proof of the lemma. \square

With the help of Lemma 7.2.1, we get an inapproximability result w.r.t. grammar size that is analogous to the result of Theorem 7.1.1 for the number of productions.

Theorem 7.2.2. *Let $X \in \Delta$. Given an X -grammar of size s generating a finite language L , it is impossible to approximate $\text{Xsz}(L)$ within a factor of $o(s^{1/7})$, unless $P = NP$.*

PROOF. Recall from the proof of Theorem 7.1.1 that the language L_φ admits a regular grammar with p productions such that

$$p = \mathcal{O}(n^6).$$

Since every word in the language L_φ is of length $\ell \geq 3$ with $\ell = \mathcal{O}(n)$, we can, by Lemma 7.1.6 and without loss of generality, assume that the right-hand side of each production in a minimal grammar that generates L_φ is of length linear in n . Thus, we obtain that the size of such a grammar is at most

$$s = \mathcal{O}(n^7).$$

Let $X \in \Delta$. Then, since, in Lemma 7.2.1, we have only obtained asymptotic bounds on the symbolic X -complexity of the language L_φ , we cannot literally apply Theorem 2.3.3 in order to obtain the inapproximability result. However, we can still follow the approach used in the proof of that theorem in order to arrive at our desired result: towards contradiction, assume that there is some polynomial-time algorithm A that approximates the minimal size of an X -grammar that generates the finite language generated by a given X -grammar of size s within a factor of

$$o(s^{1/7}).$$

Then A could be used to decide in polynomial time whether φ is a tautology as follows: Again, we set $c = 6$. Recall from Lemma 7.2.1 that

$$\text{Xsz}(L_\varphi) = \mathcal{O}(n)$$

if φ is a tautology, and

$$\text{Xsz}(L_\varphi) = \Omega(n^{c-4}) = \Omega(n^2),$$

otherwise. As a consequence, the putative approximation algorithm A returns a grammar size of at most

$$o(s^{1/7}) \cdot \mathcal{O}(n) = o(n) \cdot \mathcal{O}(n) = o(n^2)$$

if and only if φ is a tautology. However, this solves the coNP-hard 3-DNF tautology problem in deterministic polynomial time, which implies $P = NP$. This shows that, for $X \in \Delta$, the X -size of a given finite language cannot be approximated within a factor of $o(s^{1/7})$, unless $P = NP$. \square

Observe that our reduction scheme is robust enough to yield the same inapproximability result if we define the grammar size as in [CLL⁺05], i.e., as the sum of the right-hand sides of all productions. Thus, the result of Theorem 7.2.2 also holds for the alternative definition of grammar size.

7.3 The Uniform-Length Universality Problem and the ETH

Recently, a number of decision problems (see, e.g., [BI15, FHV15, Weh16, BGL17, FK17, FPSV17, PS18, dOOW20], for problems in the realm of formal language theory) have been investigated with respect to the so-called⁴

Exponential Time Hypothesis (ETH) [IP99, LMS11]:

For $k \geq 0$, let

$$s_k = \inf\{\delta \mid \text{there exists an } \mathcal{O}^*(2^{\delta \cdot n})\text{-time algorithm for solving } k\text{-SAT}\}.$$

Then, for $k \geq 3$, it holds that $s_k > 0$.

Intuitively, ETH states that, for $k \geq 3$, k -SAT does not have a subexponential-time algorithm. Recall that 3-SAT is the NP-complete problem of deciding whether a given propositional formula in 3-CNF consisting of n variables and m clauses (each of which contains at most three literals) is satisfiable. The Exponential Time Hypothesis is a stronger complexity assumption than $P \neq NP$ and is often used to obtain quantitative lower bounds on the running time of algorithms for NP-hard decision problems [CFK⁺15]. One of the seminal results related to the Exponential Time Hypothesis is the famous *Sparsification Lemma*, which allows to assume that the number of clauses of a given k -SAT formula is linear in the number of variables.

Theorem 7.3.1 (Sparsification Lemma, [IPZ01, Corollary 1]). *For all $\delta > 0$ and all positive integers k , there is a constant c such that any k -SAT formula φ with n variables can be expressed as*

$$\varphi = \bigvee_{i=1}^t \psi_i,$$

where $t \leq 2^{\delta \cdot n}$ and each ψ_i is a k -SAT formula with at most $c \cdot n$ clauses. Moreover, this disjunction can be computed by an algorithm running in time $\mathcal{O}^*(2^{\delta \cdot n})$.

As an immediate consequence of the Sparsification Lemma, we get the following theorem which will be an important ingredient in the proofs of Theorem 7.3.4 and Corollary 7.3.5.

Theorem 7.3.2 ([CFK⁺15, Theorem 14.4]). *Unless ETH fails, there is no algorithm that solves 3-SAT in time $\mathcal{O}^*(2^{o(n+m)})$, where n and m are the number of variables and clauses, respectively, in the given 3-SAT formula.*

⁴For a survey on related strong hypotheses, we refer the reader to [VW18].

The following observation, which we literally take from [CFK⁺15], can be used to transfer lower bounds between different problems.

Observation 7.3.3 ([CFK⁺15, Observation 14.7]). Suppose that there is a polynomial-time reduction from problem Π to problem Π' that, given an instance I of Π , constructs an equivalent instance of Π' having size at most $g(|I|)$, for some non-decreasing function g . Then the existence of an $\mathcal{O}^*(2^{o(f(|I|))})$ -time algorithm for Π' , for some non-decreasing function f , entails the existence of an $\mathcal{O}^*(2^{o(f(g(|I|)))})$ -time algorithm for Π . Δ

Remark. Therefore, by Observation 7.3.3, in order to exclude an algorithm for a problem Π' with running time $\mathcal{O}^*(2^{o(f(|I|))})$, we need to provide a reduction from the 3-SAT problem to Π' that outputs instances of size $\mathcal{O}(g(n + m))$, where g is the *inverse* of the function f [CFK⁺15].

In this section, we will—under the assumption of ETH—investigate the *uniform-length universality problem* for the grammar types in Δ :

UNIFORM-LENGTH UNIVERSALITY

INSTANCE: An X -grammar $G = (N, \Sigma, P, S)$, for $X \in \Delta$, and an integer $\ell \geq 0$.

QUESTION: Does it hold that $L(G) = \Sigma^\ell$?

We will follow the strategy described in the above remark in order to show the two subsequent main results of this section with the slight modification that we reduce from the coNP-complete co3-SAT problem. In Theorem 7.3.4, it is shown that under the assumption of ETH, there is no $\mathcal{O}^*(2^{o(p^{1/4})})$ -time algorithm that decides the uniform-length universality problem.

Theorem 7.3.4. *Unless ETH fails, there is no $\mathcal{O}^*(2^{o(p^{1/4})})$ -time algorithm that decides the uniform-length universality problem. Here, p is the number of productions of the given grammar.*

PROOF. We proceed by reducing the co3-SAT (that is, the complement of the 3-SAT problem) to the uniform-length universality problem. Let φ be an instance of co3-SAT, i.e., φ is a propositional formula in 3-CNF consisting of n variables and m clauses. Now, given the formula φ , we construct an instance (G, ℓ) of uniform-length universality as follows: the X -grammar G is constructed from the 3-DNF formula $\neg\varphi$ according to the polynomial-time reduction used in Section 7.1. That is,

$$G = G_{\neg\varphi}.$$

Moreover, we set

$$\ell = n.$$

Next, we show that the reduction is correct, i.e., we show that

$$\varphi \text{ is unsatisfiable if and only if } L(G) = \{0, 1\}^n.$$

First, note that, by propositional logic,

$$\varphi \text{ is unsatisfiable} \quad \text{if and only if} \quad \neg\varphi \text{ is a tautology.}$$

Thus, by Proposition 7.1.4,

$$\varphi \text{ is unsatisfiable} \quad \text{iff} \quad \neg\varphi \text{ is a tautology} \quad \text{iff} \quad L(G) = L(G_{\neg\varphi}) = \{0, 1\}^n.$$

Moreover, by Proposition 7.1.3, we know that

$$p = \mathcal{O}(n^4),$$

that is, the reduction can be computed in polynomial time.

Now that we have established the correctness of the reduction, assume that there is an algorithm with running time

$$\mathcal{O}^*(2^{o(p^{1/4})})$$

that decides uniform-length universality. Since

$$p = \mathcal{O}(n^4),$$

we could decide co3-SAT in time

$$\mathcal{O}^*(2^{o((n^4)^{1/4})}) = \mathcal{O}^*(2^{o(n)}),$$

which—under the assumption of ETH—is impossible by Theorem 7.3.2 and the fact that all deterministic time (and space) complexity classes are closed under complement [Pap95]. This finishes the proof of the theorem. \square

Using a similar line of reasoning as in the proof of the previous theorem, we can show that under the assumption of ETH, there is no $\mathcal{O}^*(2^{o(s^{1/4})})$ -time algorithm that decides the uniform-length universality problem.

Corollary 7.3.5. *Unless ETH fails, there is no $\mathcal{O}^*(2^{o(s^{1/4})})$ -time algorithm that decides the uniform-length universality problem. Here, s is the size of the given grammar.*

PROOF. For the reduction from co3-SAT to uniform-length universality, we proceed exactly as in the proof of Theorem 7.3.4. By Proposition 7.1.3, we know that

$$s = \mathcal{O}(n^4).$$

Assume that there is an algorithm with running time

$$\mathcal{O}^*(2^{o(s^{1/4})})$$

that decides uniform-length universality. Since

$$s = \mathcal{O}(n^4),$$

we could decide co3-SAT in time

$$\mathcal{O}^*(2^{o((n^4)^{1/4})}) = \mathcal{O}^*(2^{o(n)}),$$

which—under the assumption of ETH—is impossible by Theorem 7.3.2 and the fact that all deterministic time (and space) complexity classes are closed under complement [Pap95]. This finishes the proof. \square

Conclusion

We have investigated several different questions regarding context-free grammars that generate finite languages from the points of view of descriptive and computational complexity.

In Chapter 3, we have studied cover complexity measures for finite languages on three different levels of abstraction and shown that every complexity measure on finite languages naturally induces a corresponding cover complexity measure. Moreover, we have characterised the situations in which arbitrary complexity measures obtained in this way are unbounded. In particular, we have seen that all considered grammar types which are strictly weaker than context-free grammars induce an unbounded complexity measure. Based on these rather abstract results, we have shown that every class of context-free grammars that allows only a bounded number of nonterminals on the right-hand side of each production induces an unbounded production cover complexity measure. This, in turn, entails that the production cover complexity of a finite language L can be obtained as the minimum of the exact production complexities of a finite number of finite supersets L' of L .

Next, in Chapter 4, we have obtained several upper and lower bounds on various different complexity measures for both arbitrary and specific finite languages. In particular, we have proved bounds on the exact, cover, and scattered complexity of finite languages. By using the standard argument of [BMCIW81], we have shown that there are specific finite languages that are incompressible with respect to the exact complexity of certain grammar types. With the help of this standard technique, we were also able to obtain a lower bound on the symbolic complexity of the language

$$T_n = \{w\$w\#w \mid w \in \{a, b\}^n\},$$

which is more precise than the one obtained using the lower bound technique of [Fil11]. Moreover, we have also generalised the cover-incompressible sequence of finite languages constructed in [EH15a, EH18].

We have considered several different production complexity measures for finite languages with respect to three interpretations of approximation of finite languages (namely, equivalence, cover, and scattered cover as well as their infinite counterparts) in Chapter 5. In the case of the infinite variants, the language of the grammar is allowed to be infinite but its intersection with all words up to a certain length has to approximate the given finite language in the correct manner. Based on a group of relations that are inspired by the taxonomy with respect to the nonterminal complexity measures of [DP89], we have related these production complexity measures with each other and obtained relative succinctness classifications of both grammar and measure types. In particular, we have obtained two relative succinctness classifications in the following senses:

1. Fix a measure type τ and compare the different grammar types under consideration with each other with respect to the measure type τ .
2. Fix a grammar type X and compare the different measure types under consideration with each other with respect to the measure type X .

The relative succinctness classifications for grammar and measure types are summarised in Figures 53 and 55, respectively. However, as can be seen in these figures, both classifications are not complete yet, i.e., there are still a few unsettled problems left. One result that was surprising is that the scattered complexity measure is more succinct (with respect to all four relations) than the infinite scattered complexity for all non-strict grammar types. For the exact and cover complexity measures this situation is reversed, as in both cases the infinite variant is more succinct (with respect to all four relations) than its finite counterpart. The only exception is that it is still open whether

$$cc_{\infty} \leq_{CF}^i cc$$

holds for any $i \in \{1, 2, 3\}$. In many cases, it was enough to use bounds that have already been obtained in Chapter 4 in order to prove or disprove relations between certain measure or grammar types. However, for the infinite exact complexity, it was rather tedious to show certain lower bounds. Particularly, for the languages

$$P_n = \{w\$w^R \mid w \in \{a, b\}^{\leq n}\} \quad \text{and} \quad T_n = \{w\$w\#w \mid w \in \{a, b\}^n\},$$

we needed to utilise triple constructions in order to obtain an exponential lower bound on the infinite (strict) regular complexity of P_n and, for $X \in \Delta$, on the infinite X -complexity of T_n based on the corresponding exponential lower bounds on the finite complexity. The lower bound on P_n was needed in order to show that strict regular and regular grammars are less succinct with respect to the infinite exact complexity than strict linear, linear, and context-free grammars. Moreover, the lower bound on T_n was required in order to show that, e.g., both the finite and infinite exact complexity are *not* more succinct than the cover and the scattered complexity as well as the infinite cover and the infinite scattered complexity with respect to the grammar types in Δ .

Chapter 6 was devoted to investigating both the exact and the cover complexity of the language operations intersection, union, and concatenation with respect to finite

languages for several different types of context-free grammars (for the obtained bounds, see Figure 61). The proofs for showing these bounds work in many cases for both the exact and the cover complexity without any amendment. While for the cover complexity it is rather easy to obtain tight bounds for intersection, we still have not found a way to obtain a (tight) upper bound on concatenation with respect to the exact complexity. As we have seen, in contrast to union, there is no uniform upper bound on both the exact and the cover complexity of concatenation. In particular, the upper bound on concatenation for strict linear and linear grammars differs from the one for the other grammar types. This is caused by the fact that if we use the construction that is used to combine two regular grammars into a new regular grammar which generates (or covers) the concatenation of their generated (or covered, respectively) languages, then this does not necessarily give us a linear grammar again if we are given two linear grammars. Moreover, with the help of the newly constructed regular cover-incompressible sequence of finite languages, we were also able to show that the upper bound on the cover complexity of union is tight for regular grammars. However, we have not yet been able to obtain tight bounds in all cases, therefore, as depicted in Figure 61, there are still a few problems left open.

Finally, in Chapter 7, we have studied the computational complexity of some grammar-based problems. More precisely, we have shown that, for fixed alphabets of cardinality at least 5, given an arbitrary context-free grammar G with p productions that generates a finite language L , it is impossible to approximate the minimal number of productions necessary to generate L within a factor of

$$o(p^{1/6}),$$

unless $P = NP$. In addition, we have also shown that, given an arbitrary context-free grammar G with size s that generates a finite language L , the minimal size of a context-free grammar that generates L cannot be approximated within a factor of

$$o(s^{1/7}),$$

unless $P = NP$. The latter result complements the result that any approximation algorithm for the smallest grammar problem must have an approximation ratio of at least $\frac{8569}{8568}$, unless $P = NP$, as shown in [CLL⁺05]. This classic variant of the smallest grammar problem asks for the smallest (in terms of size) context-free grammar that generates exactly a single given word. Furthermore, we have also studied the uniform-length universality problem under the assumption of the Exponential Time Hypothesis. In particular, we have shown that if the Exponential Time Hypothesis holds, then there is no algorithm that decides uniform-length universality in time

$$\mathcal{O}^*\left(2^{o(p^{1/4})}\right),$$

where p is the number of productions of the given context-free grammar G . Similarly, we have also shown that if the Exponential Time Hypothesis holds, then there is no algorithm

that decides uniform-length universality in time

$$\mathcal{O}^*\left(2^{o(s^{1/4})}\right),$$

where s is the size of the given context-free grammar G . For the proofs of the inapproximability results, we have used several results that have already been obtained in previous chapters. The gap-reduction from the coNP-complete tautology problem uses the fact that the language T_n can only be generated by a grammar with an exponential number of productions/size. In particular, given a propositional formula φ in 3-DNF, we have embedded the language $T_{c \cdot \lceil \log n \rceil}$ into a language L_φ that needs a grammar with a linear number of productions/size if φ is a tautology and a grammar with at least a quadratic number of productions/size otherwise. An intermediate step in this gap-reduction featured a polynomial-time reduction from the tautology problem to the uniform-length universality problem. As a consequence, the uniform-length universality problem was shown to be coNP-hard. While showing the bound if φ is a tautology was rather straightforward, obtaining the bound if φ is not a tautology needed significantly more work. In particular, it was necessary to apply some sort of triple construction from [GS63] that constructs a grammar generating the left quotient of a word with the language of a grammar. This was necessary in order to use the lower bound on the language $T_{c \cdot \lceil \log n \rceil}$ to get a lower bound of a similar order on the language L_φ . The aforementioned reduction from tautology to uniform-length universality also came in handy in order to obtain our results under the assumption of the Exponential Time Hypothesis. Basically, the upper bound on the number of productions/size of the constructed grammar G_φ together with the correctness of the reduction already provided the necessary ingredients.

In [EEH18], it was shown that the minimal cover problem for acyclic regular grammars with a fixed bound on the number of nonterminals is NP-complete. The minimal cover problem is defined as follows: given a finite language L and a non-negative integer k , is there an acyclic regular grammar G such that G has at most k productions and satisfies $L(G) \supseteq L$? However, the computational complexity of this problem for an arbitrary number of nonterminals is still open. In [EEH18], the authors proved that it is in NP and conjectured that it is also NP-hard.

Due to the fact that all finite languages are regular, the existence of descriptional systems that accept or generate a given finite language is rather trivial: just take, for instance, finite automata or regular grammars. As a consequence, during investigations of finite languages, the attention shifts towards questions dealing with quantitative measures for finite languages based on the chosen descriptional system, such as the minimal number of states in an automaton or the minimal size of a context-free grammar. However, answering the question of how many productions a minimal grammar of a certain type needs in order to generate certain finite languages can be a difficult task.

This task becomes even harder for certain types of grammars if we move from the exact to the cover complexity. Here, the difficulty lies, among others, in the fact that, in general, those words which are generated by the grammar that are not members of the covered

finite language, can be of arbitrary shape and length. Moreover, the cover formulation makes only sense if we restrict the covering grammar in such a way that it has to generate a finite language. Otherwise, finding a grammar with a small number of productions whose generated language covers any finite language becomes trivial—just take the strict regular grammar that generates the universal language Σ^* .

The results of [EH15a, EH18] can be seen as a first step in transferring descriptonal complexity results from the realm of formal language theory to the one of proof theory. By constructing a regular cover-incompressible sequence of finite word languages, it was first shown that this result extends to tree languages in a straightforward way and then it was used to show that there is a sequence of formulae which can be compressed at most quadratically with respect to the size of a smallest cut-free¹ proof. Thus, at least in the case of regular grammars, studying questions of descriptonal complexity of finite languages with respect to the cover formulation can have interesting ramifications in the realm of proof complexity. In this sense, one can regard proof theory as being well integrated into the study of descriptonal complexity of finite languages.

In summary, we believe that the study of the complexity of problems based on finite languages is a fruitful research area with strong ties to proof theory as well as to more classic questions of descriptonal and computational complexity.

¹A *cut* is a special inference rule that, roughly speaking, formalises the use of a lemma in a mathematical proof.

List of Figures

51	DFCA \mathcal{A}_1 for the finite language $\{a_1, a_2, \dots, a_k\}^{\leq n}$	72
52	Minimal DFA \mathcal{A}_2 accepting the finite language $\{a_1, a_2, \dots, a_k\}^{\leq n}$	72
53	Relations between the grammar types w.r.t. a fixed measure type.	78
54	NFA \mathcal{A} for the finite language $\Sigma^{\leq n}$	87
55	Relations between the measure types w.r.t. a fixed grammar type.	99
56	DFA \mathcal{A} for the finite language Σ^n	103
61	Summary of descriptive complexity results for language operations.	122

Index of Notation and Abbreviations

2NF Binary normal form. 154

3-CNF 3-conjunctive normal form. See k -CNF.

3-DNF 3-disjunctive normal form. See k -DNF.

3-SAT The propositional 3-satisfiability problem. 165, see also k -SAT & SAT.

c Exact complexity measure type. 29, see also X_c .

c_∞ Infinite exact complexity measure type. 73, see also X_{c_∞} .

cc Cover complexity measure type. 29, see also X_{cc} .

cc_∞ Infinite cover complexity measure type. 73, see also X_{cc_∞} .

CF Context-free. 15

CF_c Exact complexity measure for context-free grammars. See X_c .

CF_{c_∞} Infinite exact complexity measure for context-free grammars. See X_{c_∞} .

CF_{cc} Cover complexity measure for context-free grammars. See X_{cc} .

CF_{cc_∞} Infinite cover complexity measure for context-free grammars. See X_{cc_∞} .

CFG Context-free grammar. 15

CF_{sc} Scattered complexity measure for context-free grammars. See X_{sc} .

CF_{sc_∞} Infinite scattered complexity measure for context-free grammars. See X_{sc_∞} .

CF_{sz} Symbolic complexity measure for context-free grammars. See X_{sz} .

CNF Chomsky normal form. 39

CNF_{cc} Cover complexity measure for grammars in CNF. See CNF & X_{cc} .

$co3$ -SAT The complement of the 3-SAT problem. 166, see also 3-SAT & $coNP$.

coNP The class of problems which are complements of problems in NP. 21, *see also* NP.

DFA Deterministic finite automaton. 18

DFCA Deterministic finite cover automaton. 20, *see also* DFA.

ETH Exponential Time Hypothesis. 165

k -CNF k -conjunctive normal form. 22

k -DNF k -disjunctive normal form. 22

k -SAT The propositional k -satisfiability problem. 23, *see also* 3-SAT & SAT.

LIN Linear (grammar). 16

LIN_c Exact complexity measure for linear grammars. *See* X_c .

LIN_c_∞ Infinite exact complexity measure for linear grammars. *See* $X_{c_∞}$.

LIN_{cc} Cover complexity measure for linear grammars. *See* X_{cc} .

LIN_{cc}_∞ Infinite cover complexity measure for linear grammars. *See* $X_{cc_∞}$.

LIN_{sc} Scattered complexity measure for linear grammars. *See* X_{sc} .

LIN_{sc}_∞ Infinite scattered complexity measure for linear grammars. *See* $X_{sc_∞}$.

LIN_{sz} Symbolic complexity measure for linear grammars. *See* X_{sz} .

\mathcal{M} The set of all measure types. 73, *see also* $\mathcal{M}_∞$ & \mathcal{M}_{fin} .

$\mathcal{M}_∞$ The set of infinite measure types. 73, *see also* \mathcal{M} & \mathcal{M}_{fin} .

\mathcal{M}_{fin} The set of finite measure types. 73, *see also* \mathcal{M} & $\mathcal{M}_∞$.

NFA Nondeterministic finite automaton. 19

NFCA Nondeterministic finite cover automaton. 20, *see also* NFA.

NP The class of problems which are decidable in nondeterministic polynomial time. 21

NPO NP-optimisation problem. 24, *see also* NP.

P The class of problems which are decidable in deterministic polynomial time. 21

PNF Pruned normal form. 35

REG Regular (grammar). 16

- REGc** Exact complexity measure for regular grammars. *See* [Xc](#).
- REGc_∞** Infinite exact complexity measure for regular grammars. *See* [Xc_∞](#).
- REGcc** Cover complexity measure for regular grammars. *See* [Xcc](#).
- REGcc_∞** Infinite cover complexity measure for regular grammars. *See* [Xcc_∞](#).
- REGsc** Scattered complexity measure for regular grammars. *See* [Xsc](#).
- REGsc_∞** Infinite scattered complexity measure for regular grammars. *See* [Xsc_∞](#).
- REGsz** Symbolic complexity measure for regular grammars. *See* [Xsz](#).
- SAT** The propositional satisfiability problem. [23](#), *see also* [3-SAT](#) & [k-SAT](#).
- sc** Scattered complexity measure type. [46](#), *see also* [Xsc](#).
- sc_∞** Infinite scattered complexity measure type. [73](#), *see also* [Xsc_∞](#).
- SLIN** Strict linear (grammar). [16](#)
- SLINc** Exact complexity measure for strict linear grammars. *See* [Xc](#).
- SLINc_∞** Infinite exact complexity measure for strict linear grammars. *See* [Xc_∞](#).
- SLINcc** Cover complexity measure for strict linear grammars. *See* [Xcc](#).
- SLINcc_∞** Infinite cover complexity measure for strict linear grammars. *See* [Xcc_∞](#).
- SLINsc** Scattered complexity measure for strict linear grammars. *See* [Xsc](#).
- SLINsc_∞** Infinite scattered complexity measure for strict linear grammars. *See* [Xsc_∞](#).
- SLINsz** Symbolic complexity measure for strict linear grammars. *See* [Xsz](#).
- SREG** Strict regular (grammar). [16](#)
- SREGc** Exact complexity measure for strict regular grammars. *See* [Xc](#).
- SREGc_∞** Infinite exact complexity measure for strict regular grammars. *See* [Xc_∞](#).
- SREGcc** Cover complexity measure for strict regular grammars. *See* [Xcc](#).
- SREGcc_∞** Infinite cover complexity measure for strict regular grammars. *See* [Xcc_∞](#).
- SREGsc** Scattered complexity measure for strict regular grammars. *See* [Xsc](#).
- SREGsc_∞** Infinite scattered complexity measure for strict regular grammars. *See* [Xsc_∞](#).
- SREGsz** Symbolic complexity measure for strict regular grammars. *See* [Xsz](#).

sz Symbolic complexity measure type. 56, *see also* [Xsz](#).

Xc Exact complexity measure for grammars of type X. 28

Xc_∞ Infinite exact complexity measure for grammars of type X. 71

Xcc Cover complexity measure for grammars of type X. 28

Xcc_∞ Infinite cover complexity measure for grammars of type X. 72

Xsc Scattered complexity measure for grammars of type X. 46

Xsc_∞ Infinite scattered complexity measure for grammars of type X. 72

Xsz Symbolic complexity measure for grammars of type X. 56

Bibliography

- [AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.
- [AER83] Brian Alspach, Peter Eades, and Gordon Rose. A lower-bound for the number of productions required for a certain class of languages. *Discrete Applied Mathematics*, 6(2):109–115, 1983.
- [BDG85] José L. Balcázar, Josep Díaz, and Joaquim Gabarró. Uniform characterizations of non-uniform complexity measures. *Information and Control*, 67(1–3):53–69, 1985.
- [BGL17] Karl Bringmann, Allan Grønlund, and Kasper G. Larsen. A Dichotomy for Regular Expression Membership Testing. In *Proceedings of the 58th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2017)*, pages 307–318, New Jersey, 2017. IEEE.
- [BI15] Arturs Backurs and Piotr Indyk. Which Regular Expression Patterns are Hard to Match? *CoRR*, abs/1511.07070, 2015.
- [Bir92] Jean-Camille Birget. Intersection and union of regular languages and state complexity. *Information Processing Letters*, 43(4):185–190, 1992.
- [BK99] Norbert Blum and Robert Koch. Greibach Normal Form Transformation Revisited. *Information and Computation*, 150(1):112–118, 1999.
- [BMCI83] Walter Bucher, Hermann A. Maurer, and Karel Culik II. Context-free complexity of finite languages. *Theoretical Computer Science*, 28(3):277–285, 1983.
- [BMCIW81] Walter Bucher, Hermann A. Maurer, Karel Culik II, and Detlef Wotschke. Concise description of finite languages. *Theoretical Computer Science*, 14(3):227–246, 1981.
- [Buc81] Walter Bucher. A note on a problem in the theory of grammatical complexity. *Theoretical Computer Science*, 14(3):337–344, 1981.

- [Câm14] Cezar Câmpeanu. Simplifying Nondeterministic Finite Cover Automata. In Zoltán Ésik and Zoltán Fülöp, editors, *Proceedings of the 14th International Conference on Automata and Formal Languages (AFL 2014)*, volume 151, pages 162–173, Waterloo, 2014. Open Publishing Association.
- [Câm15] Cezar Câmpeanu. Non-Deterministic Finite Cover Automata. *Scientific Annals of Computer Science*, 25(1):3–28, 2015.
- [CCISY01] Cezar Câmpeanu, Karel Culik II, Kai Salomaa, and Sheng Yu. State Complexity of Basic Operations on Finite Languages. In Oliver Boldt and Helmut Jürgensen, editors, *Proceedings of the 4th International Workshop on Implementing Automata (WIA 1999)*, volume 2214 of *Lecture Notes in Computer Science*, pages 60–70, Berlin, Heidelberg, 2001. Springer.
- [CFG⁺16] Katrin Casel, Henning Fernau, Serge Gaspers, Benjamin Gras, and Markus L. Schmid. On the Complexity of Grammar-Based Compression over Fixed Alphabets. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *Proceedings of the 43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*, volume 55 of *Leibniz International Proceedings in Informatics*, pages 122:1–122:14, Saarbrücken/Wadern, 2016. Dagstuhl Publishing.
- [CFK⁺15] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshantov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.
- [CGH05] Jean-Marc Champarnaud, Franck Guingne, and Georges Hansel. Similarity relations and cover automata. *RAIRO - Theoretical Informatics and Applications*, 39(1):115–123, 2005.
- [CKP05] Cezar Câmpeanu, Lila Kari, and Andrei Păun. Results on Transforming NFA into DFCA. *Fundamenta Informaticae*, 64(1–4):53–63, 2005.
- [CLL⁺05] Moses Charikar, Eric Lehman, Ding Liu, Rina Panigrahy, Manoj Prabhakaran, Amit Sahai, and Abhi Shelat. The Smallest Grammar Problem. *IEEE Transactions on Information Theory*, 51(7):2554–2576, 2005.
- [CMR11] Cezar Câmpeanu, Nelma Moreira, and Rogério Reis. Expected Compression Ratio for DFCA: experimental average case analysis. Technical report, Departamento de Ciência de Computadores, Universidade do Porto, 2011.
- [CMR16] Cezar Câmpeanu, Nelma Moreira, and Rogério Reis. On the dissimilarity operation on finite languages. In Henning Bordihn, Rudolf Freund, Benedek Nagy, and György Vaszil, editors, *Proceedings of the Eighth Workshop on Non-Classical Models of Automata and Applications (NCMA 2016)*, volume 321 of *books@ocg.at*, pages 105–120, Wien, 2016. Österreichische Computer Gesellschaft.

- [Coo71] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing (STOC 1971)*, pages 151–158, New York, 1971. ACM.
- [CP03a] Cezar Câmpeanu and Andrei Păun. Counting the Number of Minimal DFCA Obtained by Merging States. *International Journal of Foundations of Computer Science*, 14(6):995–1006, 2003.
- [CP03b] Cezar Câmpeanu and Andrei Păun. The Number of Similarity Relations and the Number of Minimal Deterministic Finite Cover Automata. In Jean-Marc Champarnaud and Denis Maurel, editors, *Proceedings of the 7th International Conference on Implementation and Application of Automata (CIAA 2002)*, volume 2608 of *Lecture Notes in Computer Science*, pages 67–76, Berlin, Heidelberg, 2003. Springer.
- [CP05] Cezar Câmpeanu and Andrei Păun. Tight Bounds for NFA to DFCA Transformations for Binary Alphabets. In Michael Domaratzki, Alexander Okhotin, Kai Salomaa, and Sheng Yu, editors, *Proceedings of the 9th International Conference on Implementation and Application of Automata (CIAA 2004)*, volume 3317 of *Lecture Notes in Computer Science*, pages 306–307, Berlin, Heidelberg, 2005. Springer.
- [CPS06a] Cezar Câmpeanu, Andrei Păun, and Jason R. Smith. An Incremental Algorithm for Constructing Minimal Deterministic Finite Cover Automata. In Jacques Farré, Igor Litovsky, and Sylvain Schmitz, editors, *Proceedings of the 10th International Conference on Implementation and Application of Automata (CIAA 2005)*, volume 3845 of *Lecture Notes in Computer Science*, pages 90–103, Berlin, Heidelberg, 2006. Springer.
- [CPS06b] Cezar Câmpeanu, Andrei Păun, and Jason R. Smith. Incremental construction of minimal deterministic finite cover automata. *Theoretical Computer Science*, 363(2):135–148, 2006.
- [CPY02] Cezar Câmpeanu, Andrei Păun, and Sheng Yu. An Efficient Algorithm for Constructing Minimal Cover Automata for Finite Languages. *International Journal of Foundations of Computer Science*, 13(1):83–97, 2002.
- [Cre97] Pierluigi Crescenzi. A Short Guide to Approximation Preserving Reductions. In *Proceedings of the Twelfth Annual IEEE Conference on Computational Complexity*, pages 262–273, Los Alamitos, 1997. IEEE.
- [CSY99] Cezar Câmpeanu, Nicolae Sântean, and Sheng Yu. Minimal Cover-Automata for Finite Languages. In Jean-Marc Champarnaud, Djelloul Ziadi, and Denis Maurel, editors, *Proceedings of the Third International Workshop on Implementing Automata (WIA 1998)*, volume 1660 of *Lecture Notes in Computer Science*, pages 43–56, Berlin, Heidelberg, 1999. Springer.

- [CSY00] Cezar Câmpeanu, Kai Salomaa, and Sheng Yu. State Complexity of Regular Languages: Finite versus Infinite. In Cristian Calude and Gheorghe Păun, editors, *Finite Versus Infinite*, Discrete Mathematics and Theoretical Computer Science, pages 53–73. Springer, London, 2000.
- [CSY01] Cezar Câmpeanu, Nicolae Sântean, and Sheng Yu. Minimal cover-automata for finite languages. *Theoretical Computer Science*, 267(1–2):3–16, 2001.
- [Das17] Jürgen Dassow. Descriptive Complexity and Operations—Two Non-classical Cases. In Giovanni Pighizzini and Cezar Câmpeanu, editors, *Proceedings of the 19th IFIP WG 1.02 International Conference on Descriptive Complexity of Formal Systems (DCFS 2017)*, volume 10316 of *Lecture Notes in Computer Science*, pages 33–44, Cham, 2017. Springer.
- [DH12a] Jürgen Dassow and Ronny Harbich. Descriptive Complexity of Union and Star on Context-Free Languages. *Journal of Automata, Languages and Combinatorics*, 17(2–4):123–143, 2012.
- [DH12b] Jürgen Dassow and Ronny Harbich. Production Complexity of Some Operations on Context-Free Languages. In Martin Kutrib, Nelma Moreira, and Rogério Reis, editors, *Proceedings of the 14th International Workshop on Descriptive Complexity of Formal Systems (DCFS 2012)*, volume 7386 of *Lecture Notes in Computer Science*, pages 141–154, Berlin, Heidelberg, 2012. Springer.
- [DKH11] Ding-Zhu Du, Ker-I Ko, and Xiaodong Hu. *Design and Analysis of Approximation Algorithms*. Springer, 2011.
- [dOOW20] Mateus de Oliveira Oliveira and Michael Wehar. On the Fine Grained Complexity of Finite Automata Non-Emptiness of Intersection. In *Proceedings of the 24th International Conference on Developments in Language Theory (DLT 2020)*, Lecture Notes in Computer Science, Cham, 2020. Springer. To appear.
- [DP89] Jürgen Dassow and Gheorghe Păun. *Regulated Rewriting in Formal Language Theory*, volume 18 of *EATCS Monographs in Theoretical Computer Science*. Springer, 1989.
- [DS90] Cynthia Dwork and Larry Stockmeyer. A Time Complexity Gap for Two-Way Probabilistic Finite-State Automata. *SIAM Journal on Computing*, 19(6):1011–1023, 1990.
- [DS07] Michael Domaratzki and Kai Salomaa. Transition complexity of language operations. *Theoretical Computer Science*, 387(2):147–154, 2007.
- [DS08] Jürgen Dassow and Ralf Stiebe. Nonterminal Complexity of Some Operations on Context-Free Languages. *Fundamenta Informaticae*, 83(1-2):35–49, 2008.

- [EEH17] Sebastian Eberhard, Gabriel Ebner, and Stefan Hetzl. Algorithmic Compression of Finite Tree Languages by Rigid Acyclic Grammars. *ACM Transactions on Computational Logic*, 18(4):26:1–26:20, 2017.
- [EEH18] Sebastian Eberhard, Gabriel Ebner, and Stefan Hetzl. Complexity of Decision Problems on Totally Rigid Acyclic Tree Grammars. In Mizuho Hoshi and Shinnosuke Seki, editors, *Proceedings of the 22nd International Conference on Developments in Language Theory (DLT 2018)*, volume 11088 of *Lecture Notes in Computer Science*, pages 291–303, Cham, 2018. Springer.
- [EH15a] Sebastian Eberhard and Stefan Hetzl. Compressibility of Finite Languages by Grammars. In Jeffrey O. Shallit and Alexander Okhotin, editors, *17th International Workshop on Descriptive Complexity of Formal Systems (DCFS 2015)*, volume 9118 of *Lecture Notes in Computer Science*, pages 93–104, Cham, 2015. Springer.
- [EH15b] Sebastian Eberhard and Stefan Hetzl. Inductive theorem proving based on tree grammars. *Annals of Pure and Applied Logic*, 166(6):665–700, 2015.
- [EH18] Sebastian Eberhard and Stefan Hetzl. On the compressibility of finite languages and formal proofs. *Information and Computation*, 259(2):191–213, 2018.
- [EHL⁺19] Gabriel Ebner, Stefan Hetzl, Alexander Leitsch, Giselle Reis, and Daniel Weller. On the Generation of Quantified Lemmas. *Journal of Automated Reasoning*, 63(1):95–126, 2019.
- [EKSW05] Keith Ellul, Bryan Krawetz, Jeffrey Shallit, and Ming-wei Wang. Regular Expressions: New Results and Open Problems. *Journal of Automata, Languages and Combinatorics*, 10(4):407–437, 2005.
- [Ell04] Keith Ellul. Descriptive Complexity Measures of Regular Languages. Master’s thesis, University of Waterloo, 2004.
- [Fer19] Henning Fernau. Modern Aspects of Complexity Within Formal Languages. In Carlos Martín-Vide, Alexander Okhotin, and Dana Shapira, editors, *Proceedings of the 13th International Conference on Language and Automata Theory and Applications (LATA 2019)*, volume 11417 of *Lecture Notes in Computer Science*, pages 3–30, Cham, 2019. Springer.
- [FHV15] Henning Fernau, Pinar Heggernes, and Yngve Villanger. A multi-parameter analysis of hard problems on deterministic finite automata. *Journal of Computer and System Sciences*, 81(4):747–765, 2015.
- [Fil11] Yuval Filmus. Lower bounds for context-free grammars. *Information Processing Letters*, 111(18):895–898, 2011.

- [FK17] Henning Fernau and Andreas Krebs. Problems on Finite Automata and the Exponential Time Hypothesis. *Algorithms*, 10(1), 2017.
- [FPSV17] Henning Fernau, Meenakshi Paramasivan, Markus L. Schmid, and Vojtěch Vorel. Characterization and complexity results on jumping finite automata. *Theoretical Computer Science*, 679:31–52, 2017.
- [Geo96] Gianina Georgescu. The Orthogonality of Some Complexity Measures of Context-Free Languages. In Jürgen Dassow, Grzegorz Rozenberg, and Arto Salomaa, editors, *Proceedings of the 2nd International Conference on Developments in Language Theory (DLT 1995)*, pages 73–78, Singapore, 1996. World Scientific.
- [GH05] Herman Gruber and Markus Holzer. A note on the number of transitions of nondeterministic finite automata. In Henning Fernau, editor, *Proceedings of the 15. Theorietag der GI-Fachgruppe 0.1.5 Automaten und Formale Sprachen*, pages 24–25, Tübingen, 2005. Wilhelm-Schickard-Institut für Informatik.
- [GHJ15] Hermann Gruber, Markus Holzer, and Sebastian Jakobi. More on Deterministic and Nondeterministic Finite Cover Automata. In Frank Drewes, editor, *Proceedings of the 20th International Conference on Implementation and Application of Automata (CIAA 2015)*, volume 9223 of *Lecture Notes in Computer Science*, pages 114–126, Cham, 2015. Springer.
- [GHJ17] Hermann Gruber, Markus Holzer, and Sebastian Jakobi. More on deterministic and nondeterministic finite cover automata. *Theoretical Computer Science*, 679:18–30, 2017.
- [GHW18] Hermann Gruber, Markus Holzer, and Simon Wolfsteiner. On Minimal Grammar Problems for Finite Languages. In Mizuho Hoshi and Shinnosuke Seki, editors, *Proceedings of the 22nd International Conference on Developments in Language Theory (DLT 2018)*, volume 11088 of *Lecture Notes in Computer Science*, pages 342–353, Cham, 2018. Springer.
- [GMRY16] Yuan Gao, Nelma Moreira, Rogério Reis, and Sheng Yu. A Survey on Operational State Complexity. *Journal of Automata, Languages and Combinatorics*, 21(4):251–310, 2016.
- [Gru67] Jozef Gruska. On a Classification of Context-Free Languages. *Kybernetika*, 3(1):22–29, 1967.
- [Gru69] Jozef Gruska. Some classifications of context-free languages. *Information and Control*, 14(2):152–179, 1969.
- [Gru71] Jozef Gruska. Complexity and unambiguity of context-free grammars and languages. *Information and Control*, 18(5):502–519, 1971.

- [Gru72] Jozef Gruska. On the Size of Context-free Grammars. *Kybernetika*, 8(3):213–218, 1972.
- [Gru76] Jozef Gruska. Descriptive complexity (of languages): a short survey. In Antoni Mazurkiewicz, editor, *Proceedings of the 5th International Symposium on Mathematical Foundations of Computer Science (MFCS 1976)*, volume 45 of *Lecture Notes in Computer Science*, pages 65–80, Berlin, Heidelberg, 1976. Springer.
- [GS63] Seymour Ginsburg and Edwin H. Spanier. Quotients of Context-Free Languages. *Journal of the ACM*, 10(4):487–492, 1963.
- [Har78] Michael A. Harrison. *Introduction to Formal Language Theory*. Addison-Wesley, 1978.
- [Het12] Stefan Hetzl. Applying Tree Languages in Proof Theory. In Adrian-Horia Dediu and Carlos Martín-Vide, editors, *Proceedings of the 6th International Conference on Language and Automata Theory and Applications (LATA 2012)*, volume 7183 of *Lecture Notes in Computer Science*, pages 301–312, Berlin, Heidelberg, 2012. Springer.
- [HK03] Markus Holzer and Martin Kutrib. Nondeterministic Descriptive Complexity of Regular Languages. *International Journal of Foundations of Computer Science*, 14(6):1087–1102, 2003.
- [HK09] Markus Holzer and Martin Kutrib. Nondeterministic Finite Automata—Recent Results on the Descriptive and Computational Complexity. *International Journal of Foundations of Computer Science*, 20(4):563–580, 2009.
- [HK11] Markus Holzer and Martin Kutrib. Descriptive Complexity—An Introductory Survey. In Carlos Martín-Vide, editor, *Scientific Applications of Language Methods*, pages 1–58. Imperial College Press, London, 2011.
- [HLR⁺14] Stefan Hetzl, Alexander Leitsch, Giselle Reis, Janos Tapolczai, and Daniel Weller. Introducing Quantified Cuts in Logic with Equality. In Stéphane Demri, Deepak Kapur, and Christoph Weidenbach, editors, *Proceedings of the 7th International Joint Conference on Automated Reasoning (IJCAR 2014)*, volume 8562 of *Lecture Notes in Computer Science*, pages 240–254, Cham, 2014. Springer.
- [HLRW14] Stefan Hetzl, Alexander Leitsch, Giselle Reis, and Daniel Weller. Algorithmic introduction of quantified cuts. *Theoretical Computer Science*, 549:1–16, 2014.

- [HLW12] Stefan Hetzl, Alexander Leitsch, and Daniel Weller. Towards Algorithmic Cut-Introduction. In Nikolaj Bjørner and Andrei Voronkov, editors, *Proceedings of the 18th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR 2012)*, volume 7180 of *Lecture Notes in Computer Science*, pages 228–242, Berlin, Heidelberg, 2012. Springer.
- [HMU01] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, second edition, 2001.
- [HS07] Yo-Sub Han and Kai Salomaa. State complexity of union and intersection of finite languages. In Tero Harju, Juhani Karhumäki, and Arto Lepistö, editors, *Proceedings of the 11th International Conference on Developments in Language Theory (DLT 2007)*, volume 4588 of *Lecture Notes in Computer Science*, pages 217–228, Berlin, Heidelberg, 2007. Springer.
- [HU69] John E. Hopcroft and Jeffrey D. Ullman. *Formal Languages and their Relation to Automata*. Addison-Wesley, 1969.
- [HU79] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.
- [Hun73] Harry B. Hunt, III. *On the Time and Tape Complexity of Languages*. PhD thesis, Cornell University, 1973.
- [HW18a] Stefan Hetzl and Simon Wolfsteiner. Cover Complexity of Finite Languages. In Stavros Konstantinidis and Giovanni Pighizzini, editors, *Proceedings of the 20th IFIP WG 1.02 International Conference on Descriptive Complexity of Formal Systems (DCFS 2018)*, volume 10952 of *Lecture Notes in Computer Science*, pages 139–150, Cham, 2018. Springer.
- [HW18b] Markus Holzer and Simon Wolfsteiner. On the Grammatical Complexity of Finite Languages. In Stavros Konstantinidis and Giovanni Pighizzini, editors, *Proceedings of the 20th IFIP WG 1.02 International Conference on Descriptive Complexity of Formal Systems (DCFS 2018)*, volume 10952 of *Lecture Notes in Computer Science*, pages 151–162, Cham, 2018. Springer.
- [HW19] Stefan Hetzl and Simon Wolfsteiner. On the cover complexity of finite languages. *Theoretical Computer Science*, 798:109–125, 2019.
- [IP99] Russell Impagliazzo and Ramamohan Paturi. Complexity of k-sat. In *Proceedings of the Fourteenth Annual IEEE Conference on Computational Complexity (COCO 1999)*, pages 237–240, Washington, 1999. IEEE.
- [Ipa12] Florentin Ipate. Learning finite cover automata from queries. *Journal of Computer and System Sciences*, 78(1):221–244, 2012.

- [IPZ01] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which Problems Have Strongly Exponential Complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.
- [Jir05] Galina Jirásková. State complexity of some operations on binary regular languages. *Theoretical Computer Science*, 330(2):287–298, 2005.
- [JJS05] Jozef Jirásek, Galina Jirásková, and Alexander Szabari. State Complexity of Concatenation and Complementation. *International Journal of Foundations of Computer Science*, 16(3):511–529, 2005.
- [JM11] Artur Jež and Andreas Maletti. Computing All ℓ -Cover Automata Fast. In Béatrice Bouchou-Markhoff, Pascal Caron, Jean-Marc Champarnaud, and Denis Maurel, editors, *Proceedings of the 16th International Conference on Implementation and Application of Automata (CIAA 2011)*, volume 6807 of *Lecture Notes in Computer Science*, pages 203–214, Berlin, Heidelberg, 2011. Springer.
- [Joh09] Neil F. Johnson. *Simply Complexity: A Clear Guide to Complexity Theory*. Oneworld Publications, 2009.
- [KF90] Jānis Kaneps and Rūsinš Freivalds. Minimal nontrivial space complexity of probabilistic one-way turing machines. In Branislav Rován, editor, *Proceedings of the 15th International Symposium on Mathematical Foundations of Computer Science (MFCS 1990)*, volume 452 of *Lecture Notes in Computer Science*, pages 355–361, Berlin, Heidelberg, 1990. Springer.
- [Kör03a] Heiko Körner. A Time and Space Efficient Algorithm for Minimizing Cover Automata for Finite Languages. *International Journal of Foundations of Computer Science*, 14(6):1071–1086, 2003.
- [Kör03b] Heiko Körner. On Minimizing Cover Automata for Finite Languages in $O(n \log n)$ Time. In Jean-Marc Champarnaud and Denis Maurel, editors, *Proceedings of the 7th International Conference on Implementation and Application of Automata (CIAA 2002)*, volume 2608 of *Lecture Notes in Computer Science*, pages 117–127, Berlin, Heidelberg, 2003. Springer.
- [KY00] John C. Kieffer and En-Hui Yang. Grammar-Based Codes: A New Class of Universal Lossless Source Codes. *IEEE Transactions on Information Theory*, 46(3):737–754, 2000.
- [KYNC00] John C. Kieffer, En-Hui Yang, Greg J. Nelson, and Pamela Cosman. Universal Lossless Compression Via Multilevel Pattern Matching. *IEEE Transactions on Information Theory*, 46(4):1227–1245, 2000.
- [LL09] Martin Lange and Hans Leiß. To CNF or not to CNF? An Efficient Yet Presentable Version of the CYK Algorithm. *informatica didactica*, 8, 2009.

- [LMS11] Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Lower bounds based on the Exponential Time Hypothesis. *Bulletin of the European Association for Theoretical Computer Science*, 105:41–71, 2011.
- [LV08] Ming Li and Paul Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer, 2008.
- [LZ76] Abraham Lempel and Jacob Ziv. On the Complexity of Finite Sequences. *IEEE Transactions on Information Theory*, 22(1):75–81, 1976.
- [NM96] Craig G. Nevill-Manning. *Inferring Sequential Structure*. PhD thesis, University of Waikato, 1996.
- [NMW97] Craig G. Nevill-Manning and Ian H. Witten. Identifying Hierarchical Structure in Sequences: A linear-time algorithm. *Journal of Artificial Intelligence Research*, 7:67–82, 1997.
- [Pap95] Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1995.
- [PS18] Aaron Potechin and Jeffrey O. Shallit. Lengths of Words Accepted by Nondeterministic Finite Automata. *CoRR*, abs/1802.04708, 2018.
- [PSY01] Andrei Păun, Nicolae Sântean, and Sheng Yu. An $O(n^2)$ algorithm for constructing minimal cover automata for finite languages. In Sheng Yu and Andrei Păun, editors, *Proceedings of the 5th International Conference on Implementation and Application of Automata (CIAA 2000)*, volume 2088 of *Lecture Notes in Computer Science*, pages 243–251, Berlin, Heidelberg, 2001. Springer.
- [Rév91] György E. Révész. *Introduction to Formal Languages*. Courier Corporation, 1991.
- [SB96] Jeffrey O. Shallit and Yuri Breitbart. Automaticity I: Properties of a Measure of Descriptive Complexity. *Journal of Computer and System Sciences*, 53(1):10–25, 1996.
- [SG17] Payam Siyari and Matthias Gallé. The Generalized Smallest Grammar Problem. In Sicco Verwer, Menno van Zaanen, and Rick Smetsers, editors, *Proceedings of the 13th International Conference on Grammatical Inference (ICGI 2016)*, volume 57 of *Proceedings of Machine Learning Research*, pages 79–92. PMLR, 2017.
- [Sha08] Jeffrey O. Shallit. *A Second Course in Formal Languages and Automata Theory*. Cambridge University Press, 2008.
- [Sip13] Michael Sipser. *Introduction to the Theory of Computation*. Cengage Learning, 2013.

- [SS82] James A. Storer and Thomas G. Szymanski. Data Compression via Textual Substitution. *Journal of the ACM*, 29(4):928–951, 1982.
- [Tuz87] Zsolt Tuza. On the context-free production complexity of finite languages. *Discrete Applied Mathematics*, 18(3):293–304, 1987.
- [VW18] Virginia Vassilevska Williams. On Some Fine-Grained Questions in Algorithms and Complexity. In Boyan Sirakov, Paulo Ney de Souza, and Marcelo Viana, editors, *Proceedings of the International Congress of Mathematicians (ICM 2018)*, pages 3447–3487, Singapore, 2018. World Scientific.
- [Weh16] Michael Wehar. *On the Complexity of Intersection Non-Emptiness Problems*. PhD thesis, University at Buffalo, 2016.
- [Woe08] Gerhard J. Woeginger. Open problems around exact algorithms. *Discrete Applied Mathematics*, 156(3):397–405, 2008.
- [Woo87] Derick Wood. *Theory of Computation*. John Wiley & Sons, 1987.
- [WS11] David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011.
- [YK00] En-Hui Yang and John C. Kieffer. 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.
- [Yu01] Sheng Yu. State Complexity of Regular Languages. *Journal of Automata, Languages and Combinatorics*, 6(2):221–234, 2001.
- [Yu07] Sheng Yu. Cover Automata For Finite Languages. *Bulletin of the European Association for Theoretical Computer Science*, 92:65–74, 2007.
- [YZS94] Sheng Yu, Qingyu Zhuang, and Kai Salomaa. The state complexities of some basic operations on regular languages. *Theoretical Computer Science*, 125(2):315–328, 1994.
- [ZL77] Jacob Ziv and Abraham Lempel. A Universal Algorithm for Sequential Data Compression. *IEEE Transactions on Information Theory*, 23(3):337–343, 1977.
- [ZL78] Jacob Ziv and Abraham Lempel. Compression of Individual Sequences via Variable-Rate Coding. *IEEE Transactions on Information Theory*, 24(5):530–536, 1978.