Modern Aspects of Complexity Within Formal Languages

Henning Fernau(✉)

Fachbereich 4 – Abteilung Informatikwissenschaften, CIRT, Universität Trier, 54286 Trier, Germany
fernau@uni-trier.de

Abstract. We give a survey on some recent developments and achievements of modern complexity-theoretic investigations of questions in Formal Languages (FL). We will put a certain focus on multivariate complexity analysis, because this seems to be particularly suited for questions concerning typical questions in FL.

Keywords: String problems · Finite automata · Context-free grammars · Multivariate analysis · Fixed-parameter tractability · Fine-grained complexity

1 Introduction

Formal Languages and Complexity Theory have a long (common) history. Both fields can be seen as two of the major backbones of Theoretical Computer Science. Both fields are often taught together in undergraduate courses, mostly obligatory in Computer Science curricula. This is also testified by looking at classical textbooks like [50]. Yet, modern developments in complexity and algorithmics are barely mirrored in Formal Languages. We want to argue in this paper that this is a fact that need to be changed.

We will work through six case studies to explain several findings in recent years. We will also expose a number of open problems in each of these cases. This should motivate researchers working in Formal Languages to look deeper into recent developments in Complexity Theory, but also researchers more oriented towards these modern aspects of Complexity Theory to look into (possibly apparently rather old) problems in Formal Languages to see if they could offer solutions or at least new approaches to these problems.

In most cases, the problems we discuss in the area of Formal Languages can be easily understood with the already mentioned background knowledge each computer scientists gets already during the corresponding bachelor courses. Therefore, we will only fix notations but assume that no further explanations are necessary. By way of contrast, we will spend some more time explaining at least some ideas of the concepts discussed nowadays in Complexity Theory and Algorithms, so that readers with a background rooted in Formal Languages

C. Martín-Vide et al. (Eds.): LATA 2019, LNCS 11417, pp. 3–30, 2019.
https://doi.org/10.1007/978-3-030-13435-8_1

could easily follow this introductory exposition. In any case, we want to make clear why these modern approaches are particularly suited for attacking decision problems in Formal Languages.

2 Some Modern Concepts of Complexity Theory

As most of our case studies deal with multivariate analysis, let us first delve into the general scheme behind this idea. This is intimately linked to the basic ideas of Parameterized Complexity, which could be paraphrased as follows. If we can show that some decision problem is computationally complicated, which is typically formalized by proving that it is NP-hard or PSPACE-hard, what can we do about it then, assuming that we still like to solve it with the help of computers? More traditionally, help was expected from approximation algorithms or, more practically, from heuristics, and in fact the proof of computational hardness often gave a sort of excuse of using these rules of thumb called heuristics. However, in particular using heuristics with no success guarantees is not very satisfying from a more theoretical perspective at least. What can be done about such a situation?

It is exactly here where Parameterized Complexity and also the more practical side of it, namely Parameterized Algorithms, steps in. The basic idea is to not merely look at the bitstring length as the only source of information that an instance of a computational problem might give as defining or measuring its complexity, but to also look at other aspects of the instance, formalized in terms of a so-called parameter. What happens if they are always small in the concrete instances that practitioners look at? Does this still mean that the problem is computationally hard? Or can we possibly solve these practically relevant instances efficiently, although the problem is NP-hard when considering all possible instances? Interestingly, one of the most successful approaches within Parameterized Algorithms can be viewed as an analysis of natural heuristics, which means, in more formal terms, the use of reduction rules to provide means to (repeatedly) move from an instance to an equivalent instance of smaller size.

What could these parameter be? Let us look at various examples.

- The most classical NP-hard problem is arguably Satisfiability, or SAT for short. Given a Boolean formula φ in conjunctive normal form (CNF), decide if φ is satisfiable. A possibly natural choice of a parameter could be a bound k on the number of literals that may appear in clauses. When fixing this upper-bound to k, we arrive at problems like k-SAT. This parameter choice is not that helpful, because it is well-known that even 3-SAT remains NP-hard, and in fact there are quite a number of much more restricted variants of SAT that cannot be solved in polynomial time, assuming that P does not equal NP, see [57,61,93] for several such restrictions. Only 2-SAT is still solvable in polynomial time.
- Looking back at the proof of the theorem of Cook, phrased in (nowadays) non-standard terminology in [21], one can argue that the more basic NP-complete problem is in fact the following one, which can be considered as a

problem arising in the field of Formal Languages: Given a nondeterministic Turing machine M, an input string x and an integer k, does M halt within k steps accepting x? Now, if k is fixed to a small constant, this problem looks more amenable than the previous one. More precisely, let us focus on one-tape machines for now. Assume that we have ℓ symbols in the work alphabet of M and assume that M has t states. Then at each time step, being in a specific state, upon reading a symbol, M has at most $3\ell t$ choices to continue its computation, the factor three entering because of the different choices for moves. Moreover, after k steps, at most $\mathcal{O}(tk\ell^k)$ many configurations are possible, simply because at most k tape cells could have been visited within k steps, and also the current state and head position has to be memorized. The difference to Turing machines with a dedicated read-only input tape is not of importance here. It can be easily checked if an accepting configuration can be reached from the initial configuration within the directed graph implicitly given by the configuration and the reachability relation. Instead of writing ℓ and t, we can also consider the size of M, with reasonable ways to measure this; the size of the state transition table should be always encaptured here, and this also bounds the number of choices. So, we might write the running time of this algorithm like $\mathcal{O}(|M|^k)$, which is polynomial when k is really fixed, but if k is considered as part of the input, this algorithm is clearly exponential. Similar considerations are valid for multi-tape nondeterministic Turing machines. For future reference, let us call these problems Short NTM Acceptance (when referring to single tapes) and Short Multi-Tape NTM Acceptance.

– Recall that a hypergraph can be specified as $G = (V, E)$ with $E \subseteq 2^V$. Elements of V are called vertices, while elements of E are called (hyper-)edges. In different terminology, V is the ground-set or universe, and E is a set system. $C \subseteq V$ is called a *hitting set* if $e \cap C \neq \emptyset$ for all $e \in E$. In the decision problem Hitting Set, we are given a hypergraph $G = (V, E)$ and an integer k, and the question is if one can find a hitting set of cardinality at most k for G. This question is well-known to be NP-hard. There is a simple relation to SAT: One can view the hyperedges as clauses. Now, a hitting set corresponds to the set of variables set to true. It is also evident why we need the bound k here: setting all variables to true corresponds to selecting V as a hitting set, and this is clearly a satisfying assignment, because none of the clauses contains any negation. This relation also motivates to study d-Hitting Set, where d upper-bounds the number of vertices in each hyperedge. In contrast to SAT, this restriction looks quite promising, considering the following algorithm: As long as there are hyperedges in G and as long as the integer k is positive, do the following: pick such a hyperedge e and branch according to the at most d possibilities for hitting e. In each case, delete all hyperedges from G that contain e, decrement k and continue. If and only if this loop is ever exited with $E = \emptyset$, the original instance was a YES-instance. As at most d^k many possibilities are tested, the running time of this algorithm can be estimated as $\mathcal{O}(d^k p(|G|))$, where p is some polynomial and $|G|$ gives the size of the original

input. Without this additional accounting of d, it is not clear how to solve this problem better than $\mathcal{O}(|G|^k)$.
– Observe that hypergraph instances of 2-Hitting Set can be also viewed as undirected simple graphs, because loops (i.e., singletons in the set of hyperedges) can be easily removed, as the constituent elements must be put into the hitting set. 2-Hitting Set is also known as Vertex Cover. Then, an alternative parameterization might be $k_d = |V| - k$. This problem is also known as Independent Set, and it can be rephrased as asking for a set I of k_d many vertices such that no edge contains two vertices from I. Such a set is also known as an *independent set*. A related question asks, given a graph G and an integer ℓ, if there is a *clique* of size ℓ in G, i.e., a set K of ℓ vertices such that each pair of vertices from K is adjacent. All these decision problems are also known to be NP-complete.

With these problems from different fields in mind, it might make sense to consider a decision problem $\mathcal{P}$ equipped with a computable parameterization function κ that maps instances to integers. Two ways in which algorithms can behave nicely on instances x might be considered. (a) There is an algorithm that solves instances x of $\mathcal{P}$ in time $\mathcal{O}(|x|^{\kappa(x)})$. (b) There is an algorithm that solves instances x of $\mathcal{P}$ in time $\mathcal{O}(|x|^d f(\kappa(x)))$ for some constant degree d and some (computable) function f. Both definitions imply that $\mathcal{P}$ can be solved in polynomial time, when restricted to instances whose parameter is smaller than some constant c. Yet, possibility (b) means that, assuming d to be reasonably small and f not behaving too badly, then $\mathcal{P}$ can be solved not only for instances where the parameter $\kappa(x)$ is bounded by a constant but it may grow moderately, i.e., this is a far more desirable property. (Parameterized) problems $(\mathcal{P}, \kappa)$ that satisfy (a) are also said to belong to XP, while if $(\mathcal{P}, \kappa)$ satisfies (b), it belongs to FPT (fixed parameter tractable).

Let us look at our example problems again.

– Let κ_1 map a Boolean formula φ to the number of variables that occur in φ. Then, (SAT, κ_1) is in FPT, because there are only $2^{\kappa_1(\varphi)}$ many assignments one has to check to see if φ is satisfiable. Let κ_2 map a Boolean formula φ in CNF to the maximum number of literals appearing in any clause of φ. Then, unless P equals NP, (SAT, κ_2) does not belong to XP.
– For an instance $I = (M, x, k)$ of Short NTM Acceptance, let $\kappa_3(I) = k$. As argued above, (Short NTM Acceptance, κ_3) belongs to XP. However, it seems to be hard to put it in FPT. If we consider the size of M as an additional parameter, i.e., $\kappa_4(I) = |M| + k$, then we have also seen that (Short NTM Acceptance, κ_4) belongs to FPT. Similar considerations hold true for the multi-tape case. Possibly more interestingly, our considerations show that also with the parameterization κ_5 that adds k and the size of the overall alphabet, we end up in FPT. A reader knowledgeable about the early days of Descriptional Complexity (within FL) might ponder for a moment if there might be a possibility to put (Short NTM Acceptance, κ_3) into FPT by resorting to a theorem of Shannon [84] that states that Turing machines with an arbitrary number of working tape symbols can be simulated by Turing

machines with binary tapes. However, this idea is problematic at least for two reasons: (a) the simulating machine (with binary working tape) needs a considerable amount of time for the simulation, this way changing the upper-bound k on the number of steps; (b) in Shannon's simulation, the order of magnitude of the product of alphabet size and number of states stays the same. In combination, both effects counter-act this idea.

- Let us study various parameterizations for Hitting Set. Let $G = (V, E)$ with $E \subseteq 2^V$ and k form an instance and define $\kappa_6 : (G, k) \mapsto k$, $\kappa_7 : (G, k) \mapsto |V|$, $\kappa_8 : (G, k) \mapsto |E|$ and $\kappa_9 : (G, k) \mapsto \max\{|e| : e \in E\}$. As $\binom{n}{k} \in \mathcal{O}(n^k)$, in roughly $\mathcal{O}(|V|^k)$ steps, the instance (G, k) can be solved by testing all k-element subsets if they form a hitting set, putting (Hitting Set, κ_6) in XP. With the same idea, (Hitting Set, κ_7) is in FPT. By using dynamic programming as explained in [30,41], also (Hitting Set, κ_8) is in FPT. Due to the NP-hardness of Vertex Cover, if (Hitting Set, κ_9) is in XP, then P equals NP. When combining parameters, $\kappa_{10} := \kappa_6 + \kappa_9$, we conclude with the reasoning given above that (Hitting Set, κ_{10}) is in FPT. This is quite instructive, because it shows that even combining relatively weak parameters, one can obtain relatively nice algorithmic results. More details, also for special cases, can be found in [30–32,99].
- Reconsider $\kappa_6 : (G, k) \mapsto k$ for Vertex Cover. Then, by the equivalence to 2-Hitting Set, (Vertex Cover, κ_6) belongs to FPT. However, reparameterizing by $\kappa_{11} : (G, k) \mapsto |V| - k$, we only know membership in XP for (Vertex Cover, κ_{11}). Recall that this is equivalent to considering (Independent Set, κ_6). By moving over from G to the graph complement $\overline{G}$, with $\overline{(V, E)} = (V, \binom{V}{2} \setminus E)$, one understands that also (Clique, κ_6) has the same complexity status as (Vertex Cover, κ_{11}).

So far, we introduced the classes FPT and XP of parameterized problems. Clearly, FPT $\subseteq$ XP. As often in Complexity Theory, it is unknown if this inclusion is strict, but it is generally assumed that this is the case. Moreover, we have seen examples of parameterized problems that are not in XP, assuming that P is not equal to NP. In order to have a more refined picture of the world of parameterized problems, it is a good idea to define appropriate reductions. It should be clear what properties such a *many-one* FPT*reduction* relating problem $(\mathcal{P}, \kappa)$ to $(\mathcal{P}', \kappa')$ should satisfy: (a) it should translate an instance I of $\mathcal{P}$ to an instance I' of $\mathcal{P}'$ within time $f(\kappa(I))|I|^{\mathcal{O}(1)}$ for some computable function f; (b) it should preserve the parameterization in the sense there is a function g such that $\kappa'(I') \leq g(\kappa(I))$; (c) I is a YES-instance of $\mathcal{P}$ if and only if I' is a YES-instance of $\mathcal{P}'$. Such a notion allows us to define further complexity classes, based on the idea of being interreducibility with respect to FPT reductions, or FPT-*equivalent*. Observe that the classes FPT and XP studied so far are closed under FPT reductions. Let W[1] be the class of parameterized problems that are FPT-equivalent to (Short NTM Acceptance, κ_3), and let W[2] be the class of parameterized problems that are FPT-equivalent to (Short Multitape NTM Acceptance, κ_3). In fact, there is a whole (presumably infinite) hierarchy of

complexity classes captured in the following inclusion chain:

$$\mathsf{FPT} \subseteq \mathsf{W}[1] \subseteq \mathsf{W}[2] \subseteq \cdots \subseteq \mathsf{XP}.$$

Looking back at our examples, it is known that (Independent Set, κ_6) and (Clique, κ_6) are complete for W[1], while (Hitting Set, κ_6) is complete for W[2]. As we will see in the following sections, many natural parameterizations of computational problems stemming from Formal Languages lead to W[1]-hardness results. If we still want to employ the idea of getting FPT algorithms, we need to find different, often multiple parameterizations. This approach is also called *mutlivariate analysis*. We refer to [12,28,71] for further discussions. At this point, we only recall that we also used this idea when looking at (Hitting Set, κ_{10}).

There is a nice characterization of FPT based on the idea of the existence of a polynomial-time many-one reduction termed *kernelization* that maps instances I of $(\mathcal{P}, \kappa)$ to instances I', also of $(\mathcal{P}, \kappa)$, satisfying $|I'| \leq f(\kappa(I))$ for some computable function f, where $|I|$ yields the size of instances I in the classical sense.[1] I' is also called the *kernel* of I. The existence of kernelizations is often shown by providing a collection of so-called reduction rules that should be applied exhaustively in order to produce the kernel. As an example, the two reduction rules together provide a kernelization for (Vertex Cover, κ_6). (a) Delete vertices of degree zero, or, more formally, $((V,E),k) \mapsto ((V \setminus \{v \mid v \notin \bigcup_{e \in E} e\}),k)$, and (b) $(G,k) \mapsto (G-v, k-1)$ if there is some $v \in V(G)$ with more than k neighbors. In fact, it is not hard to see that the resulting kernels (G',k') even satisfy a polynomial bound on the size of G' (with some reasonable size measure) with respect to $\kappa_6(G',k') = k'$. We also say that (Vertex Cover, κ_6) admits a polynomial kernel.

With this notion at hand, the question is if one can always produce kernels of polynomial size for parameterized problems in FPT. This is not the case unless the polynomial-time hierarchy collapses to the third level, which is considered to be unlikely. For instance, under this condition, it can be shown that both (Hitting Set, κ_7) and (Hitting Set, κ_8) and hence also (SAT, κ_1) have no polynomial-size kernels.

Another venue that one could follow is refining the questions about optimality of existing algorithms further beyond the question if the decision problem at hand belongs to P or if it is NP-hard. This line of research is nowadays captured under the umbrella of *Fine-Grained Complexity*.

For instance, as discussed, there is a trivial algorithm to solve a SAT instance by testing all assignments. Neglecting polynomial factors, as standard by the very definition of FPT, we can also state that SAT instances on n variables can be solved in time $O^*(2^n)$, where the O^*-notation was invented just to suppress polynomial factors. Now, one can ask if there is any algorithm that solves SAT instances in time $O^*((2-\varepsilon)^n)$ for any $\varepsilon > 0$. No such algorithm is known today.

[1] In the literature, is often required that $|I'| + \kappa(I') \leq f(\kappa(I))$, but this is equivalent to the present requirement, because the parameterization can be computed from I', i.e., $\kappa(I')$ is also bounded by a function in $\kappa(I)$ if $|I'|$ is.

The hypothesis that no such algorithm exists is also called *Strong Exponential-Time Hypothesis*, or SETH for short. A weaker assumption is to believe that there is no function $f(n) \in o(n)$ such that SAT, or in this case equivalently also 3-SAT, can be solved in time $O^*(2^{f(n)})$. This hypothesis is also known as *Exponential-Time Hypothesis*, or ETH for short.[2] The outcome of the famous sparsification lemma is sometimes good to know: Under ETH, there is also no $O^*(2^{o(n)})$-time algorithm for 3-SAT on instances that have $\mathcal{O}(n)$ many clauses. For instance, it is known that under ETH, no $2^{o(n)}$ algorithm exists for solving Hitting Set on instances with n vertices. We also refer to [22]. Further consequences of this approach to the (non-)existence of certain types of FPT algorithms are also discussed in the survey paper [62]. For instance, while there are quite a number of algorithms that solve (Vertex Cover, κ_6) in time $O^*(2^{\mathcal{O}(\kappa_6(G))})$, under ETH there is no algorithm for doing this in time $O^*(2^{o(\kappa_6(G))})$. It should be clear that for obtaining such results, another form of reduction is needed. We do not explain any details here, but just observe that many well-known reductions suffice for showing some basic ETH-based results. For instance, the typical textbook reductions for showing NP-hardness of Vertex Cover start from 3-SAT and then introduce gadgets with two or three vertices for variables or clauses, respectively. Hence, by the outcome of the sparsification lemma, under ETH there is no $O^*(2^{o(|V|)})$-time algorithm for computing a minimum vertex cover for $G = (V, E)$. However, not all lower bounds of this type that one assumes to hold can be shown to be rooted in ETH. For instance, the *Set Cover Conjecture* claims (using the vocabulary of this paper) that (Hitting Set, κ_7) cannot be solved in time $O^*(2^{o(\kappa_7(G))})$. As discussed in [22], it is not clear how this (plausible) conjecture relates to ETH.

Finally, one might wonder how to attack XP-problems, trying to understand how the parameter(s) influence the running time. For instance, consider the problem of finding a clique of size k in a graph with n vertices. Using brute-force, this problem can be solved in time $O^*(\binom{n}{k}) = \mathcal{O}(n^{k+c})$ for some small constant c. Nešetřil and Poljak showed in [66] that this can be improved to $\mathcal{O}(n^{k\omega/3+c})$, where ω is the exponent such that $n \times n$ matrices can be multiplied in time $\mathcal{O}(n^{\omega})$. The underlying idea is that triangles can be found by multiplying adjacency matrices. Nowadays, it is believed that this is indeed the correct bound for detecting k-cliques. The hypothesis that no better algorithms are possible than those intimately linked to matrix multiplication is known as *k-Clique Conjecture*. This is one of the various examples of conjectures within the realm of polynomial-time algorithms on which several hardness assertions are based. In this context, it is also worth mentioning that there is a common belief that $\omega > 2$; this also links to interesting combinatorial conjectures as exhibited in [5]. We will re-encounter this conjecture in our last case study. Virginia Vassilevska Williams wrote several surveys on Fine-Grained Complexity, the most recent published one being [104].[3] When dealing with distinguishing problems within P, also adapted

[2] The definitions in [52] are a bit different, but this can be neglected in the current discussion.

[3] A new survey is announced to appear in [85].

notions of reductions have to be introduced. For the sake of space and because this is not that central to this paper, we are not going to present them here but refer to the mentioned survey papers. Also due to the nature of these reductions, in this part of the complexity world, polylogarithmic factors in the running time are often ignored, leading to notations like $\tilde{O}(n^2)$ for denoting quadratic running times up to terms like $(\log(n))^3$. Vassilevska Williams put the central question of Fine-Grained Complexity as follows in the survey to appear in [85]: *For each of the problems of interest with textbook running time $\mathcal{O}(t(n))$ and nothing much better known, is there a barrier to obtaining an $\mathcal{O}(t(n)^{1-\varepsilon})$ time algorithm for $\varepsilon > 0$?* Notice this formulation ignores polylogarithmic factors. Also, SETH perfectly fits into this line of questions.

Many more details can be found in the textbooks that have appeared in the meantime in the context of Parameterized Complexity, often also capturing aspects of ETH and also of SETH. We refer to [23–25,40,42,70].

In the following, we present six case studies, focusing on typical effects that show up when dealing with computational problems in Formal Languages. We start with a problem dealing with strings only, continuing with problems involving grammars and automata. There are some common themes throughout all these studies, for instance, the (sometimes surprising) role played by the size of the alphabet concerning the complexity status of the problems. Another recurring theme is that rather simple algorithms can be shown to be optimal.

3 First Case Study: String-to-String Correction S2S

The *edit distance* is a measure introduced to tell the distance between two strings $S, T \in \Sigma^*$, where S is the source string and T is the target string, by counting the number of elementary edit operations that are necessary to turn S into T. The complexity of this problem depends on the permitted operations. Let O be the set of permitted operations, $O \subseteq \{\mathbf{C}, \mathbf{D}, \mathbf{I}, \mathbf{S}\}$, with:

C Change: replace/substitute a letter
D Delete a letter
I Insert a letter
S Swap: transpose neighboring letters

For each O, define the problem O-String-to-String Correction, or O-S2S for short, with input Σ, $S, T \in \Sigma^*$, $k \in \mathbb{N}$, to be the following question: Is it possible to turn S into T with a sequence of at most k operations from O?

Wagner [97] obtained a by now classical dichotomy result that can be stated as follows.[4]

Theorem 1. *Consider $O \subseteq \{\boldsymbol{C}, \boldsymbol{D}, \boldsymbol{I}, \boldsymbol{S}\}$.*

– *If $O \in \{\{\boldsymbol{S}, \boldsymbol{D}\}, \{\boldsymbol{S}, \boldsymbol{I}\}\}$, then O-S2S is* NP*-complete.*

[4] The result was phrased in different terminology back in 1975. Wagner actually proved stronger results in the sense that weights on the operations are permitted.

– *If* $O \notin \{\{\boldsymbol{S}, \boldsymbol{D}\}, \{\boldsymbol{S}, \boldsymbol{I}\}\}$*, then* O*-S2S is solvable in polynomial time.*

Note: $\{\mathbf{S}, \mathbf{D}\}$-S2S is equivalent to $\{\mathbf{S}, \mathbf{I}\}$-S2S.

How is this dichotomy result obtained? What is the *source of* NP*-hardness*? Conversely, how do the algorithms work? Here, dynamic programming is the key, and the corresponding algorithms (or variants thereof) have made their way into textbooks on algorithms.

Let us first study the NP-hard variant, focusing on $\{\mathbf{S}, \mathbf{D}\}$-S2S. What are natural parameters of an instance I defined by Σ, $S, T \in \Sigma^*$, $k \in \mathbb{N}$? From the viewpoint of now traditional parameterized complexity, $\kappa_1(I) = k$ is a first pick. This has been considered in [3]. Its main result can be stated as follows.

Theorem 2. $\{\boldsymbol{S}, \boldsymbol{D}\}$*-S2S with parameter* $\kappa_1(\Sigma, S, T \in \Sigma^*, k) = k$ *is in* FPT*. More precisely, an instance* $I = (\Sigma, S, T \in \Sigma^*, k)$ *can be solved in time* $\mathcal{O}(\varphi^k(|S|)\log(|\Sigma|))$*, where* $\varphi < 1.62$ *is the golden ratio number.*

In addition, a polynomial kernel was obtained in [101]. One of the important observations is that we can assume to always execute $k_1 = |S| - |T|$ deletions prior to swaps. Moreover, the at most $k - k_1$ swaps can be described by one position in the string. Hence, k is upper-bounded by a function in $|S|$ and we can use the previously mentioned algorithm to prove:

Proposition 3. $\{\boldsymbol{S}, \boldsymbol{D}\}$*-S2S with parameter* $\kappa_2(\Sigma, S, T, k) = |S|$ *is in* FPT*.*

With quite a similar reasoning, one can obtain the next result.

Proposition 4. $\{\boldsymbol{S}, \boldsymbol{D}\}$*-S2S with parameter* $\kappa_3(\Sigma, S, T, k) = |T|$ *is in* FPT*.*

The previous two results seem to be a bit boring, because $|S|$ and $|T|$ appear to be the natural choice to describe the overall size of the input. Also, these string lengths would be rather big in practice, while one could assume k to be rather small if one compares strings that are somehow similar to each other.

There is one last choice of a parameter that one might tend to overlook when first being confronted with this problem, namely, the size of the alphabet over which the strings S and T are formed. Yet, studying the proof of NP-hardness of Wagner, one is led to the conclusion that $|\Sigma|$ is crucial for this proof, which simply does not work if $|\Sigma|$ is bounded, for instance, if we consider binary alphabets only. This might look a bit surprising, as it might be hard to imagine that the alphabet size itself could carry such an importance, given the fact that the alphabet carries no visible or obvious structure. Yet, the consideration of the alphabet size will be a recurring theme in this paper, and we will see various situations where this is in fact a crucial parameter. This problem was first resolved in [36] for binary alphabets, showing the following result.

Proposition 5. $\{\boldsymbol{S}, \boldsymbol{D}\}$*-S2S on binary alphabets can be solved in cubic time.*

This was soon superseded by a more general result by Meister [64], showing that indeed the size of the alphabet was the crucial source of hardness for this problem. This was later improved by Barbay and Pérez-Lantero [10] concerning the dependence of the degree of the polynomial describing the running time on the alphabet size parameter.

Theorem 6. *$\{\boldsymbol{S}, \boldsymbol{D}\}$-S2S with parameter $\kappa_4(\Sigma, S, T, k) = |\Sigma|$ is in* XP. *More precisely, it can be solved in time $\mathcal{O}(|S|^{|\Sigma|+1})$.*

Still, this result is likely only practical only for very small alphabet sizes. Also, it is still open if one can put the problem in FPT. From a more practical perspective, as the parameters k and Σ are rather unrelated and also because the algorithmic approaches leading to Theorems 2 and 6 are quite different, it would be interesting to see if one can combine both ideas to produce an algorithm that is really useful for instances with a small alphabet size and a moderate number of permitted edit operations. Of course, this should be also checked by computational experiments that seem to be lacking so far.

It would also be interesting to study further parameters for this problem. In [10], several such suggestions have been considered. For instance, Barbay and Pérez-Lantero [10] have shown the following consequence for their algorithm. This relates to the number of deletions ($|T| - |S|$) already studied above.

Proposition 7. *$\{\boldsymbol{S}, \boldsymbol{D}\}$-S2S with parameter $\kappa_5(\Sigma, S, T, k) = |\Sigma| + (|T| - |S|)$ is in* FPT. *More precisely, it can be solved in time $\mathcal{O}^*((|T| - |S|)^{|\Sigma|})$.*

Let us also discuss some fine-grained complexity results for $\{\mathbf{C}, \mathbf{D}, \mathbf{I}\}$-S2S. This problem is also known as computing the edit distance (in a more restricted sense) or as computing the Levenshtein-distance between two strings. The already mentioned textbook algorithms, often based on [98], take quadratic time. Whether or not these are optimal was actually a question already investigated 40 years ago. In those days, however, lower bound results were dependent on particular models of computation, while more modern approaches to lower bounds are independent of such a choice. For instance, Wong and Chandra [106] showed a quadratic lower bound assuming that only equality tests of single symbols are permitted as the basic operation of comparison. Masek and Paterson [63] managed to shave off a logarithmic factor by making some clever use of matrix multiplication tricks. Yet, whether essentially better algorithms are possible remained an open question up to recently. Backurs and Indyk [8] finally proved that assuming SETH, no $\mathcal{O}(n^{2-\varepsilon})$-time algorithm can be expected for any $\varepsilon > 0$. Interestingly enough, their proof was working only for alphabet sizes at least seven. This result has then been improved by Bringmann and Künnemann [14] to binary alphabets. Let us summarized these results in the following statement.

Theorem 8. *$\{\boldsymbol{S}, \boldsymbol{D}, \boldsymbol{I}\}$-S2S can be solved in quadratic time. Assuming SETH, no $\mathcal{O}(|S|^{2-\varepsilon})$-time algorithm exists even on binary alphabets, for any $\varepsilon > 0$.*

Suggestions for Further Studies. (a) Does $\{\mathbf{S}, \mathbf{D}\}$-S2S with parameterization $\kappa_4(\Sigma, S, T, k) = |\Sigma|$ belong to FPT, or is it hard or even complete for some level of the W-hierarchy? (b) Assuming that $\{\mathbf{S}, \mathbf{D}\}$-S2S with parameter $\kappa_4(\Sigma, S, T, k) = |\Sigma|$ belongs to FPT, is it possible to give, e.g., SETH-based lower bounds for showing that existing algorithms are (close to) optimal? (c) Assuming that existing algorithms for $\{\mathbf{S}, \mathbf{D}\}$-S2S with parameter $\kappa_4(\Sigma, S, T, k) = |\Sigma|$ are optimal, it might make sense to study FPT-approaches even for fixed alphabets, because the running times that are obtainable at present are impractical for, say, the ASCII alphabet, not to speak about Unicode. (D) We are not aware of any studies concerning the polynomial-time approximability of the minimization problem related to $\{\mathbf{S}, \mathbf{D}\}$-S2S.

4 Second Case Study: Grammar-Based Compression

One of the main ideas behind data compression algorithms is to use regularities found in an input to find representations that are much smaller than the original data. Although data compression algorithms usually come with their own special data structures, there are some common schemes to be found in the algorithms of Lempel, Ziv, Storer, Szymanski and Welch from 1970s and 1980s [90,103, 108] that easily generalize to the idea to use context-free grammars producing singleton languages for data compression purposes. Such context-free grammars are also called straight-line programs in the literature. Another perspective on this question is offered by Grammatical Inference, a perspective that can be traced back to the work of Nevill-Manning and Witten [67–69] and Kieffer and Yang [55]. We refer to [86] for quite a number of other papers that link to Grammatical Inference and applications thereof.

This leads us to consider the following decision problem, called Grammar-Based Compression, or GBC for short: Given a word w over some alphabet Σ and an integer k, decide if there exists a context-free grammar G of size at most k such that $L(G) = \{w\}$.

In a sense, this question is still ill-posed, because we did not explain how to measure the size of a grammar. In fact, there are several natural ways to do this. The main results that we are citing in the following do not really depend on the choice. Hence, we follow the common definition that the size of a context-free grammar G is computed by summing up the lengths of all right-hand sides of rules of G.

Based on reductions due to Storer [89], Charikar *et al.* [19] showed the following complexity result.

Theorem 9. *GBC is* NP*-complete (on unbounded terminal alphabets).*

Although occasionally there were claims in the literature that such a hardness result would be also true for bounded alphabet sizes (see [7]), this question was in fact open until recently. In the journal version of [17], the authors showed the following result.

Theorem 10. *GBC is* NP*-complete (on terminal alphabets of size at least 17).*

Let us now again study this problem with a multivariate analysis. With grammars, we have some natural choices of parameters, given as input I an alphabet Σ, a word $w \in \Sigma^*$ and an integer k: $\kappa_1(I) = \Sigma$, $\kappa_2(I) = |w|$, $\kappa_3(I) = k$. Observe that we can assume (after some straightforward reductions) that $\kappa_1(I) \leq \kappa_3(I) \leq \kappa_2(I)$. This indicates that κ_1 is the most challenging parameterization, while finding FPT-results should be easiest for κ_2. We now look at these parameters in the chosen order. Our intuition on the strength of the parameters will be certified.

From Theorem 10, we can conclude:

Corollary 11. *GBC with parameterization κ_1 is not in* XP*, unless* P = NP.

Theorem 12 [17]**.** *GBC with parameterization κ_2 belongs to* FPT*. More precisely, instance $I = (\Sigma, w, k)$ can be solved in time $\mathcal{O}^*(3^{\kappa_2(I)})$.*

We now demonstrate that such an FPT-result also holds for κ_3.

Theorem 13. *GBC with parameterization κ_3 belongs to* FPT*.*

Proof. A context-free grammar $G = (N, \Sigma, R, S)$ generating a singleton is called an F-grammar if $F = \{u \in \Sigma^+ \mid \exists A \in N : A \Rightarrow^* u\}$. As exhibited in [17], a related Independent Dominating Set problem can be used to compute, for a given finite set F with $w \in F$, the smallest F-grammar that generates $\{w\}$ in polynomial time. For each F with $F \subseteq \Sigma^+$, $\sum_{u \in F} |u| \leq \kappa_3(\Sigma, w, k) = k$ we can find the smallest F-grammar for w in polynomial time. As $|\Sigma| \leq k$, only $f(k)$ many sets F have to be considered. □

In [17], the idea of a multivariate analysis has been taken further by considering further properties of the grammars we are looking for. For instance, is there a shortest grammar for w that uses $\leq r$ rules? The paper shows that this question is NP-hard. Considering r as a parameter, this problem is in XP and W[1]-hard.

There are also some studies on the approximability of the related minimization problem Min-GBC, see [7,17,19,51,59,81]. We now summarize the main results in this direction. Notice the strange role that the size of the alphabet plays again.

Theorem 14. *If $m^*(w)$ denotes the size of the smallest context-free grammar for string w, then* Min-GBC *can be approximated in polynomial time up to a factor of $\mathcal{O}\left(\log\left(\frac{|w|}{m^*(w)}\right)\right)$. Conversely, there is no polynomial-time algorithm achieving an approximation ratio better than $\frac{8569}{8568}$ unless* P = NP *(unbounded terminal alphabets). Furthermore, if there would be a polynomial-time constant-factor approximation algorithm for* Min-GBC *on binary alphabets, there would be also some polynomial-time constant-factor approximation algorithm for* Min-GBC *on unbounded alphabets.* Min-GBC *is APX-hard on bounded terminal alphabets.*

Charikar *et al.* [19] also proved an interesting connection to a long standing open problem on approximating so-called addition chains. This approach might be interesting from a Fine-Grained Complexity perspective.

Suggestions for Further Studies. (A) The most natural complexity task is to further reduce the size of the terminal alphabet in Theorem 10. More specifically: Is GBC still NP-hard for binary terminal alphabets? From a practical point of view, i.e., when applying this technique to compressing data files, this is crucial to know. (B) Is there a polynomial-time constant-factor approximation algorithm for the smallest grammar problem? (C) Storer and Szymanski [89,90] studied macro schemes that can be viewed as extensions of context-free grammars. No multivariate analysis of the related NP-hard compression problems has been undertaken so far. Notice that recent experimental studies [60] show the potential of these ideas. (D) We also mention generalizations of GBC to finite languages described by context-free grammars (and use them to encode specific words) as proposed in [86] which have not yet been studied from a more theoretical perspective.

5 Third Case Study: Synchronizing Words

A word $x \in \Sigma^*$ is called *synchronizing* for a deterministic finite automaton A, or DFA for short, with $A = (S, \Sigma, \delta, s_0, F)$ if there is a state s_f, such that for all states s, $\delta^*(s, x) = s_f$. An automaton is called *synchronizable* if it possesses a synchronizing word. It is known that A is synchronizable iff for every pair (s, s') of states, there exists a word $x_{s,s'}$ such that $\delta^*(s, x_{s,s'}) = \delta^*(s', x_{s,s'})$. This notion relates to the best known open combinatorial problem in Formal Languages, namely Černý's Conjecture: Any synchronizable DFA with t states has a synchronizing word of length $\leq (t-1)^2$. We are not going to give further details on this famous combinatorial problem, but only refer to the original paper by Černý [18], to two survey articles [82,95] that also describe a couple of applications, to one very recent paper [92] that describes the best upper bound of $(85059t^3 + 90024t^2 + 196504t - 10648)/511104$ on the length of a synchronizing word for a t-state synchronizable DFA.

Rather, we will now turn to the related decision problem Synchronizing Words, or DFA-SW for short. The input consists of a DFA A and an integer k. The question is if there is a synchronizing word w for A with $|w| \leq k$. In [26], Eppstein has shown the following complexity result:

Theorem 15. DFA-SW *is* NP*-complete.*

How could a multivariate analysis of this problem look like? Natural parameterizations of an instance $I = (A, k)$ with $A = (S, \Sigma, \delta, s_0, F)$ include: $\kappa_1(I) = |\Sigma|$, $\kappa_2(I) = |S|$, and $\kappa_3(I) = k$. Clearly, one could also study combined parameters, like $\kappa_4(I) = |I| + k$. Also, notice that $\kappa_5(I) = |\delta|$ corresponds to $|S^{\Sigma \times S}|$, which would therefore be again a combined parameter. We are going to report on results from [33,96].

Theorem 16. DFA-SW *with parameterization* $\kappa_1(I) = |\Sigma|$ *does not belong to* XP*, unless* P = NP*.*

In fact, the reduction from [26] can be used to show the previous results, as it shows NP-hardness for binary input alphabets.

Theorem 17. DFA-SW *with parameterization* $\kappa_2(I) = |S|$ *lies in* FPT*. More precisely, it can be solved in time* $\mathcal{O}^*(2^{\kappa_2(I)})$*. Yet, it does not admit polynomial kernels unless the polynomial-time hierarchy collapses to the third level.*

The FPT-algorithm is actually quite simple. It is based on the well-known subset construction, reducing the problem to a path-finding problem in an exponentially large graph. Yet, this algorithm is close to optimal in the following sense.

Proposition 18. *Assuming ETH,* DFA-SW *is not solvable in time* $\mathcal{O}^*(2^{o(\kappa_2(I))})$*.*

This result can be easily obtained by re-analyzing the reduction leading to Theorem 17.

Theorem 19. DFA-SW *with parameterization* $\kappa_3(I) = |k|$ *is* W*[2]-hard.*

We provide a sketch of this hardness result in Fig. 1, also to give an example of a parameterized reduction from Hitting Set introduced above. Observe that now the size of the input alphabet of the resulting DFA is unbounded, as it is the vertex set of the given hypergraph. Recently, Montoya and Nolasco [65] showed that (even) for planar automata, DFA-SW with parameterization $\kappa_3(I) = |k|$ is complete for the parameterized space complexity class NWL that embraces the whole W-hierarchy. Also, strong inapproximability results are known for the corresponding minimization problem; see [43].

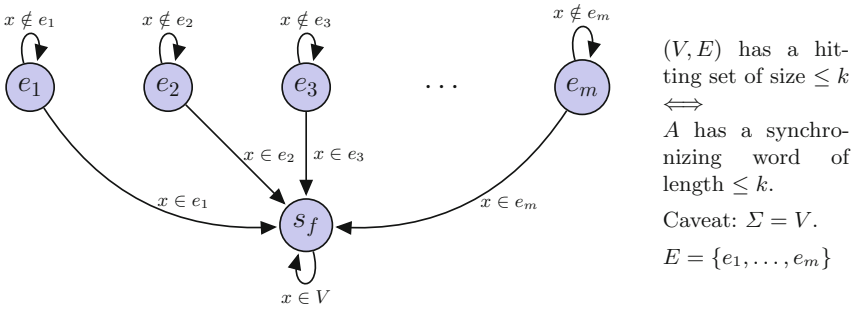

Fig. 1. An example showing how an FPT reduction works.

For the combined parameterization κ_4, we can state:

Theorem 20. DFA-SW *with parameterization* $\kappa_4(I) = |\Sigma| + k$ *lies in* FPT. *More precisely, it can be solved in time* $\mathcal{O}^*(|\Sigma|^k)$. *Yet, it does not admit polynomial kernels unless the polynomial-time hierarchy collapses to the third level. Moreover, there is no* $\mathcal{O}^*((|\Sigma| - \varepsilon)^k)$*-time algorithm, unless SETH fails.*

Suggestions for Further Studies. (A) Although the proof sketch in Fig. 1 indicates that DFA-SW remains NP-hard for rather restricted forms of automata, it might be interesting to study classes of subregular languages regarding the question if DFA-SW might become simpler when restricted to these classes. (B) There are quite a number of notions similar to synchronizing words that have been introduced over the years, also due to the practical motivation of this notion, see [82]. No systematic study of computational complexity aspects has been undertaken for all these notions. (C) In view of the FPT result concerning parameterization κ_2, the number of states, the parameterization κ_5 might not look that interesting. Yet, as κ_2 does not allow for polynomial kernels, this question could be of interest for the variation $\kappa_5'(I) = |S| + |\Sigma|$.

6 Fourth Case Study: Consistency Problem for DFAs

The problem DFA-Consistency takes as input analphabet Σ, two disjoint finite sets of words $X^+, X^- \subseteq \Sigma^*$, and some integer t. The question is if there is a DFA A with $\leq t$ states that is *consistent* with X^+, X^-, i.e., $L(A) \supseteq X^+$ and $L(A) \cap X^- = \emptyset$. This problem arises in various contexts, for instance, also in connection with Grammatical Inference; see [47]. Its classical complexity status was settled four decades ago.

Theorem 21 [6,44]**.** DFA-Consistency *is* NP*-complete.*

Let us explore the possible natural choices for parameterizations for instance $I = (\Sigma, X^+, X^+, t)$. We could look at $\kappa_1(I) = |\Sigma|$, $\kappa_2 = |X^+ \cup X^-|$, $\kappa_3(I) = \max\{|w| \mid w \in X^+ \cup X^-\}$, $\kappa_4(I) = t$, and there are quite a number of further ways to parameterize with respect to the sets X^+ and X^-. The NP-hardness results (with different constructions) extend to situations when $\kappa_1(I) = 2$ or when $\kappa_3(I) = 2$ or when $\kappa_4(I) = 2$. This immediately entails the following results.

Theorem 22. DFA-Consistency *with parameterization* $\kappa_1(I) = |\Sigma|$ *does not belong to* XP*, unless* P = NP.

Theorem 23. DFA-Consistency *with parameterization* $\kappa_3(I) = \max\{|w| \mid w \in X^+ \cup X^-\}$ *does not belong to* XP*, unless* P = NP.

Theorem 24. DFA-Consistency *with parameterization* $\kappa_4(I) = t$ *does not belong to* XP*, unless* P = NP.

Intuitively, this last result might appear most surprising, because there are only four ways how a letter can act with respect to two states. The literature situation was also a bit weird for some time. The result was mentioned in Sect. 1.2 of [75], as well as in [47], referring to an unpublished work of Angluin. However, no proof was given in these two references. Therefore, in the process of writing [33], we contacted Angluin, also to see how our solution compared to hers. We were quite impressed to receceive an email from Angluin within a couple of days, sending us a scanned copy of her proof, dating back to August 2nd, 1988. An NP-hardness proof can now be found in [33].

Interestingly, it is open if DFA-Consistency is NP-hard for any constant value of $\kappa_2(I)$. In the sense of multivariate analysis, we should continue to look into combined parameters. Let $\kappa_{i,j}$ for $1 \leq i < j \leq 4$ the parameterization given by $\kappa_{i,j}(I) = \kappa_i(I) + \kappa_j(I)$.

It was shown in [33] that DFA-Consistency is NP-hard even for 3-state DFAs with word lengths at most two in $X^+ \cup X^-$. This implies:

Theorem 25. DFA-Consistency *with parameterization* $\kappa_{3,4}(I)$ *does not belong to* XP*, unless* P = NP*.*

Conversely, by a trivial algorithm one can show the following (positive) result. The ETH hardness follows from a construction in [33].

Theorem 26. DFA-Consistency *with parameterization* $\kappa_{1,4}(I) = |\Sigma| + t$ *belongs to* FPT*, namely in time* $\mathcal{O}^*(t^{|\Sigma|t})$*. Assuming ETH, there is no* $\mathcal{O}^*(t^{o(|\Sigma|t)})$*-time algorithm for* DFA-Consistency*.*

The remaining parameter combinations seem to be open. Only one three-parameter combination was found in [33] that admitted a further FPT-result, combining κ_2, κ_3 and κ_4.

Suggestions for Further Studies. (A) Quite a number of parameter combinations are still open regarding their complexity status. Also, there are more parameters that could be related to X^+ and X^-. One potentially interesting scenario (pondering practical applications) would be to see what happens if there are much less negative than positive samples. (B) It might be an idea to look into classes of subregular languages and find some that allow for efficient consistency checks. (C) There are related questions that have not yet been studied from a multivariate perspective, for instance, what about Regular Expression Consistency?

7 Fifth Case Study: Lower Bounds for Universality

Possibly, the reader would have expected that we focus on problems like DFA Intersection Emptiness and similar problems traditionally studied (with respect to their complexity) in textbooks on Formal Languages. This will be partially rectified in this section. We will mainly concentrate on Universality,

which is the following problem. Given a finite automaton A with input alphabet Σ, is $L(A) = \Sigma^*$? Clearly, this makes only sense for nondeterministic automata, as the problem can be solved in linear time for DFAs. Natural parameters are the number t of states of A and the size of Σ. $|\Sigma|$ plays again an important role. Notice that the problem is PSPACE-complete in general, but co-NP-complete for unary input alphabets; see [56,87,88] and possibly more explicit in [34]. Natural parameterizations are $\kappa_1(A) = |\Sigma|$ and $\kappa_2(A) = |S|$, where $A = (S, \Sigma, \delta, s_0, F)$ is an NFA. By the classical hardness results, we see:

Proposition 27. Universality. *parameterized by κ_1, is not in* XP*, unless* P = NP.

Conversely, by the classical subset construction to produce a DFA, followed by final state complementation and a simple emptiness check, one sees:

Proposition 28. Universality. *parameterized by κ_2, belongs to* FPT.

We are now going to study complexity aspects under the ETH perspective.

Theorem 29 [34]**.** *Unless ETH fails, there is no $\mathcal{O}^*(2^{o(t^{1/3})})$ -time algorithm for deciding* Universality *on t-state NFAs with unary inputs.*

There is a slight gap to the known upper bound by Chrobak [20] who showed:

Theorem 30. Universality *on t-state NFAs with unary inputs can be solved in time $2^{\Theta((t \log t)^{1/2})}$.*

For larger alphabets, the situation looks a bit different for Universality.

Theorem 31 [34]**.** *Unless ETH fails, there is no $\mathcal{O}^*(2^{o(t)})$ -time algorithm for deciding* Universality *on t-state NFAs with binary inputs, or larger alphabets.*

The results is obtained by a parsimonous reduction from 3-Colorability. This is the correct bound, because the power-set construction gives that Universality on t-state NFAs can be solved in time $\mathcal{O}^*(2^t)$.

Also to overcome the fact that Universality is PSPACE-complete, a length-bounded variant has been introduced. LB-Universality: Given NFA A and length bound ℓ, does A accept all words up to length ℓ? This length bound puts the problem into NP. In fact, it is NP-complete. From a multivariate perspective, this introduces a natural third parameter, $\kappa_3(A, \ell) = \ell$.

Theorem 32. LB-Universality*, parameterized by κ_3, is* W*[2]-hard.*

As there is no formal proof of this result in the literature, we provide an explicit construction. In fact, it is quite similar to the construction illustrated in Fig. 1 which the reader might want to consult.

Proof. We show how to solve any instance $G = (V, E)$ and k of Hitting Set, parameterized by the size k of the solution, with the help of an instance of LB-Universality, parameterized by κ_3. Set $\Sigma = V$, $S = \{s_0, s_f\} \cup E$. Let s_0 be the initial and $E \cup \{s_0\}$ be the set of final states. We include the following transitions in the transition relation.

- (s_0, a, e) for any $a \in \Sigma$ and any $e \in E$;
- (e, a, e) for any $e \in E$ and $a \notin e$;
- (e, a, s_f) for any $e \in E$ and $a \in e$;
- (s_f, a, s_f) for any $a \in \Sigma$.

Furthermore, we set $\ell = k + 1$. Now, we claim that there is a hitting set of size at most k in G if and only if there is a word of length at most $k + 1$ that is not accepted by the constructed automaton. Namely, the only way not to accept a word by the automaton would be a word ending in s_f irrespectively what state $e \in E$ was entered inbetween. This shows that the encoded set of vertices of the hypergraph indeed hits all hyperedges. □

Membership in W[2] is unknown. As the parameterized complexity results for the other two parameters transfer, we get a rather diverse picture of what could happen in a multivariate analysis. As parameters κ_1 and κ_3 yield intractability results, the following (straightforward) result is interesting for the combined parameter $\kappa_1 + \kappa_3$.

Proposition 33. LB-Universality *can be solved in time* $\mathcal{O}^*(|\Sigma|^\ell)$.

Namely, just enumerate and test all strings up to length ℓ. This has been complemented by the following result that proves conditional optimality of this simple algorithm.

Theorem 34 [34]**.** *There is no algorithm that solves* LB-Universality *in time* $\mathcal{O}^*\left((|\Sigma| - \varepsilon)^\ell\right)$ *for any* $\varepsilon > 0$*, unless SETH fails.*

Let us finally discuss the issue of kernelization for this problem. The size of an instance is vastly dominated by the size of the transition table. Measured in terms of number of states and input alphabet size, this size can be as large as $\mathcal{O}(2^{|\Sigma||S|^2})$. Is there any hope to bring this down to a size only polynomial in $|S|$, a result that would complement Proposition 28? Interestingly, this question seems to be open, while it is possible to show the non-existence of polynomial kernels for the length-bounded variation. We can even show this result for the combined parameter $\kappa_2 + \kappa_3$.

Theorem 35. LB-Universality*, parameterized by* $\kappa_2 + \kappa_3$*, does not admit polynomial kernels, unless the polynomial-time hierarchy collapses to the third level.*

Proof. It is known that under the stated complexity assumptions, Hitting Set, parameterized by the number of hyperedges plus an upper-bound k on the size of the solution, does not admit polynomial kernels; see [23]. Now, assume that LB-Universality, parameterized by $\kappa_2 + \kappa_3$, would have a kernelization algorithm A that produces polynomial kernels. Now, start with an instance (G, k) of Hitting Set, where $G = (V, E)$, and first translate it to an equivalent instance (A, ℓ) of LB-Universality, using the construction from Theorem 32. Observe that $\kappa_2(A, \ell) = |E|$ and $\kappa_3(A, \ell) = k + 1$. Next, run algorithm A, yielding an

instance (A', ℓ') of size polynomial in $\kappa_2(A, \ell) + \kappa_3(A, \ell)$ and hence polynomial in $|E| + k$. Finally, observe that as LB-UNIVERSALITY is in NP, while HITTING SET is NP-hard, there is a polynomial-time transformation of (A', ℓ') into an equivalent instance (G', k') of HITTING SET. Clearly, also (G', k') would be of polynomial size, measured in $|E| + k$, which contradicts the non-existence of polynomial kernels. □

Let us mention one further exploit of the construction of Theorem 32.

Theorem 36. *Under the Set Cover Conjecture, there is no* $\mathcal{O}^*(2^{o(\kappa_2(A))})$*-time algorithm for solving instances* A *of* UNIVERSALITY.

Suggestions for Further Studies. (A) For the simple FPT-results for this (and similar) automata problems, polynomial kernel questions have barely been studied. This is also true for all the related classical automata problems. (B) There are slight but noticeable gaps between lower and upper bounds on running times (assuming ETH). More gaps can be found in related automata problems, as discussed in [34]. Is it possible to close these gaps, possibly using hypotheses different from ETH? (C) Unlike this section might suggest, most work has been put into studying automata intersection problems (among the classical algorithmic questions about finite automata); see [34,73,91,100,102]. Relatively few efforts have been put into related questions or into other automata models; we only mention here [37,38] and the references given in these papers.

8 Sixth Case Study: Parsing Theory

Coming from FL theory courses, where the Chomsky hierarchy is often taught with indicating a certain relevance to areas like Compiler Construction or also to (Computational) Linguistics, one might get disappointed when actually encountering these two mentioned areas. In the former case, the regular languages seem to be relevant and also some parts of the context-free languages, but not much more. In the second case, the situation is even more disillusioning: there, formal language classes more expressive than context-free but being not much more complex with respect to parsing are most interesting.

Even the typical parsing algorithms like CYK or Earley's mostly taught at FL undergraduate courses are not really relevant, as their cubic complexity is too much for typical applications in Compiler Construction. Rather, one resorts to *deterministic* context-free languages, also because they allow for giving unambiguous interpretations in the sense of unique parse trees. But is really necessary to spend cubic time for parsing context-free languages?

In a positive (algorithmic) sense, this question was answered already by Valiant [94] in a paper entitled *Parsing (general context-free recognition) in time* $\mathcal{O}(n^\omega)$. Here, n is the length of the string to be parsed, and ω is the exponent of multiplying two square matrices. At the time of that paper, this was still Strassen's multiplication, i.e., $\omega \approx 2.81$. If we want to use this method in a

practical algorithm, this might be still a method of choice. Yet, in theory, ω has improved to 2.3727, as shown by Vassilevska Williams in [105]. Whether it can be further improved or not, as well as relations to other problems, is discussed in a recent FOCS paper of Alman and Vassilevska Williams [4]. Alternatives to Valiant's original algorithm are discussed in [45,80]. Actually, Rytter called Valiant's algorithm *probably the most interesting algorithm related to formal languages*. This is a good reason to study it further here.

A natural question in our context is: Can we parse context-free grammars faster than multiplying matrices? This question was first addressed in a paper of Lee [58] with the title *Fast context-free parsing requires fast Boolean matrix multiplication.* The drawback of the underlying construction is that this is only true for grammars whose size grows with n^6, where n is the length of the string to be parsed. This is not a very realistic scenario. Abboud, Backurs, and Vassilevska Williams have fixed this issue in [1]. This fine-grained reduction works for a specific CF grammar, so that the previous dependence between grammar size and string length no longer holds. To get an idea how these results look like on a more technical level, we cite the following theorem.

Theorem 37. *There is context-free grammar G_{fix} of constant size such that if we can determine if a string of length n belongs to $L(G_{fix})$ in $T(n)$ time, then k-*Clique *on n-vertex graphs can be solved in $\mathcal{O}(T(n^{k/3+1}))$ time, for any $k \geq 3$.*

Hence, under the mentioned k-Clique Conjecture, context-free parsing cannot be faster than $\mathcal{O}(n^\omega)$.

We remark that there are extensions of context-free grammars, like Boolean grammars [72], that admit parsers like Valiant's; therefore, the lower bounds transfer to them immediately. For several related problems in computational biology, we refer to [13,74,107].

We are now reporting on one more problem directly related to parsing, namely to parsing tree-adjoining grammars. Notice that these are quite important for computational linguistics. We are not going to give a detailed introduction into tree-adjoining grammars, but rather refer to the textbook [54] that covers this and similar mechanisms from a linguistic yet mathematically profound perspective. Tree-adjoining grammars extend context-free ones in a way that allows for representing several linguistically relevant features beyond context-free languages. They yield one of the basic examples for mildly context-sensitive languages. The parsing is still possible in polynomial time, more precisely, the textbook algorithm will be in $\mathcal{O}(n^6)$, where n is the length of the string to be parsed. Yet, Rajasekaran and Yooseph's parser [76,77] runs in $\mathcal{O}(n^{2\omega})$. While this solved a previously well-known open problem in Computational Linguistics, it is interesting that a negative result pre-dated this algorithmic one. Satta [83] showed a reverse relation, actually inspiring Lee's work. But, not surprisingly, it comes with a similar drawback: This lower-bound is only true for grammars whose size grows with n^6. Bringmann and Wellnitz [15] have improved this result as follows.

Theorem 38. *There is a tree-adjoining grammar G_{fix} of constant size such that if we can decide in time $T(n)$ whether a given string of length n can be generated from G_{fix}, then $6k$-*Clique *can be solved in time $\mathcal{O}(T(n^{k+1} \log n))$, for any fixed $k \geq 1$.*

A consequence would be: An $\mathcal{O}(n^{2\omega-\varepsilon})$-algorithm for TAL recognition would prove that $6k$-Clique can be solved in time $\widetilde{\mathcal{O}}(n^{(2\omega-\varepsilon)(k+1)}) \subseteq \mathcal{O}(n^{(\omega/3-\delta)6k})$, contradicting the k-Clique Conjecture.

Suggestions for Further Studies. (A) Tree-adjoining grammars (TAGs) have been quite popular in Computational Linguistics in the 1990s, but this has calmed down a bit due to various shortcomings, both regarding parsing complexity (as discussed above) and the expressiveness of this formalism. Possibly, Formal Languages could help in the second issue by coming up with grammatical mechanisms that are more powerful than TAGs but do not need more computational resources for parsing. For instance, can the ideas underlying Boolean grammars be extended towards TAGs? (B) The whole topic of parsing has been a bit neglected in the Formal Language community. This is something that should change, in the best interest of the FL community. Whoever likes to start working in this direction should not overlook the rich annotated bibliography with nearly 2000 entries by Grune and Jacobs [46], available at https://dickgrune.com/Books/PTAPG_2nd_Edition/Additional.html. (C) Since four decades, it is open if EDT0L systems can be parsed in polynomial time [79, Page 315]. Weakening this question, one could also ask [29] if there is some $\mathcal{O}^*(f(|N|))$-time algorithm for parsing, where N is the set of nonterminal symbols.

9 Conclusions

With this survey, we could only highlight some of the many results that have been obtained in the meantime regarding multivariate analysis, but also regarding fine-grained complexity results. Yet, there are some common themes, as the role of the alphabet size, or also the richness of natural parameter choices. Another typical observation is that often simple parameterized algorithms cannot be improved under certain complexity assumptions. All this gives these problem a flavor different from, say, graph problems.

We preferred to focus on six problems, rather than trying to discuss all of them. Yet, in these conclusions, we are going to mention at least some further papers.

For instance, there is a vast body of literature on string problems. In fact, string problems were among the first ones where a true multivariate analysis was undertaken (without naming it such); see [27]. For a survey on these types of analyses for string problems, we refer to [16]. String problems have been also further investigated from the viewpoint of fine-grained complexity; see [2].

The related area of pattern matching would have also deserved a closer look. Let us suggest [35,39] and the literature cited therein for further reading. To the

readers otherwise more interested in graph-theoretic problems, it might be interesting to learn that the parameter treewidth well-known from graph algorithms has been also introduced in the context of patterns in [78].

String problems have also tight connections to several problems arising in computational biology. We refrain from giving any further references here, as this would finally surpass any reasonable length of the list of citations, but it should be clear that there are scores of papers on the parameterized and also on the fine-grained complexity of such problems.

In the context of stochastic automata, the Viterbi algorithm is central; its optimality is considered in [9].

Finally, let us discuss possible connections to Descriptional Complexity (within FL). One question of this sort is about smallest representations (within certain formalisms). One such example is also grammar-based compression, another one the minimization of automata or expressions, see [11,53]. Further on, one could consider questions as *Given an automaton, is there an equivalent representation with certain additional restrictions?* which are typical for this area, but have not yet been considered from a multivariate or fine-grained angle. We only refer to two survey papers of Holzer and Kutrib [48,49].

Acknowledgements. We are grateful to many people giving feedback to the ideas presented in this paper. In particular, Anne-Sophie Himmel, Ulrike Stege, and Petra Wolf commented on earlier versions of the manuscript.

References

1. Abboud, A., Backurs, A., Williams, V.V.: If the current clique algorithms are optimal, so is Valiant's parser. In: Guruswami, V. (ed.) IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS, pp. 98–117. IEEE Computer Society (2015)
2. Abboud, A., Williams, V.V., Weimann, O.: Consequences of faster alignment of sequences. In: Esparza, J., Fraigniaud, P., Husfeldt, T., Koutsoupias, E. (eds.) ICALP 2014. LNCS, vol. 8572, pp. 39–51. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-43948-7_4
3. Abu-Khzam, F.N., Fernau, H., Langston, M.A., Lee-Cultura, S., Stege, U.: A fixed-parameter algorithm for string-to-string correction. Discrete Optim. **8**, 41–49 (2011)
4. Alman, J., Williams, V.V.: Limits on all known (and some unknown) approaches to matrix multiplication. In: Thorup, M. (ed.) 59th IEEE Annual Symposium on Foundations of Computer Science, FOCS, pp. 580–591. IEEE Computer Society (2018)
5. Alon, N., Shpilka, A., Umans, C.: On sunflowers and matrix multiplication. Comput. Complex. **22**(2), 219–243 (2013)
6. Angluin, D.: On the complexity of minimum inference of regular sets. Inf. Control (Now Inf. Comput.) **39**, 337–350 (1978)
7. Arpe, J., Reischuk, R.: On the complexity of optimal grammar-based compression. In: 2006 Data Compression Conference (DCC), pp. 173–182. IEEE Computer Society (2006)

8. Backurs, A., Indyk, P.: Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). SIAM J. Comput. **47**(3), 1087–1097 (2018)
9. Backurs, A., Tzamos, C.: Improving Viterbi is hard: better runtimes imply faster clique algorithms. In: Precup, D., Teh, Y.W. (eds.) Proceedings of the 34th International Conference on Machine Learning, ICML, Proceedings of Machine Learning Research, vol. 70, pp. 311–321. PMLR (2017)
10. Barbay, J., Pérez-Lantero, P.: Adaptive computation of the swap-insert correction distance. ACM Trans. Algorithms **14**(4), 49:1–49:16 (2018)
11. Björklund, H., Martens, W.: The tractability frontier for NFA minimization. J. Comput. Syst. Sci. **78**(1), 198–210 (2012)
12. Bodlaender, H.L., Downey, R., Fomin, F.V., Marx, D. (eds.): The Multivariate Algorithmic Revolution and Beyond - Essays Dedicated to Michael R. Fellows on the Occasion of His 60th Birthday. LNCS, vol. 7370. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-30891-8
13. Bringmann, K., Grandoni, F., Saha, B., Williams, V.V.: Truly sub-cubic algorithms for language edit distance and RNA-folding via fast bounded-difference min-plus product. In: Dinur, I. (ed.) IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS, pp. 375–384. IEEE Computer Society (2016)
14. Bringmann, K., Künnemann, M.: Quadratic conditional lower bounds for string problems and dynamic time warping. In: Guruswami, V. (ed.) IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS, pp. 79–97. IEEE Computer Society (2015)
15. Bringmann, K., Wellnitz, P.: Clique-based lower bounds for parsing tree-adjoining grammars. In: Kärkkäinen, J., Radoszewski, J., Rytter, W. (eds.) 28th Annual Symposium on Combinatorial Pattern Matching, CPM. LIPIcs, vol. 78, pp. 12:1–12:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2017)
16. Bulteau, L., Hüffner, F., Komusiewicz, C., Niedermeier, R.: Multivariate algorithmics for NP-hard string problems. EATCS Bull. **114** (2014). http://bulletin.eatcs.org/index.php/beatcs/article/view/310/292
17. Casel, K., Fernau, H., Gaspers, S., Gras, B., Schmid, M.L.: On the complexity of grammar-based compression over fixed alphabets. In: Chatzigiannakis, I., Mitzenmacher, M., Rabani, Y., Sangiorgi, D. (eds.) International Colloquium on Automata, Languages and Programming, ICALP, Leibniz International Proceedings in Informatics (LIPIcs), vol. 55, pp. 122:1–122:14. Schloss Dagstuhl-Leibniz-Zentrum für Informatik (2016)
18. Černý, J.: Poznámka k homogénnym experimentom s konečnými automatmi. Matematicko-fyzikálny časopis **14**(3), 208–216 (1964)
19. Charikar, M., et al.: The smallest grammar problem. IEEE Trans. Inf. Theory **51**(7), 2554–2576 (2005)
20. Chrobak, M.: Finite automata and unary languages. Theor. Comput. Sci. **47**, 149–158 (1986)
21. Cook, S.A.: The complexity of theorem-proving procedures. In: Proceedings of the Third Annual ACM Symposium on Theory of Computing, STOC, pp. 151–158. ACM (1971)
22. Cygan, M., et al.: On problems as hard as CNF-SAT. ACM Trans. Algorithms **12**(3), 41:1–41:24 (2016)
23. Cygan, M., et al.: Parameterized Algorithms. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-319-21275-3
24. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Springer, Heidelberg (1999). https://doi.org/10.1007/978-1-4612-0515-9

25. Downey, R.G., Fellows, M.R.: Fundamentals of Parameterized Complexity. Texts in Computer Science. Springer, Heidelberg (2013)
26. Eppstein, D.: Reset sequences for monotonic automata. SIAM J. Comput. **19**(3), 500–510 (1990)
27. Fellows, M.R., Gramm, J., Niedermeier, R.: On the parameterized intractability of Closest Substring and related problems. In: Alt, H., Ferreira, A. (eds.) STACS 2002. LNCS, vol. 2285, pp. 262–273. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45841-7_21
28. Fellows, M.R., Jansen, B.M.P., Rosamond, F.A.: Towards fully multivariate algorithmics: parameter ecology and the deconstruction of computational complexity. Eur. J. Combin. **34**(3), 541–566 (2013)
29. Fernau, H.: Parallel grammars: a phenomenology. GRAMMARS **6**, 25–87 (2003)
30. Fernau, H.: Parameterized Algorithmics: A Graph-Theoretic Approach. Universität Tübingen, Germany, Habilitationsschrift (2005)
31. Fernau, H.: Parameterized algorithmics for d-hitting set. Int. J. Comput. Math. **87**(14), 3157–3174 (2010)
32. Fernau, H.: A top-down approach to search-trees: improved algorithmics for 3-hitting set. Algorithmica **57**, 97–118 (2010)
33. Fernau, H., Heggernes, P., Villanger, Y.: A multi-parameter analysis of hard problems on deterministic finite automata. J. Comput. Syst. Sci. **81**(4), 747–765 (2015)
34. Fernau, H., Krebs, A.: Problems on finite automata and the exponential time hypothesis. Algorithms **10**(24), 1–25 (2017)
35. Fernau, H., Manea, F., Mercaş, R., Schmid, M.L.: Pattern matching with variables: fast algorithms and new hardness results. In: Mayr, E.W., Ollinger, N. (eds.) 32nd International Symposium on Theoretical Aspects of Computer Science (STACS 2015), Leibniz International Proceedings in Informatics (LIPIcs), vol. 30, pp. 302–315. Schloss Dagstuhl-Leibniz-Zentrum für Informatik (2015)
36. Fernau, H., Meister, D., Schmid, M.L., Stege, U.: Editing with swaps and inserts on binary strings (2014). Manuscript
37. Fernau, H., Paramasivan, M., Schmid, M.L., Thomas, D.G.: Simple picture processing based on finite automata and regular grammars. J. Comput. Syst. Sci. **95**, 232–258 (2018)
38. Fernau, H., Paramasivan, M., Schmid, M.L., Vorel, V.: Characterization and complexity results on jumping finite automata. Theor. Comput. Sci. **679**, 31–52 (2017)
39. Fernau, H., Schmid, M.L., Villanger, Y.: On the parameterised complexity of string morphism problems. Theory Comput. Syst. **59**(1), 24–51 (2016)
40. Flum, J., Grohe, M.: Parameterized Complexity Theory. Springer, Heidelberg (2006). https://doi.org/10.1007/3-540-29953-X
41. Fomin, F.V., Kratsch, D., Woeginger, G.J.: Exact (exponential) algorithms for the dominating set problem. In: Hromkovič, J., Nagl, M., Westfechtel, B. (eds.) WG 2004. LNCS, vol. 3353, pp. 245–256. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30559-0_21
42. Fomin, F.V., Kratsch, D.: Exact Exponential Algorithms. Texts in Theoretical Computer Science. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-16533-7
43. Gawrychowski, P., Straszak, D.: Strong inapproximability of the shortest reset word. In: Italiano, G.F., Pighizzini, G., Sannella, D.T. (eds.) MFCS 2015. LNCS, vol. 9234, pp. 243–255. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48057-1_19
44. Gold, E.M.: Complexity of automaton identification from given data. Inf. Control (Now Inf. Comput.) **37**, 302–320 (1978)

45. Graham, S.L., Harrison, M.A., Ruzzo, W.L.: An improved context-free recognizer. ACM Trans. Program. Lang. Syst. **2**(3), 415–462 (1980)
46. Grune, D., Jacobs, C.J.H.: Parsing Techniques - A Practical Guide. Monographs in Computer Science. Springer, Heidelberg (2008). https://doi.org/10.1007/978-0-387-68954-8
47. Higuera, C.: Grammatical inference. Learning automata and grammars. Cambridge University Press, Cambridge (2010)
48. Holzer, M., Kutrib, M.: The complexity of regular(-like) expressions. Int. J. Found. Comput. Sci. **22**(7), 1533–1548 (2011)
49. Holzer, M., Kutrib, M.: Descriptional and computational complexity of finite automata - a survey. Inf. Comput. **209**(3), 456–470 (2011)
50. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley, Reading (1979)
51. Hucke, D., Lohrey, M., Reh, C.P.: The smallest grammar problem revisited. In: Inenaga, S., Sadakane, K., Sakai, T. (eds.) SPIRE 2016. LNCS, vol. 9954, pp. 35–49. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46049-9_4
52. Impagliazzo, R., Paturi, R., Zane, F.: Which problems have strongly exponential complexity? J. Comput. Syst. Sci. **63**(4), 512–530 (2001)
53. Jiang, T., Ravikumar, B.: Minimal NFA problems are hard. SIAM J. Comput. **22**(6), 1117–1141 (1993)
54. Kallmeyer, L.: Parsing Beyond Context-Free Grammars. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14846-0
55. Kieffer, J.C., Yang, E.: Grammar-based codes: a new class of universal lossless source codes. IEEE Trans. Inf. Theory **46**(3), 737–754 (2000)
56. Kozen, D.: Lower bounds for natural proof systems. In: 18th Annual Symposium on Foundations of Computer Science, FOCS, pp. 254–266. IEEE Computer Society (1977)
57. Kratochvíl, J.: A special planar satisfiability problem and a consequence of its NP-completeness. Discrete Appl. Math. **52**, 233–252 (1994)
58. Lee, L.: Fast context-free grammar parsing requires fast boolean matrix multiplication. J. ACM **49**(1), 1–15 (2002)
59. Lehman, E., Shelat, A.: Approximations algorithms for grammar-based compression. In: Thirteenth Annual Symposium on Discrete Algorithms SODA. ACM Press (2002)
60. Liao, K., Petri, M., Moffat, A., Wirth, A.: Effective construction of relative lempel-ziv dictionaries. In: Bourdeau, J., Hendler, J., Nkambou, R., Horrocks, I., Zhao, B.Y. (eds.) Proceedings of the 25th International Conference on World Wide Web, WWW, pp. 807–816. ACM (2016)
61. Lichtenstein, D.: Planar formulae and their uses. SIAM J. Comput. **11**, 329–343 (1982)
62. Lokshtanov, D., Marx, D., Saurabh, S.: Lower bounds based on the exponential time hypothesis. EATCS Bull. **105**, 41–72 (2011)
63. Masek, W.J., Paterson, M.: A faster algorithm computing string edit distances. J. Comput. Syst. Sci. **20**(1), 18–31 (1980)
64. Meister, D.: Using swaps and deletes to make strings match. Theor. Comput. Sci. **562**, 606–620 (2015)
65. Andres Montoya, J., Nolasco, C.: On the synchronization of planar automata. In: Klein, S.T., Martín-Vide, C., Shapira, D. (eds.) LATA 2018. LNCS, vol. 10792, pp. 93–104. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-77313-1_7
66. Nešetřil, J., Poljak, S.: On the complexity of the subgraph problem. Comment. Math. Univ. Carolinae **26**(2), 415–419 (1985)

67. Nevill-Manning, C.G.: Inferring sequential structure. Ph.D. thesis, University of Waikato, New Zealand (1996)
68. Nevill-Manning, C.G., Witten, I.H.: Identifying hierarchical structure in sequences: a linear-time algorithm. J. Artif. Intell. Res. **7**, 67–82 (1997)
69. Nevill-Manning, C.G., Witten, I.H.: On-line and off-line heuristics for inferring hierarchies of repetitions in sequences. Proc. IEEE **88**, 1745–1755 (2000)
70. Niedermeier, R.: Invitation to Fixed-Parameter Algorithms. Oxford University Press, Oxford (2006)
71. Niedermeier, R.: Reflections on multivariate algorithmics and problem parameterization. In: Marion, J.Y., Schwentick, T. (eds.) 27th International Symposium on Theoretical Aspects of Computer Science (STACS 2010), Leibniz International Proceedings in Informatics (LIPIcs), vol. 5, pp. 17–32. Schloss Dagstuhl-Leibniz-Zentrum für Informatik (2010)
72. Okhotin, A.: Parsing by matrix multiplication generalized to Boolean grammars. Theor. Comput. Sci. **516**, 101–120 (2014)
73. de Oliveira Oliveira, M., Wehar, M.: Intersection non-emptiness and hardness within polynomial time. In: Hoshi, M., Seki, S. (eds.) DLT 2018. LNCS, vol. 11088, pp. 282–290. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98654-8_23
74. Pinhas, T., Zakov, S., Tsur, D., Ziv-Ukelson, M.: Efficient edit distance with duplications and contractions. Algorithms Mole. Biol. **8**, 27 (2013)
75. Pitt, L., Warmuth, M.K.: The minimum consistent DFA problem cannot be approximated within any polynomial. J. ACM **40**, 95–142 (1993)
76. Rajasekaran, S.: Tree-adjoining language parsing in $O(n^6)$ time. SIAM J. Comput. **25**(4), 862–873 (1996)
77. Rajasekaran, S., Yooseph, S.: TAL recognition in $O(M(n^2))$ time. J. Comput. Syst. Sci. **56**(1), 83–89 (1998)
78. Reidenbach, D., Schmid, M.L.: Patterns with bounded treewidth. Inf. Comput. **239**, 87–99 (2014)
79. Rozenberg, G., Salomaa, A.K.: The Mathematical Theory of L Systems. Academic Press, Cambridge (1980)
80. Rytter, W.: Context-free recognition via shortest paths computation: a version of Valiant's algorithm. Theor. Comput. Sci. **143**(2), 343–352 (1995)
81. Rytter, W.: Application of Lempel-Ziv factorization to the approximation of grammar-based compression. Theor. Comput. Sci. **302**, 211–222 (2003)
82. Sandberg, S.: 1 homing and synchronizing sequences. In: Broy, M., Jonsson, B., Katoen, J.-P., Leucker, M., Pretschner, A. (eds.) Model-Based Testing of Reactive Systems. LNCS, vol. 3472, pp. 5–33. Springer, Heidelberg (2005). https://doi.org/10.1007/11498490_2
83. Satta, G.: Tree-adjoining grammar parsing and Boolean matrix multiplication. J. Comput. Linguist. **20**(2), 173–191 (1994)
84. Shannon, C.E.: A universal Turing machine with two internal states. In: Shannon, C.E., McCarthy, J. (eds.) Automata Studies, Annals of Mathematics Studies, vol. 34, pp. 157–165. Princeton University Press (1956)
85. Sirakov, B., de Souza, P.N., Viana, M. (eds.): Proceedings of the International Congress of Mathematicians 2018 (ICM 2018). World Scientific (2019)
86. Siyari, P., Gallé, M.: The generalized smallest grammar problem. In: Verwer, S., van Zaanen, M., Smetsers, R. (eds.) Proceedings of the 13th International Conference on Grammatical Inference, ICGI 2016, JMLR Workshop and Conference Proceedings, vol. 57, pp. 79–92. JMLR.org (2017)

87. Stockmeyer, L.J.: The complexity of decision problems in automata theory and logic. Ph.D. thesis, Massachusetts Institute of Technology, Department of Electrical Engineering (1974)
88. Stockmeyer, L.J., Meyer, A.R.: Word problems requiring exponential time: preliminary report. In: Aho, A.V., et al. (eds.) Proceedings of the 5th Annual ACM Symposium on Theory of Computing, STOC, pp. 1–9. ACM (1973)
89. Storer, J.A.: NP-completeness results concerning data compression. Technical report 234, Department of Electrical Engineering and Computer Science, Princeton University, USA, November 1977
90. Storer, J.A., Szymanski, T.G.: Data compression via textual substitution. J. ACM **29**(4), 928–951 (1982)
91. Swernofsky, J., Wehar, M.: On the complexity of intersecting regular, context-free, and tree languages. In: Halldórsson, M.M., Iwama, K., Kobayashi, N., Speckmann, B. (eds.) ICALP 2015. LNCS, vol. 9135, pp. 414–426. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-47666-6_33
92. Szykuła, M.: Improving the upper bound on the length of the shortest reset word. In: Niedermeier, R., Vallée, B. (eds.) 35th Symposium on Theoretical Aspects of Computer Science (STACS 2018), Leibniz International Proceedings in Informatics (LIPIcs), vol. 96, pp. 56:1–56:13. Schloss Dagstuhl-Leibniz-Zentrum für Informatik (2018)
93. Tovey, C.A.: A simplified NP-complete satisfiability problem. Discrete Appl. Math. **8**, 85–89 (1984)
94. Valiant, L.G.: General context-free recognition in less than cubic time. J. Comput. Syst. Sci. **10**(2), 308–315 (1975)
95. Volkov, M.V.: Synchronizing automata and the Černý conjecture. In: Martín-Vide, C., Otto, F., Fernau, H. (eds.) LATA 2008. LNCS, vol. 5196, pp. 11–27. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-88282-4_4
96. Vorel, V., Roman, A.: Parameterized complexity of synchronization and road coloring. Discrete Math. Theor. Comput. Sci. **17**, 283–306 (2015)
97. Wagner, R.A.: On the complexity of the extended string-to-string correction problem. In: Proceedings of seventh Annual ACM Symposium on Theory of Computing, STOC 1975, pp. 218–223. ACM Press (1975)
98. Wagner, R.A., Fischer, M.J.: The string-to-string correction problem. J. ACM **21**(1), 168–173 (1974)
99. Wahlström, M.: Algorithms, measures and upper bounds for satisfiability and related problems. Ph.D. thesis, Department of Computer and Information Science, Linköpings universitet, Sweden (2007)
100. Todd Wareham, H.: The parameterized complexity of intersection and composition operations on sets of finite-state automata. In: Yu, S., Păun, A. (eds.) CIAA 2000. LNCS, vol. 2088, pp. 302–310. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44674-5_26
101. Watt, N.: String to string correction kernelization. Master's thesis, University of Victoria, Canada (2013)
102. Wehar, M.: Hardness results for intersection non-emptiness. In: Esparza, J., Fraigniaud, P., Husfeldt, T., Koutsoupias, E. (eds.) ICALP 2014. LNCS, vol. 8573, pp. 354–362. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-43951-7_30
103. Welch, T.A.: A technique for high-performance data compression. IEEE Comput. **17**, 8–19 (1984)

104. Williams, V.V.: Hardness of easy problems: basing hardness on popular conjectures such as the strong exponential time hypothesis (invited talk). In: Husfeldt, T., Kanj, I.A. (eds.) 10th International Symposium on Parameterized and Exact Computation, IPEC, LIPIcs, vol. 43, pp. 17–29. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2015)
105. Williams, V.V.: Multiplying matrices faster than Coppersmith-Winograd. In: Karloff, H.J., Pitassi, T. (eds.) Proceedings of the 44th Symposium on Theory of Computing Conference, STOC, pp. 887–898. ACM (2012)
106. Wong, C.K., Chandra, A.K.: Bounds for the string editing problem. J. ACM **23**(1), 13–16 (1976)
107. Zakov, S., Tsur, D., Ziv-Ukelson, M.: Reducing the worst case running times of a family of RNA and CFG problems, using Valiant's approach. Algorithms Mole. Biol. **6**, 20 (2011)
108. Ziv, J., Lempel, A.: Compression of individual sequences via variable-rate coding. IEEE Trans. Inf. Theory **24**, 530–536 (1978)