A Multi-Parameter Analysis of Hard Problems on Deterministic Finite Automata[☆]

Henning Fernau[a,*], Pinar Heggernes[b], Yngve Villanger[b]

[a] *Fachbereich 4, Abteilung Informatikwissenschaften, Universität Trier, D-54286 Trier, Germany*
[b] *Department of Informatics, University of Bergen, N-5020 Bergen, Norway*

Abstract

We initiate a multi-parameter analysis of two well-known NP-hard problems on deterministic finite automata (DFAs): the problem of finding a short synchronizing word, and that of finding a DFA on few states consistent with a given sample of the intended language and its complement. For both problems, we study natural parameterizations and classify them with the tools provided by Parameterized Complexity. Somewhat surprisingly, in both cases, rather simple FPT algorithms can be shown to be optimal, mostly assuming the (Strong) Exponential Time Hypothesis.

Keywords: Deterministic finite automata, NP-hard decision problems, synchronizing word, consistency problem

1. Introduction

The multi-parameter analysis of computationally hard problems [28, 50] tries to answer fundamental questions about what actually makes a problem hard, by systematically considering so-called natural parameters that can be singled out in an instance or in a target structure. The importance of this research area is underlined by European Association of Theoretical Computer Science through a recently introduced award, the Nerode Prize for outstanding papers in multivariate algorithmics. In problems dealing with finite automata, natural parameters could be the size of the input alphabet or the number of states. For instance, if some hardness reduction produces or requires automata with large input alphabets, then this proof does not reveal much for the cases where only binary input alphabets are of interest. Parameterized Complexity offers tools to tell whether hardness could be expected when fixing, say, the alphabet size.

[☆]This work is supported by the Norwegian and the German Research Councils. An extended abstract has appeared at LATA 2013 [29].

[*]Corresponding author; Tel. +49 651 201 2827.

Email addresses: fernau@uni-trier.de (Henning Fernau), pinar.heggernes@ii.uib.no (Pinar Heggernes), yngve.villanger@ii.uib.no (Yngve Villanger)

Using these tools, we aim to answer the question what aspects of our problem cause it to become hard. The possible choices of parameters are abundant, and we consider our paper as a starting point of multivariate algorithmics within the theory of finite automata. This line of research has been pursued only to a very limited extent so far on finite automata problems, two exceptions being NFA minimization [8] and Mealy machines with census requirements [27], offering ample ground to work on.

In this paper, we study the parameterized complexity, with respect to various natural parameters, of two problems related to deterministic finite automata (DFAs): the problem of finding a shortest synchronizing word in a DFA, and that of finding the smallest DFA consistent with a given sample consisting of positive and of negative examples of the intended language. Both problems have a long history, dating back to the very beginning of automata theory, and both questions have found many practical applications. We classify these problems using the tools provided by Parameterized Complexity, like fixed-parameter tractability, W-hardness, and kernelization, with respect to the studied parameters. In several cases, we are able to observe that the positive results we obtain are optimal up to widely used complexity theoretical assumptions.

Our first main problem, Synchronizing Word (SW), is the following decision problem. Given a DFA $A = (S, I, \delta, s_0, F)$ with state set S, input alphabet I, transition function $\delta : S \times I \to S$, initial state s_0 and set of final states F, together with an integer $k \geq 0$, decide if there exists a synchronizing word of length at most k for A. Here, a *synchronizing word* is a string $\sigma \in I^*$ such that there exists a state $s_f \in S$ such that, for any start state $s \in S$, $\delta^*(s, x) = s_f$. Hence, a synchronizing word enables to reset an automaton to some well-defined state, wherever it may start. Therefore, it is also known as a *reset sequence* and even under many other different names. This notion and several related ones that we are going to discuss draw their practical motivation from testing circuits and automata; we refer to [43, 45, 61].

Eppstein showed that SW is NP-complete [24]. Later, Berlinkov proved that the related *optimization* problem MIN-SW cannot belong to APX under some complexity assumptions [6, 7]. Walker observed that Eppstein's reduction not only works when starting from 3-SAT, but also when using SAT [71]. This will be useful for our purposes.

We show that SW is W[2]-hard when parameterized by the natural parameter k. This also provides an alternative proof of the mentioned NP-completeness result. As our reduction is from Hitting Set, it shows in addition that MIN-SW cannot be approximated even up to some logarithmic factor, where the logarithmic function takes the size of the input alphabet as its argument [2, 25, 45]. This is not the same as Berlinkov's result, as he focuses on small alphabet sizes. It would be interesting to know if SW with parameter k actually belongs to W[2]. Otherwise, it might be one of the few natural problems known to be placed in higher levels of the W-hierarchy; see the discussions in [13].

Related complexity considerations on Synchronizing Word can be found in [34, 52, 60]. Also experimental results on this problem have been pub-

lished [15, 54, 56, 57]. The related combinatorial questions are nicely reported and reviewed in [61, 68]. The most important question in that area is settling Černý's conjecture [11] that each t-state DFA that has a synchronizing word has also one of length at most $\frac{t(t-1)}{2}$; it was even argued that this bound is only seldomly attained [63]. Currently best upper bounds are of size cubic in t, the record holder being Trahtman [65]. The conjecture has been verified for several classes of automata, most notably aperiodic automata; see [66]. The relation to the (polynomially solvable) Road Coloring Problem should also be noted here [48, 58, 59, 67, 64].

In this paper, we also briefly consider related problems, mostly on Mealy machines, like finding shortest homing sequences.

As our second main problem on automata, we consider DFA Consistency, which is also a decision problem. Here, the input consists of an alphabet I, two finite disjoint set of words $X^+, X^- \subseteq I^*$, and an integer t. The question is whether there exists a DFA A with at most t states such that $X^+ \subseteq L(A)$ and $X^- \cap L(A) = \emptyset$. This problem was extensively studied in the area of Algorithmic Learning Theory, especially in Grammatical Inference, as it is central to the model of *Learning in the Limit*, initiated by Gold's seminal work [35]. As the problem was shown to be NP-hard [3, 36, 38], and its optimization version hard to approximate [46, 53], several heuristics and translations to other problems were proposed. In our context, reductions to coloring problems seem to be most relevant [17, 18], but using SAT solvers is also an interesting approach [37]. Results concerning heuristics are reviewed in [9, 14, 44, 51] and the literature quoted therein, which shows the practical importance of this problem, for instance, for verification. We can further motivate our studies with a quotation from [38, p. 139]: *"Alternative proofs of the hardness of the consistency problem would be of help, in order to better understand what is really hard."* This can be seen as a quest for a multi-parameter analysis for DFA Consistency, as we commence in this paper.

Note that DFA Consistency can be seen as an implementation of Occam's razor in the sense that the goal is the shortest explanation (in terms of DFAs) for the given sample. This principle can lead to computationally hard problems in the context of regular languages when only positive examples are given, taking into account questions concerning the representability or coding of the sample words. This kind of questions were investigated in [26] from the viewpoint of Parameterized Complexity. Clearly, the consistency problem can be also asked for other types of automata and grammars. As long as the universal language I^* has a simple representation in the corresponding class of languages, the consistency problem is trivial if $X^- = \emptyset$. However, there are interesting classes of languages where this is not the case. For instance, it has been shown in [16] that this type of consistency problem is W[2]-hard for a whole range of categorial grammar families. Further Parameterized Complexity results for algorithmic learning problems can be found in [4, 21, 62].

We would like to point out that, in addition to the fact that there is almost

no earlier work dealing with a multi-parameter analysis of finite automata problems, there are at all not many papers dealing with Parameterized Complexity of problems concerning finite automata or related models in general; we refer to [5, 27, 55, 72, 73].

Our paper is organized as follows. In the next section we give the necessary background, definitions and notation. Section 3 deals with the multi-parameter analysis of the SW problem, and Section 4 that of DFA Consistency. In Section 5, we briefly study some related problems, and we give our concluding remarks in Section 6.

2. Preliminaries

In this paper, a graph $G = (V, E)$ is undirected and unweighted, with vertex set V and edge set E. Given a subset $X \subseteq V$, the subgraph of G induced by X is denoted by $G[X]$. A vertex subset X is an *independent set* if $G[X]$ has no edges. A partition of V into $V_1, V_2, \ldots, V_r$ is called a *proper r-coloring* if $G[V_i]$ is an independent set for $1 \leq i \leq r$.

A *deterministic finite automaton (DFA)* A is a tuple (S, I, δ, s_0, F), where S is the set of states, I is the input alphabet, $\delta : S \times I \to S$ is the transition function, s_0 is the initial state, and F is the set of final states. We will use $t = |S|$.

For an introduction to the well established field of Parameterized Complexity, we refer the reader to the textbooks [22, 23, 30, 49]. Here we give a short and informal overview. A decision problem is said to be a *parameterized* problem if its input can be partitioned into a *main part* J and a *parameter* P. A parameterized problem with main input size $|J|$ and parameter size $|P|$ is said to be *fixed-parameter tractable* (FPT) if it can be solved by an algorithm with running time $O^*(f(|P|))$, where f is a computable function depending only on P and not on J, and the O^*-notation suppresses all factors that are polynomial in $|J|$. It is well-known that a parameterized problem is FPT if and only if it has a *kernel*, meaning that there is a polynomial time algorithm that produces an equivalent instance J' of size $|J'| \in O(g(|P|))$, where g is again a function depending only on P. If g is a polynomial function, then the problem is said to admit a *polynomial kernel*. Whether or not a fixed-parameter tractable problem admits a polynomial kernel is a broad subfield of Parameterized Complexity which attracts more and more attention.

The complexity class FPT consists of the fixed-parameter tractable problems, and there exists a hierarchy of complexity classes above FPT:

$$\text{FPT} \subseteq \text{W[1]} \subseteq \text{W[2]} \subseteq \ldots \subseteq \text{W}[p] \subseteq \text{XP}$$

Showing that a parameterized problem is hard for any of the classes above FPT makes it unlikely to be fixed-parameter tractable. Proving such hardness can be done by reducing from a problem that is known to be hard for any of these classes, as in Classical Complexity. However, in this case the reduction needs

to be *parameter preserving*, meaning that the size of the parameter in the new instance has to be bounded by a function of the size of the parameter in the original instance. Another way to prove hardness is using the class XP, which is the class of parameterized problems that are solvable in time $O(|J|^{h(|P|)})$ for some function h. Consequently, if a problem is NP-hard when $|P|$ is a constant, then it is not even likely to belong to XP, let alone FPT.

Next we define some well-known problems and complexity theoretical assumptions, which will be useful for proving hardness results and lower bounds in the rest of the paper.

Problem: r-SAT
Input: A boolean CNF formula φ on n variables and m clauses, where each clause contains at most r literals.
Question: Is there a truth assignment that satisfies φ?

If there is no bound on the number of literals that a clause can contain, then we simply refer to the problem as SAT.

One common way to argue for the unlikeness of a subexponential algorithm is to use the *Exponential Time Hypothesis* (ETH) by Impagliazzo, Paturi, and Zane. For an introduction to ETH, see [31, 47]. By the observation that each variable is used at least once, and by the Sparsification Lemma [41], ETH can be expanded as follows:

Exponential Time Hypothesis (ETH)[41]: There is a positive real s such that 3-SAT instances on n variables and m clauses cannot be solved in time $2^{sn}(n+m)^{O(1)}$. Moreover, there is a real $s' > 0$ such that 3-SAT instances on m clauses cannot be solved in time $2^{s'(m+n)}(n+m)^{O(1)}$.

A slightly stronger assumption is the following one.

Strong Exponential Time Hypothesis (SETH)[41][10]: SAT cannot be solved in time $2^{sn}(n+m)^{O(1)}$ for $s < 1$.

In order to argue for the unlikeliness of polynomial kernels we will use the following result.

Proposition 1 ([32]). SAT*, parameterized by the number n of variables, does not have a kernel that is polynomial in n unless* NP $\subseteq$ coNP/poly*.*

We will also make use of the following hard problems in our reductions.

Problem: r-Coloring
Input: A graph $G = (V, E)$ on n vertices and m edges.
Question: Is there a proper r-coloring of G?

It is well-known that 3-Coloring is NP-complete [33].

Problem: Hitting set
Input: A family $\mathcal{F}$ of sets over a universe $\mathcal{U}$ and an integer k.
Parameter: k
Question: Does there exist a set $\mathcal{S} \subset \mathcal{U}$ such that $|\mathcal{S}| \leq k$ and $F \cap \mathcal{S} \neq \emptyset$ for each $F \in \mathcal{F}$?

Hitting set is W[2]-complete with respect to parameter k [22].

We are now equipped with the necessary background to start proving our results. We start with Synchronizing Word in the next section.

3. Synchronizing Word

Given a DFA $A = (S, I, \delta, s_0, F)$, a *$k$-synchronizing word* is a string $x \in I^*$ of length at most k that satisfies the following condition: there exists a state $s_f \in S$ such that, for any start state $s \in S$, $\delta^*(s, x) = s_f$. Note that s_f need not be a final state. In fact, the final states or the initial state do not play a role at all. In this section, we consider different parameterizations of the following problem.

Problem: Synchronizing Word (SW)
Input: A DFA $A = (S, I, \delta, s_0, F)$ and an integer k.
Question: Is there a k-synchronizing word for A?

The most natural parameter is of course k. As we will show, the problem is W[2]-hard with respect to this parameter. We will consider other natural parameters as well; these are $t = |S|$ and $|I|$. We will show that whereas the problem remains intractable with respect to parameter $|I|$, it becomes FPT with respect to parameter t. The mentioned two intractability results motivate parameterizing the problem with both k and $|I|$, and we obtain tractability in this case. Recall that s_0 and F are irrelevant for SW, so they do not contribute to natural parameters. Observe that $|\delta|$ is basically t times $|I|$, hence this parameter is implicitly covered by our studies. Table 1 summarizes the results of this section, and we prove each of them separately.[1]

For the first hardness result, we first need the following lemma.

Lemma 2. *Given a* Hitting Set *instance with a set family $\mathcal{F}$ over a universe $\mathcal{U}$ and some integer k, a DFA $A = (S, I, \delta, s_0, F)$ can be constructed in time $O(|\mathcal{F}||\mathcal{U}|)$, such that $|S| = |\mathcal{F}| + k + 1$, $|I| = |\mathcal{U}|$, and A has a k-synchronizing word if and only if $\mathcal{F}$ has a hitting set of size k.*

Proof. Let $\mathcal{U} = \{e_1, \ldots, e_n\}$ be the set of elements comprising the universe $\mathcal{U}$ and let $\mathcal{F} = \{F_1, \ldots, F_m\}$ be a set of subsets of the universe. $\mathcal{U}$, $\mathcal{F}$ and the integer k together yield the Hitting Set instance. The construction of the DFA $A = (S, I, \delta, s_0, F)$ is performed as follows. For each element in the

[1] The second negative kernel result is not shown in this paper; we refer to Remark 2.

Parameter	Parameterized Complexity	Polynomial Kernel
k	W[2]-hard	—
$\|I\|$	NP-complete for $\|I\| = 2$	—
k and $\|I\|$	FPT with running time $O^*(\|I\|^k)$	Not unless NP $\subseteq$ coNP/poly
t	FPT with running time $O^*(2^t)$	Not unless NP $\subseteq$ coNP/poly

Table 1: A summary of the results of Section 3. In addition, we show that the two given running time upper bounds are tight in the sense that we cannot expect to solve SW in time $O^*(2^{o(t)})$ or in time $O^*((|I| - \epsilon)^k)$ for any $\epsilon > 0$.

universe $\mathcal{U}$ we create a symbol in the alphabet, i.e., for simplicity, $e_i \in I$ for $1 \leq i \leq n$, so that $I = \mathcal{U}$. The state set S contains the following elements:

- elementary states q_i, $1 \leq i \leq k$;
- set family states f_j, $1 \leq j \leq m$;
- one sink state s.

Hence, $|S| = |\mathcal{F}| + k + 1$. As mentioned above, s_0 and F need not be explicitly specified. The transitions are described next:

1. $\delta(q_i, e_j) = q_{i+1}$, with $q_{k+1} = s$,for every $e_j \in \mathcal{U}$ and $1 \leq i \leq k$;
2. $\delta(f_j, e_i) = f_j$ for every element $e_i \notin F_j$.
   $\delta(f_j, e_i) = s$ for every element $e_i \in F_j$.
3. $\delta(s, e_i) = s$ for any element $e_i \in \mathcal{U}$.

Notice that, as there are no transitions leading from the sink state s to any other state, the state in which the synchronizing word (if it exists) must end is clear, it must be s. A shortest path from q_1 to s goes through $q_2, \ldots, q_k$ and thus any k-synchronizing word is of length exactly k. In order to reach s from every set family state f_j, the word has to contain a symbol corresponding to an element in the set F_j. As a result we get that if $\sigma = b_1 \cdots b_k$ is a synchronizing word, then $\{b_1, \ldots, b_k\}$ is a hitting set of size k for our given Hitting Set instance.

For the opposite direction, let $\{b_1, \ldots, b_k\}$ be a hitting set of size at most k. Then we simply use $\sigma = b_1 \cdots b_k$ as the synchronizing word. If the hitting set is smaller than k, we can simply repeat one symbol until a word of length k is obtained. □

As the reduction given in the proof of Lemma 2 is parameter preserving for parameter k, we can deduce the first row of Table 1.

Theorem 3. Synchronizing Word *is W[2]-hard when parameterized with* k.

Proof. By Lemma 2 any Hitting Set instance with universe $\mathcal{U}$, family $\mathcal{F}$, and integer k, can be reduced to a synchronizing word instance $(A = (S, I, \delta, s_0, F), k)$

where $|S| = k + |\mathcal{F}| + 1$, $|I| = |\mathcal{U}|$, and any synchronizing word is of size exactly k. If the SW instance could be solved it in time $O^*(f(k))$, then it would imply that W[2] = FPT, since Hitting Set is W[2]-complete with respect to parameter k. □

Regarding the optimization versions of the two problems, the reduction in the proof of Lemma 2 can be seen as an approximation preserving transformation from Minimum Hitting Set to Minimum Synchronizing Word, since the parameter k does not change. In particular, any approximate solution obtained for the latter problem can be transformed back to an approximate solution of the originally given Minimum Hitting Set instance with the same approximation guarantee. We can therefore conclude the following from [25].

Corollary 4. Minimum Synchronizing Word *cannot be approximated within a factor of* $(1-\varepsilon)\ln(|I|)$ *for any* $\varepsilon > 0$*, unless* NP $\subseteq$ DTIME$(n^{\ln(\ln(n))})$.

When we instead parameterize SW with $|I|$, we will obtain that the problem is not even likely to be in XP. For that result, we first need the following.

Lemma 5 ([24],[71]). *Given a CNF formula* φ *with* n *variables and* m *clauses, a DFA* $A = (S, I, \delta, s_0, F)$ *can be constructed in* $O(nm)$ *time, such that* $|S| = nm + m + 1$*,* $|I| = 2$*, and* A *has an* n*-synchronizing word if and only if* φ *has a satisfying truth assignment.*

Theorem 6. Synchronizing Word *is NP-complete when* $|I| = 2$.

Proof. By Lemma 5 any CNF formula can be reduced to a SW instance $(A = (S, I, \delta, s_0, F), k)$ where $|I| = 2$. As $f(2)$ is a constant, the existence of an $O^*(n^{f(|I|)})$-time algorithm for SW would imply that SAT could be solved in polynomial time. □

As neither parameter k nor parameter $|I|$ is useful for fixed-parameter tractability, a natural next step is to use both k and $|I|$ as a combined parameter. In that case the problem becomes trivially FPT, as we show below.

Theorem 7. Synchronizing Word *is FPT when parameterized with* $|I|$ *and* k*; it can be solved in time* $O^*(|I|^k)$.

Proof. Observe that there are exactly $|I|^k$ words of length k. By increasing length, enumerate all words up to length k, and check if the current word is synchronizing for the DFA. As there are at most $k|I|^k$ words on at most k symbols the claim holds. □

Since the above result is so straightforward, one could hope for an improvement or a polynomial kernel for SW when parameterized with both k and $|I|$. Interestingly, no such improvement seems likely, as we show next, hence the above result is tight.

Theorem 8. Synchronizing Word *cannot be solved in time* $O^*((|I|-\epsilon)^k)$ *for any* $\epsilon > 0$ *unless SETH fails.*

Proof. By Lemma 5, any CNF formula can be reduced to a SW instance $(A = (S, I, \delta, s_0, F), k)$ where $|I| = 2$ and $k = n$. If there existed an algorithm that solved the latter in time $O^*((|I|-\epsilon)^k)$ for $\epsilon > 0$ that would imply that SAT could be solved in time $O^*(2^{sn})$ for $s < 1$, contradicting SETH. □

Theorem 9. Synchronizing Word *does not have a polynomial kernel when parameterized with both* k *and* $|I|$ *unless* NP $\subseteq$ coNP/poly.

Proof. By Lemma 5, any CNF formula can be reduced to an SW instance $(A = (S, I, \delta, s_0, F), k)$ with $|I| = 2$ and $k = n$. If there existed a polynomial algorithm that produced an equivalent instance of size polynomial in $k, |I|$, this would mean that the number of states is reduced. As SW is NP-complete, there exists a polynomial time reduction back to a CNF formula with n' variables and m' clauses where $n' + m'$ is polynomial in $k, |I|$. This would imply that the number of clauses in this new CNF formula is bounded by a polynomial in n, and it is thus a polynomial kernel for SAT when parameterized by the number of variables n. By Proposition 1 this implies that NP $\subseteq$ coNP/poly. □

Finally we turn our attention to the version of the problem parameterized with t. Again, there is a straightforward FPT algorithm. However, as above, the running time of this algorithm is best possible up to some complexity theoretical assumptions, as we show below.

Theorem 10 ([61]). Synchronizing Word *is FPT when parameterized with* t*; it can be solved in time* $O^*(2^t)$.

Proof. Let (S, I, δ, s_0, F) be a DFA and k be an integer, which together comprise the given SW instance. The algorithm starts by incrementally constructing an auxiliary directed graph $D = (V, A)$, with $V = 2^S$. Initially let $A = \emptyset$. For each pair (X, z) where $X \in V$, i.e., $X \subseteq S$, and $z \in I$ add an arc (X, Y) to A where $Y = \{\delta(x, z) \mid x \in X\}$ is the set of states that can be reached from a state in X by reading symbol z. The graph D has 2^t vertices and $2^t|I|$ arcs. To check if there is a k-synchronizing word, do a breadth-first search in D of depth k starting in the vertex S. The DFA has a k-synchronizing word if and only if the breadth-first search reaches a vertex $Z \subseteq S$ such that $|Z| = 1$. □

Again, this simple FPT result cannot be improved, assuming ETH holds. In order to prove this lower bound result, we need the following lemma.

Lemma 11. *Given a CNF formula* φ *with* n *variables and* m *clauses, a DFA* $A = (S, I, \delta, s_0, F)$ *can be constructed in* $O(nm)$ *time, such that* $|S| = n+m+1$, $|I| = 2n$, *and* A *has an* n*-synchronizing word if and only if* φ *is satisfiable.*

PROOF. Let $V = \{x_1, \ldots, x_n\}$ be a set of variables in φ. Let $C = \{c_1, \ldots, c_m\}$ be the set of clauses in φ. We assume, without loss of generality, that no variable occurs twice in any clause. Hence, each c_i can be viewed as a subset of the set of literals

$$L = \{x_1, \ldots, x_n\} \cup \{\overline{x_1}, \ldots, \overline{x_n}\}.$$

The alphabet I contains $2n$ symbols x_i and $\overline{x_i}$ for $1 \leq i \leq n$, corresponding to the literals in the formula, i.e., $I = L$. We have the following $n + m + 1$ states in S:

- Variable states q_i, $1 \leq i \leq n$;
- clause states c_j , $1 \leq j \leq m$;
- one sink state s.

The transitions are as follows:

1. $\delta(q_i, x_i) = \delta(q_i, \overline{x_i}) = q_{i+1}$, with $q_{n+1} = s$;
   $\delta(q_i, x_j) = \delta(q_i, \overline{x_j}) = q_i$ if $j \neq i$.
2. $\delta(c_j, l) = c_j$ for literal l if $l \notin c_j$.
   $\delta(c_j, l) = s$ for literal l if $l \in c_j$.
3. $\delta(s, l) = s$ for any literal l.

Notice that, as there are no transitions leading from the sink state to any other state, the state in which the synchronizing word (if it exists) must end is clear, it must be s. Any synchronizing word must be of length at least n as this is the length of the shortest path from q_1 to s. More precisely, any synchronizing word must be of the form described by the following regular expression:

$$(x_1 \cup \overline{x_1})^+(x_2 \cup \overline{x_2})^+ \cdots (x_v \cup \overline{x_n})^+(x_1 \cup \overline{x_1} \cup x_2 \cup \overline{x_2} \cup \cdots \cup x_n \cup \overline{x_n})^*.$$

This word should reflect the variable assignment. Namely, if there is a synchronizing word $\sigma = l_1 \cdots l_n$ of length n, then we can read off a variable assignment $\Phi : V \to \{0, 1\}$ as follows:

$$\Phi(x_i) = \begin{cases} 1, & \text{if } l_i = x_i \\ 0, & \text{if } l_i = \overline{x_i} \end{cases}$$

As σ leads into s in particular for each state c_j, this means that each clause c_j is satisfied by construction. Conversely, if there is a satisfying truth assignment $\Phi : V \to \{0, 1\}$, then we build the string $\sigma = l_1 \ldots l_n$ of length n where

$$l_i = \begin{cases} x_i, & \text{if } \Phi(x_i) = 1 \\ \overline{x_i}, & \text{if } \Phi(x_i) = 0 \end{cases}$$

It is not hard to verify that σ is a synchronizing word. □

Theorem 12. SYNCHRONIZING WORD *cannot be solved in time* $O^*(2^{o(t)})$ *unless ETH fails.*

PROOF. We start by using Lemma 11 to reduce a 3-SAT instance on n variables and m clauses to a SW instance $(A = (S, I, \delta, s_0, F), k)$ where $t = n + m + 1$, $|I| = 2n$, and $k = n$. If an algorithm existed that solved any SW instance in $O^*(2^{o(t)})$, then it would also solve 3-SAT in $O^*(2^{o(n+m)})$ time, contradicting ETH. □

We have thus completed the proofs of the results presented in Table 1, and we can move to the next main problem of our paper.

4. DFA Consistency

In this section, we consider various parameterizations of the following problem.

Problem: DFA CONSISTENCY
Input: An alphabet I, two finite disjoint sets of words $X^+, X^- \subseteq I^*$, and an integer t.
Question: Is there a DFA A with at most t states such that X^+ is accepted by A and X^- is rejected by A?

The natural parameters we work with here are the number of states t in the target DFA, the alphabet size $|I|$, the number of given words $c = |X^+ \cup X^-|$, and the maximum length of any of the words in $X^+ \cup X^-$, $\ell = \max\{|\sigma| \mid \sigma \in X^+ \cup X^-\}$. The results of this section are summarized in Table 2. Like in the previous section, when the problem remains intractable when parameterized with one of these parameters, we seek tractability by combining several of the parameters. Note that the special case when $c = 2$ has been studied as a separate problem, called SEPARATING WORD (for DFAs); see [20] for a recent overview.

Parameter	Parameterized Complexity	Lower bound (under ETH)
t	NP-complete for $t = 2$	-
ℓ	NP-complete for $\ell = 2$	-
$\|I\|$	NP-complete for $\|I\| = 2$	-
c	Open	-
t, ℓ	NP-complete for $t\ell = 6$	-
$t, \|I\|$	FPT, running time $O^*(t^{t\|I\|})$	No $O^*(t^{o(t\|I\|)})$-time algorithm
t, c	Open	-
t, c, ℓ	FPT, running time $O^*(t^{c\ell})$	No $O^*(t^{o(c\ell)})$-time algorithm
$t, c, \ell, \|I\|$	FPT by the above	No $O^*((t\ell)^{o(t\ell(c+\|I\|))})$-time algorithm
$t, c, \ell, \|I\|$	FPT by the above	No $O^*(\|I\|^{o(\|I\|(t+c+\ell))})$-time algorithm

Table 2: The table summarizes the results of Section 4. The last two rows give the most general lower bound claim we can give under the current reductions. In addition we show that the parameter combination $(t, |I|)$ does not admit a polynomial kernel.

We now proceed to prove the results of the table. First we need two lemmas which deal with the cases $t = 2$ and $t = 3$: Lemmas 13 and 14. For the first case when $t = 2$, we fix notations as follows. Let p be the initial state and q be the second state. Let $\delta : S \times I \to S$, with $S = \{p, q\}$, be the transition function of the 2-state DFA we are considering. Hence, each letter $a \in I$ induces a mapping $\delta_a : S \to S$ with $\delta_a(x) = \delta(x, a)$ for $x \in S$. There are four different mappings δ_a, that are depicted in Table 3. If a is synchronizing word leading into state s, then we call δ_a an *s-synchronizer*. Notice that in [1], slightly different names are used for the four possible mappings.

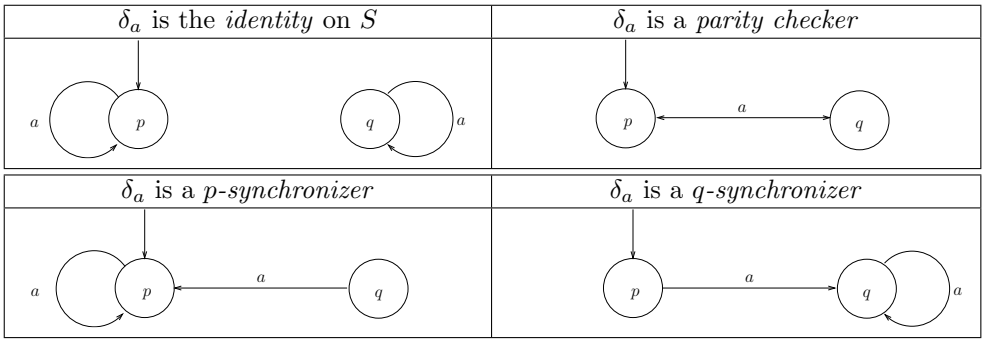

Table 3: Four different types of 2-state DFAs.

Lemma 13. *Given a CNF formula φ with n variables and m clauses, an instance of* DFA Consistency *can be constructed in time $O(nm)$, where $t = 2$, $|I| = 3n+1$, $\ell = 2n$, and $c = 6n+m+3$, such that there exists a DFA on these parameters that accepts X^+ and rejects X^- if and only if φ is satisfiable.*

Upon completing this work, we learned that Dana Angluin already knew the NP-hardness of DFA consistency when $t = 2$. This is mentioned as a note in [1] and also, without reference, in the textbook [38]. However, whereas Angluin's proof is similar to a proof of a related problem in [1, Theorem 5.1], our proof differs from these proofs substantially. As a consequence, we decided to keep Lemma 13 and our proof it in this paper, as it is one of the cornerstones of the overall multi-parameter analysis that we give.

Proof. We reduce from a SAT instance, i.e., a CNF formula φ, on variables $V = \{x_1, \ldots, x_n\}$ and clauses $C = \{c_1, c_2, \ldots, c_m\}$. The DFA Consistency instance is created as follows: Number of states $t = 2$ and the alphabet I is set to be $\{x, \overline{x}, x' \mid x \in V\} \cup \{a\}$ where $x, \overline{x}$ are the positive and negative literals for variable $x \in V$. Let clause c consist of literals $l_1, l_2, \ldots, l_q$ (in an arbitrary but fixed sequence) and let $\sigma_c = l_1 l_1 l_2 l_2 \ldots l_q l_q$ be the word representing clause c_i. The accept set X^+ and the reject set X^- of words are now defined as follows:

$$\begin{aligned} X^+ &= \{a\} \cup \{x, \overline{x}, ax'x' \mid x \in V\} \cup \{\sigma_c \mid c \in C\} \\ X^- &= \{\varepsilon, aa\} \cup \{x'x', a\overline{x}ax', axx' \mid x \in V\} \end{aligned}$$

Assume that there exists a DFA $A = (S, I, \delta, p, F)$ respecting the parameters that accepts X^+ and rejects X^-. As $|S| = 2$, let us name the two states p, q and, without loss of generality, let p be the start state. By the empty word $\varepsilon \in X^-$, it is clear that p is the only reject state and q is the only accept state.

Based on the restriction that $t = 2$, $I = \{x, \overline{x}, x' \mid x \in V\} \cup \{a\}$, $\{a\} \cup \{x, \overline{x}, ax'x' \mid x \in V\} \subseteq X^+$, and $X^- = \{\varepsilon, aa\} \cup \{x'x', a\overline{x}ax', axx' \mid x \in V\}$, some basic observations can be made regarding the transition function δ:

$$\delta(p, a) = q, \delta(q, a) = p \text{ as } a \in X^+ \text{ and } aa \in X^-$$

$$\delta(p, y) = q, \text{ where } y \in \{x, \overline{x} \mid x \in V\}$$

Having $x'x' \in X^-$ and $ax'x' \in X^+$ leaves us with two possible state transition functions corresponding to x'. Either x' loops in both p and q (giving the identity) or it moves from p to q and back (implementing a parity test).

Both words $a\overline{x}ax'$ and axx' should lead into p. This means that (irrespectively of the function that x' realizes) $a\overline{x}a$ and ax lead into the same state. If both x and $\overline{x}$ loop in q, then $a\overline{x}$ and ax lead into the same state, namely q. If both x and $\overline{x}$ cause transitions from q to p, then $a\overline{x}$ and ax lead into the same state, namely p. In both cases, as a lets the automaton change states, $a\overline{x}a$ and ax would end up in different states. Thus, we can conclude that for each $x \in V$,

$$(\delta(q, x) = p, \delta(q, \overline{x}) = q, \delta(p, x') = p, \delta(q, x') = q) \text{ or}$$

$$(\delta(q, x) = q, \delta(q, \overline{x}) = p, \delta(p, x') = q, \delta(q, x') = p).$$

Consider now a clause word $\sigma_c = l_1 l_1 l_2 l_2 \ldots l_q l_q \in X^+$. Let the symbols of σ_c be numbered $a_1, a_2, \ldots, a_{2q}$. Consider a number i where $1 \leq i \leq q$. There are two cases when reading symbol number $2i - 1$ from σ_c, depending on whether DFA A is in p or q. Remember that $a_{2i-1} = a_{2i}$.

In the first case A is in state q when reading a_{2i-1}. If $\delta(q, a_{2i-1}) = q$ then $\delta(q, a_{2i}) = q$ and A is in state q when reading symbol number $2(i + 1) - 1$. Otherwise, $\delta(q, a_{2i-1}) = p$ but as $\delta(p, a_{2i}) = q$ we start in q when reading symbol number $2(i + 1) - 1$. So for any $1 \leq i \leq q$, if A is in q when reading a_{2i-1}, it means that σ_c will end in q and is thus accepted.

In the remaining case A is in state p when reading a_{2i-1}. We already know that $\delta(p, a_{2i-1}) = q$. If $\delta(q, a_{2i}) = q$ then the word will be accepted by the q case above. Otherwise, $\delta(q, a_{2i}) = p$ and we are back at p when reading symbol number $2(i + 1) - 1$.

As $\sigma_c \in X^+$, the crucial thing for us is that there exists some integer $1 \leq i \leq q$ such that $\delta(q, a_{2i}) = q$. The satisfying assignment of variables in the SAT instance can now be read off from the DFA A. Variable x is assigned true value if $\delta(q, x) = q$ and false value if $\delta(q, \overline{x}) = q$. By the arguments above exactly one of these cases occurs. As every word σ_c for some $c \in C$ is accepted it is clear that this is a satisfying assignment for the SAT instance.

In the opposite direction, let V_a be a subset of V such that the CNF formula is satisfied by setting variables from V_a to true and from $V \setminus V_a$ to false. Let us construct the δ function accepting X^+ and rejecting X^-.

Like in the opposite direction, we get by words not associated to clauses that

$$\delta(p,a) = q, \delta(q,a) = p, \text{namely by } a \in X^+ \text{and } aa \in X^- \text{ and}$$

$$\delta(p,y) = q \text{ for } y \in \{x, \overline{x} \mid x \in V\}.$$

Using the subset V_a of V, the rest of the δ function is defined as follows:

$$(\delta(q,x) = p, \delta(q,\overline{x}) = q, \delta(p,x') = p, \delta(q,x') = q) \text{ if } x \notin V_a$$
$$(\delta(q,x) = q, \delta(q,\overline{x}) = p, \delta(p,x') = q, \delta(q,x') = p) \text{ if } x \in V_a$$

As every word σ_c for a clause $c \in C$ has a true literal it is clear that all words in X^+ are accepted and all words in X^- are rejected. □

Remark 1. The reduction from SAT in this proof is important when we aim at optimality results based on ETH. Conversely, if one aims at "small instances", a quick re-analysis of our proof allows us to state the following.

Given a 3-CNF formula φ with n variables and m clauses, an instance of DFA Consistency can be constructed in time $O(nm)$, where $t = 2$, $|I| = 3n + 1$, $\ell = 6$, and $c = 6n + m + 3$, such that there exists a DFA on these parameters that accepts X^+ and rejects X^- if and only if φ is satisfiable.

Lemma 14. *Let $G = (V, E)$ on n vertices and m edges be an instance of* 3-Coloring. *Then an instance of* DFA Consistency *can be constructed in time $O(n + m)$, where $t = 3$, $|I| = n + m$, $\ell = 2$, $c = 2m$, and such that there exists a DFA on these parameters that accepts X^+ and rejects X^- if and only if G is 3-colorable.*

Proof. Let us first construct the DFA Consistency instance and then argue that it is a YES instance if and only if the 3-Coloring instance is a YES instance. We start by setting $t = 3$ and $I = V \cup E$, which leaves the definition of X^+ and X^-. Let $v_1, \ldots, v_n$ be an arbitrary numbering of the vertices in V. The sets of words X^+ and X^- are now constructed as follows:

- $X^+ = \{v_i e \mid e = v_i v_j \in E, i < j\}$;
- $X^- = \{v_j e \mid e = v_i v_j \in E, i < j\}$.

This completes the construction of the DFA Consistency instance.

Let us now argue for the equivalence of the two instances. For the first direction we assume that there exists a DFA $A = (S, I, \delta, s_0, F)$ on three states and alphabet $I = V \cup E$ that accept X^+ and rejects X^-. Let s_0, s_1, s_2 be the states of A and let v_i be contained in V_q for $0 \leq q \leq 2$ if $\delta(s_1, v_i) = s_q$. This gives us a partitioning V_0, V_1, V_2 of V. Our objective will now be to argue that V_q is an independent set in G for $0 \leq q \leq 2$. For the sake of contradiction,

let $e = v_iv_j \in E$ where $i < j$ be an edge such that $v_i, v_j \in V_q$. From the construction of X^+ and X^- it is clear that set X^+ contains word v_ie and set X^- contains v_je. As the only difference between these two words is the first symbol and one word is accepted and the other one is rejected, it is clear that different states are reached by reading v_i and v_j from the start state s_0. Thus, either v_i or v_j is not contained in V_q and the contradiction is obtained.

For the second direction assume that there is a partitioning V_0, V_1, V_2 of V such that V_q is an independent set for $0 \leq q \leq 2$. Name the three states s_0, s_1, s_2 and let s_0 be the start state and s_1 the only accepting state. The function δ is now defined as follows:

1. $\delta(s_1, v_i) = s_q$, for $v_i \in V_q$ where $0 \leq q \leq 2$;
2. $\delta(s_q, v_iv_j) = s_1$ for $0 \leq q \leq 2$ and $i < j$;
3. $\delta(s_q, v_iv_j) = s_2$ for $0 \leq q \leq 2$ and $i > j$;

It is not hard to verify that all words in X^+ are accepted and all words in X^- are rejected. □

The two lemmas above correspond to the results presented in the first two rows of Table 2. Before we state these results formally, we also show the following.

Lemma 15. *Given a CNF formula φ with n' variables and m' clauses, an instance of* DFA Consistency *can be constructed in time $O((n'+m')^2)$, where $t \leq 3m' + n' + 2$, $|I| = 2$, $\ell \leq 3m' + n' + 2$, and $c \leq 13m' + n' + 4$, such that there exists a DFA on these parameters that accepts X^+ and rejects X^- if and only if φ is satisfiable.*

Our proof parallels the ones sketched in [36, 38], but it makes the handling of the parameters more lucid.

Proof. Observe first that if a clause $c = \{\ell_1, \ldots, \ell_p; \ell'_1, \ldots, \ell'_r\}$ of φ contains $p > 0$ positive literals and $r > 0$ negative ones, then we can simply add a new variable y_c and replace the clause c by two clauses, $c_+ = \{\ell_1, \ldots, \ell_p, y\}$ and $c_- = \{\ell'_1, \ldots, \ell'_r, \overline{y}\}$ to get an equivalent formula. We can thus assume that each clause in φ contains either only positive literals or only negative literals.

Let $\{x_1, \ldots, x_n\}$ be the variables of the new φ and $C = \{c_1, \ldots, c_m\}$ be the clauses. By the previous reduction we have that $n \leq n' + m'$ and $m \leq 2m'$. We will show how to build an instance of DFA Consistency as described in the statement of the lemma. Let C_P denote the set of clauses containing only positive literals and let C_N denote the rest of the clauses, with $C = C_P \cup C_N$. We build a DFA Consistency instance with alphabet $I = \{a, b\}$, $t = m+n+2$, and with the following sets X^+ and X^-. Start with $X^+ = X^- = \emptyset$.

1. Add $\varepsilon, b, a^{m+n+2}, a^{m+n+1}b$ to X^+.
2. Add a^k to X^-, for $1 \leq k \leq m + n + 1$.
3. Add a^ib, a^iba to X^- and a^ibbb to X^+, for $1 \leq i \leq m$.

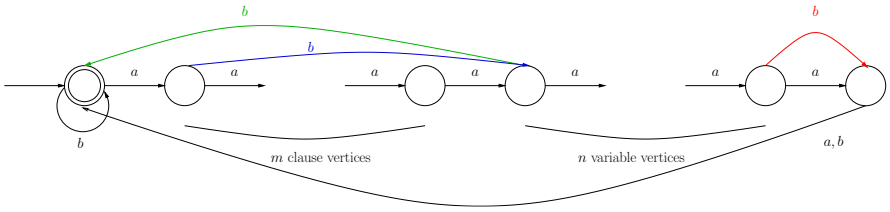

Figure 1: Blue b-arcs model the choice of literals per clause. Property (B) excludes b-arcs between clause vertices. Green b-arcs indicate variables set to true. Red b-arcs indicate variables set to false.

4. Add $a^i bb$ to X^+, for every $c_i \in C_P$.
5. Add $a^i bb$ to X^- and $a^i bba$ to X^+, for $c_i \in C_N$.

This completes the construction of the DFA Consistency instance. Since we can assume that $n \geq 1$, the length of each of the words in $X^+ \cup X^-$ is at most $\ell = m + n + 2$. We can furthermore assume that $C_P \neq \emptyset$, and thus $|X^+ \cup X^-| \leq 4 + (m+n+1) + 3m + (2m-1) = 6m+n+4$. Due to $n \leq n' + m'$ and $m \leq 2m'$, the bounds claimed in the statement of the lemma follow.

We will now list some properties, called (A) and (B) below, that every DFA consistent with X^+ and X^- must satisfy. In the following arguments, we will write q_w to denote the state of the DFA that is reached after reading the input word w. The reader might find Figure 1 useful while reading the below.

(A) Due to items 1 and 2, every DFA consistent with X^+ and X^- has at least $m+n+2$ states that can be denoted by the a-sequences that lead to them, i.e., $q_{a^0}, \ldots, q_{a^{m+n+1}}$. Note that $q_{a^i} \neq q_{a^k}$ for $0 \leq i < k \leq m+n+1$, since $q_{a^k a^{m+n+2-k}}$ is an accepting state due to item 1, while $q_{a^i a^{m+n+2-k}}$ is not an accepting state if $i < k$. Hence all these states are different.

(B) In every DFA consistent with X^+ and X^-, any b-arc from q_{a^i} (for $c_i \in C$) always leads to a state q_{a^k} for some $k = m+1, \ldots, m+n$. To see this (by contradiction), assume first that there were a b-arc from q_{a^i} to q_{a^k} for some $c_i \in C$, $0 < k \leq m$. Due to item 3, $a^k b \in X^-$. In our case, $q_{a^k b} = q_{a^i bb}$, which means that $a^i bb$ is rejected by the DFA. By items 4 and 5, this implies that $c_i \in C_N$. Item 3 then requires that $a^i bbb \in X^+$, which means that $a^k bb$ is accepted by the DFA, as $q_{a^i bbb} = q_{a^k bb}$. By the case distinction from items 4 and 5, this entails that $c_k \in C_P$. By item 3, $q_{a^k ba}$ is not an accepting state, while $q_{a^i bba}$ is accepting due to item 5. This is a contradiction, as $q_{a^k ba} = q_{a^i bba}$. Assume next that there were a b-arc from q_{a^i} to q_ε. By item 1, q_ε is accepting, while by item 3, $q_{a^i b}$ is not an accepting state. This is a contradiction, as by our assumption, $q_\varepsilon = q_{a^i b}$. Assume finally that there were a b-arc from q_{a^i} to $q_{a^{m+n+1}}$. By item 3, $q_{a^i ba}$ is rejecting, while $q_{a^{m+n+1}a}$ is accepting due to item 1, which is a contradiction, as by assumption $q_{a^i ba} = q_{a^{m+n+1}a}$.

We are now ready to prove that the formula φ is satisfiable if and only if the DFA Consistency instance $(X^+, X^-, t = m+n+2)$ we constructed above has a solution.

For the first direction, assume that φ is satisfiable. We are going to construct a DFA A with $t = m + n + 2$ states and transition function δ that is consistent with X^+ and X^-. As indicated by (A), we denote the states of A as $q_{a^0}, \ldots, q_{a^{m+n+1}}$. To satisfy items 1 and 2, we define:

- $\delta(q_{a^i}, a) = q_{a^{i+1}}$ for $0 \leq i \leq m + n + 1$. (Here, $q_\varepsilon = q_{a^{m+n+2}}$.)
- $\delta(q_\varepsilon, b) = \delta(q_{a^{m+n+1}}, b) = q_\varepsilon$.

What is left to be defined are the b-arcs emanating from states q_{a^i} for $1 \leq i \leq m + n$. Let Φ be a satisfying assignment of φ. Add a b-arc from $q_{a^{m+j}}$ to q_ε if $\Phi(x_j) = 1$; add such a b-arc from $q_{a^{m+j}}$ instead to $q_{a^{m+n+1}}$ if $\Phi(x_j) = 0$.

As Φ is a satisfying assignment of φ, we can also define a mapping $\sigma_\Phi : C \to \{1, \ldots, n\}$ such that, for every $c \in C$, the variable $x_{\sigma_\Phi(c)}$ is contained in c and, moreover, $\Phi(x_{\sigma_\Phi(c)}) = 1$ if $c \in C_P$, while $\Phi(x_{\sigma_\Phi(c)}) = 0$ if $c \in C_N$. Intuitively speaking, σ_Φ (applied to $c \in C$) explains how clause c is satisfied by the assignment Φ. Now, in A, we add a b-arc from q_{a^i} to $q_{a^{m+j}}$ for $1 \leq i \leq m$ if $j = \sigma_\Phi(c_i)$. We have to show that A is consistent with X^+ and X^-. What is left to prove are the properties of the sample listed in items 3, 4, and 5.

Item 3: For $1 \leq i \leq m$, $q_{a^ib} = q_{a^{m+j}}$ for some $j \in \{1, \ldots, n\}$. Hence, q_{a^ib} is not an accepting state, nor is q_{a^iba}, as in particular $q_{a^{m+n+1}}$ is not accepting. By construction, $q_{a^ibb} = q_\varepsilon$ or $q_{a^ibb} = q_{a^{m+n+1}}$. In either case, $\delta(q_{a^ibb}, b) = q_\varepsilon$, i.e., q_{a^ibbb} is an accepting state. Consistency with items 4 and 5 immediately follow by the given construction of the b-arcs.

For the reverse direction, let A be a DFA that is consistent with X^+ and X^- and has $m+n+2$ states $q_{a^0}, \ldots q_{a^{m+n+1}}$, see Property (A). As $t = n+m+2$, we conclude that $q_\varepsilon = q_{a^0} = q_{a^{m+n+2}} = q_b$ is the only accepting state. In the graph representation of A, $q_{a^1}, \ldots, q_{a^m}$ are the *clause vertices* and $q_{a^{m+1}}, \ldots, q_{a^{m+n}}$ are the *variable vertices*. We can read off the assignment $\Phi_A(x_j)$ from the b-transitions emanating from $q_{a^{m+j}}$ for $j = 1, \ldots, n$ by the following prescription: If the only accepting state q_ε is reached, let $\Phi_A(x_j) = 1$, otherwise let $\Phi_A(x_j) = 0$.

Property (B) ensures that b-arcs from the clause vertices always lead to the variable vertices in the automaton graph. This means that each clause $c_i \in C$, $1 \leq i \leq m$, is satisfied by the assignment Φ_A as follows.

Clause $c_i \in C_P$ is satisfied by $x_j \in c_i$ if $q_{a^ib} = q_{a^{m+j}}$ due to $a^ibb \in X^+$, which means that $q_{a^ibb} = q_\varepsilon$ and hence $\Phi_A(x_j) = 1$.

Clause $c_i \in C_N$ is satisfied by $\overline{x_j} \in c_i$ if $q_{a^ib} = q_{a^{m+j}}$ because of $a^ibb, a^ibba \in X^-$, $a^ibbb \in X^+$, which means that $q_{a^ibb} = q_{a^{m+n+1}}$, which is not accepting, and hence $\Phi_A(x_j) = 0$.

As A is deterministic, clearly no contradictory variable assignments are possible, and the proof is complete. □

With the three lemmas above, we obtain the results presented in rows 1, 2, 3, and 5 of Table 2, that are summarized as follows.

Theorem 16. DFA Consistency *problem is NP-complete when* $t = 2$, $|I| = 2$, *or* $\ell = 2$, *and even when* $t\ell = 6$.

PROOF. Lemma 13 reduces SAT to a DFA CONSISTENCY instance where $t = 2$ and thus the DFA CONSISTENCY problem is NP-complete when $t = 2$. By the reduction from the 3-COLORING instance in Lemma 14 it follows that the DFA CONSISTENCY problem is NP-complete for $t = 3, \ell = 2$, which gives $t\ell = 6$. From the SAT reduction given in Lemma 15, our problem is NP-complete for $|I| = 2$. □

After these hardness results, we go to the next natural step of combining some of the above studied parameters. Also for this problem, it turns out that there are straightforward FPT algorithms when we use several parameters, but as before, we are able to show lower bounds to prove that these simple algorithms are best possible.

Theorem 17. DFA CONSISTENCY *is FPT when parameterized with* t *and* $|I|$*; it can be solved in time* $O^*(t^{t|I|})$.

PROOF. For each of the $t|I|$ different combinations of a state $q \in S$ and a symbol $a \in I$ we brute force guess among the t states where to go. This gives a total of $t^{t|I|}$ possibilities. For each such DFA A and for each state s in A we check if X^+ and X^- ends up in a disjoint set of states when using s as a start state. □

We next show that the running time of the simple FPT algorithms above is best possible. Note that by the classical reduction from 3-SAT to 3-COLORING, the number of vertices of the obtained 3-COLORING instance is linear in the number of clauses of the 3-SAT instance.

Theorem 18. DFA CONSISTENCY *cannot be solved in time* $O^*(t^{o(t|I|)})$ *unless ETH fails.*

PROOF. Through the standard reduction from 3-SAT to 3-COLORING it follows that a 3-COLORING instance on n vertices and m edges cannot be solved in time $O^*(2^{o(n+m)})$ unless ETH fails.

By the reduction of Lemma 14 we get an instance of DFA CONSISTENCY where $t = 3, |I| = n + m, \ell = 2$, and $c = 2m$. Any algorithm for DFA CONSISTENCY solving it in $O^*(t^{o(t|I|)})$ time will also solve 3-COLORING in $O^*(2^{o(n+m)})$ and ETH will fail. □

Although combining the parameters t and $|I|$ easily gives an FPT algorithm, we show next that a polynomial kernel is not expected with this combination.

Theorem 19. DFA CONSISTENCY *does not have a polynomial kernel when parameterized with both* t *and* $|I|$ *unless* NP $\subseteq$ coNP/poly.

PROOF. By Lemma 13 any CNF formula can be reduced to a a DFA CONSISTENCY instance where $t = 2$, $|I| = 3n + 1$, $\ell = 2n$, and $c = 6n + m + 3$ in polynomial time. If there existed a polynomial time algorithm that produced an equivalent instance of size polynomial in $t, |I|$, this would mean that the number of words in $X^+ \cup X^-$ is reduced. As DFA CONSISTENCY is NP-complete, there

exists a polynomial time reduction back to a SAT instance with n' variables and m' clauses where $n' + m'$ is polynomial in $t, |I|$. This would imply that the number of clauses in this CNF formula is bounded by a polynomial in n, and it is thus a polynomial kernel for SAT when parameterized by the number of variables. By Proposition 1 this implies that $NP \subseteq coNP/poly$. □

Next we turn to the parameter combination (t, c, ℓ), which again gives a trivial FPT algorithm whose running time seems unlikely to be improvable.

Theorem 20. DFA Consistency *is FPT when parameterized with t, c, and ℓ; it can be solved in time $O^*(t^{c\ell})$.*

Proof. The algorithm is simply brute force guessing the walk created by each word $\sigma \in X^+ \cup X^-$ from the start state s to the final state reached by σ. The total number of words is c and each word is of length at most ℓ. Consider a word $\sigma = a_1 a_2 \cdots a_\ell$. After reading i symbols, where $0 \leq i \leq \ell$ some state q_i is reached. From state q_i we read next symbol a_{i+1} and have t states to choose from where to go next. So the total number of DFAs enumerated is $t^{c\ell}$. For each such DFA A and for each state s in A check if X^+ and X^- ends up in a disjoint set of states. □

Corollary 21. DFA Consistency *is FPT when parameterized with c and ℓ; it can be solved in time $O^*((c\ell)^{c\ell})$.*

Proof. One DFA that is consistent with the sample can be constructed as the minimum-state DFA accepting (exactly) X^+. This DFA has at most $c\ell$ states, as it is not bigger than the so-called prefix-tree acceptor for X^+. Hence, Theorem 20 shows the claim. □

This might motivate to study an even coarser parameter, which is the sum of the lengths of all the words in $X^+ \cup X^-$, $s = ||X^+ \cup X^-|| = \sum\{|w| \mid w \in X^+ \cup X^-\}$. Notice that $s+1$ is a trivial upper bound on t, as the so-called prefix tree acceptor (PTA) A with $L(A) = X^+ \cup X^-$ has at most $s+1$ states; cf. [38]. Clearly, $c \leq s + 1$ (the empty word might play a special role) and $\ell \leq s$. It follows immediately that DFA Consistency is FPT when parameterized with s, since it can be solved in time $O^*(s^{s^2})$.

We can improve on this running time by the following observation: If there is some t-state DFA that is consistent with $X^+ \cup X^-$, then there is also a t-state DFA consistent with $X^+ \cup X^-$ that is obtained by state-mergings from the PTA for $X^+ \cup X^-$. As that PTA has at most $s + 1$ states and as the corresponding state-mergings can be described as mappings from $\{0, \ldots, s\}$ to $\{0, \ldots, s\}$, we can infer:

Corollary 22. DFA Consistency *is FPT when parameterized with s; it can be solved in time $O^*(s^s)$.*

Now we turn to negative results for these parameters, conditioned on plausible hypotheses.

Theorem 23. DFA Consistency *cannot be solved in time* $O^*(t^{o(c\ell)})$ *unless ETH fails.*

Proof. Through the standard reduction from 3-SAT to 3-Coloring it follows that 3-Coloring instance on n vertices and m edges can not be solved in time $O^*(2^{o(n+m)})$ unless ETH fails. By the reduction of Lemma 14 we get an instance of DFA Consistency, where $t = 3, |I| = n + m, \ell = 2$, and $c = 2m$. Any algorithm for the DFA Consistency problem solving the problem in $O^*(t^{o(c\ell)})$ time will also solve the 3-Coloring problem in $O^*(2^{o(m)})$ and ETH will fail. □

With the following two results, we give even stronger lower bounds for the parameter combination $(t, c, \ell, |I|)$.

Theorem 24. DFA Consistency *cannot be solved in time* $O^*(|I|^{o(|I|(t+c+\ell))})$ *unless ETH fails.*

Proof. From the SAT reduction given in Lemma 15 we reduce 3-SAT to an instance of DFA Consistency where $t \leq 3m+n+2$, $|I| = 2$, $\ell \leq 3m+n+2$, and $c \leq 13m + n + 4$. Unless ETH fails there exists no $O^*(2^{o(n+m)})$ time algorithm that solves a 3-SAT instance on n variables and m clauses. By the previous reduction, this means that there exists no $O^*(|I|^{o(|I|(t+c+\ell))})$ time algorithm for DFA Consistency unless ETH fails. □

Theorem 25. DFA Consistency *cannot be solved in time* $O^*((t\ell)^{o(t\ell(c+|I|))})$ *unless ETH fails.*

Proof. By the reduction of Lemma 14 we get an instance of DFA Consistency where $t = 3, |I| = n + m, \ell = 2$, and $c = 2m$. Unless ETH fails there exists no $O^*(2^{o(n+m)})$ time algorithm that solves a 3-Coloring instance on n variables and m clauses. By the previous reduction, this means that there exists no $O^*((t\ell)^{o(t\ell(c+|I|))})$ time algorithm for DFA Consistency unless ETH fails. □

We end this section by turning our attention to the parameter c. We do not know whether the problem is FPT with this parameter only, or when parameterized with both t and c. Could it be that DFA Consistency is NP-hard when $t = 2$ and c is a sufficiently large constant? We are able to answer this question partially with the following positive result.

Theorem 26. DFA Consistency *can be solved in polynomial time when* $t = 2$ *and* $c = 2$*, with no restriction on the alphabet size.*

Proof. When reasoning about 2-state DFAs, recall the basic structure of the induced mappings as presented in Table 3. We need the following properties and types of arguments frequently, referring to them as observations:

1. If there are both positive and negative examples, i.e., if $|X^+| \cdot |X^-| > 0$, then one state is accepting and one is not.

2. The problem is non-trivial, as there exist examples like $X^+ = \{a\}$, $X^- = \{aaa\}$, that have no 2-state DFA solution.
3. If there is both a positive example x^+ and a negative example x^- ending with some common suffix word ω, then for any 2-state DFA that solves the consistency problem, its transition function δ obeys that, on each letter a occurring in ω, δ_a is the identity or the parity checker.
4. If there is both a positive example x^+ and a negative example x^- such that for all letters a, $\#_a(x^+) \equiv \#_a(x^-) \pmod 2$, then for any 2-state DFA that solves the consistency problem, there is some letter a that is a synchronizing word.
5. If a 2-state automaton should profit from parity information on strings x^+, x^- leading into different states when starting in the same state, and if there are no synchronizing letters in x^+ nor in x^-, then clearly, for some non-empty subset P of letters, all and exactly those letters a in P induce parity checkers δ_a. This means that $\sum_{a\in P} \#_a(x^+) \not\equiv \sum_{a\in P} \#_a(x^-) \pmod 2$. Then, we can assume (without loss of generality) that $|P| = 1$. Namely, we could first eliminate all letters from P that occur an even number of times both in x^+ and in x^- or an odd number of times both in x^+ and in x^-. Then, pairs of different letters $a, b \in P$ with $\#_a(x^+) \not\equiv \#_a(x^-) \pmod 2$ and $\#_b(x^+) \not\equiv \#_b(x^-) \pmod 2$ can be eliminated from P without changing the validity of $\sum_{a\in P} \#_a(x^+) \not\equiv \sum_{a\in P} \#_a(x^-) \pmod 2$. We end up in the situation that $|P| = 1$ as claimed.

To prove the theorem, we describe the general structure of a polynomial-time algorithm, given one positive sample word x^+ and one negative sample word x^-, in Table 4. Every time the algorithm answers YES, there is a (different) construction justifying this answer. Details are given in the analysis that follows.

Let us first describe what to do in the different (positive) cases, where a 2-state DFA can be constructed, before arguing why the algorithm catches all situations to conclude a correct NO answer (the correctness of the YES answers is seen by the constructions).

Case (A) Why is there is a 2-state DFA for solving (x^+, x^-) in this situation? Two symmetric subcases occur: (A1) $\#_a(x^+) = 0$ and (A2) $\#_a(x^-) = 0$. We only consider (A1) in the following. Due to Step 1, $\#_a(x^-) > 0$. The first automaton given in Table 5 rejects any string containing a, and hence rejects x^-, but accepts any string not containing a, and hence accepts x^+, as required. For this DFA, all letters but a induce the identity, while δ_a is a p-synchronizer, with p being the accepting state. Clearly, in this case there is also a symmetric but completely different solution, having δ_a as a q-synchronizer, with q as the accepting state.

Case (B) Again, we have two symmetric subcases: (B1) $\#_a(x^+) \equiv 0 \pmod 2$ and $\#_a(x^-) \equiv 1 \pmod 2$ or (B2) $\#_a(x^+) \equiv 1 \pmod 2$ and $\#_a(x^-) \equiv 0 \pmod 2$. We only consider (B1) in the following. The second automaton in Table 5 counts modulo two and hence classifies x^+ and x^- as required. Here , all letters but a induce the identity, while δ_a is a parity checker.

1. For all letters $a \in I$ with $\#_a(x^+) = \#_a(x^-) = 0$, remove a from I.
   If now $I = \emptyset$, then $x^+ = x^- = \varepsilon$, so return NO.
2. If there is a letter $a \in I$ that satisfies $\#_a(x^+) \cdot \#_a(x^-) = 0$ (Case A) or $\#_a(x^+) \not\equiv \#_a(x^-) \pmod 2$ (Case B), then answer YES.
3. If $x^+ = x^-$, return NO.
4. Otherwise, there exist decompositions $x^+ = \alpha^+ \cdot a^+ \cdot \omega$, $x^- = \alpha^- \cdot a^- \cdot \omega$ for some $\alpha^+, \alpha^-, \omega \in I^*$ and two different letters a^+, a^-.
5. For each letter a, consider now the case that a is the last occurrence of some letter in x^+ such that some DFA solving the consistency problem is a synchronizing letter. Without loss of generality, assume that δ_a is a p-synchronizer.
   (a) Let $x^+ = \alpha^+ a \omega^+$, with ω^+ containing no synchronizing letter.
   (b) Let $x^- = \alpha^- a \omega^-$, with $\#_a(\omega^-) = 0$.
   (c) If there is some $b \in I$ in ω^- that is not in ω^+, then answer YES (Case C).
   (d) If there is some $b \in I$ that occurs both in ω^+ and in ω^-, with $\#_b(x^+) \not\equiv \#_b(x^-) \pmod 2$, then answer YES (Case D).

Table 4: The main structure of an algorithm for deciding if there exists a 2-state DFA that accepts string x^+ and rejects string x^-.

Notice that due to Observation 5, there is no other way of profiting from parity checks (without synchronizing letters).

Situation before the FOR loop First notice that in this case, $|x^+| \cdot |x^-| > 0$, as otherwise either both strings are empty (excluded by Step 1) or one is empty and the other one not, which has been caught by Step 2. Hence, all letters that occur in x^+ occur in x^- and vice versa, comprising the alphabet I. By Observation 4, in order to separate x^+ and x^-, there must exist at least one letter, say a, that is a synchronizing word. This synchronizing letter a occurs both in x^+ and in x^-. Going through all possibilities, we can now assume that a is the last letter in x^+ that is synchronizing, and we focus on the last occurrences of a in x^+ and in x^-.

Case (C) Recall that δ_a is a p-synchronizer. Let δ_b be a q-synchronizer, all other letters yielding identity mappings. This gives another YES-answer to the consistency problem, as the synchronization and final state selection can be adapted to the situation; see Table 5. Upon reading the last occurrence of a in x^+, our DFA enters state p and stays there when reading the end ω^+. As b is the last occurrence of a synchronizing letter in x^-, our DFA ends in q upon reading x^-.

Case (D) Notice that δ_a is a p-synchronizer. Let δ_b be a parity checker. Let all letters but a, b induce identities. This gives another YES-answer to the consistency problem. After the last reset due to reading a, we start a parity check on b. For instance, if b occurs an even number of times in ω^+ but an odd number of times in ω^-, we get the picture shown in Table 5.

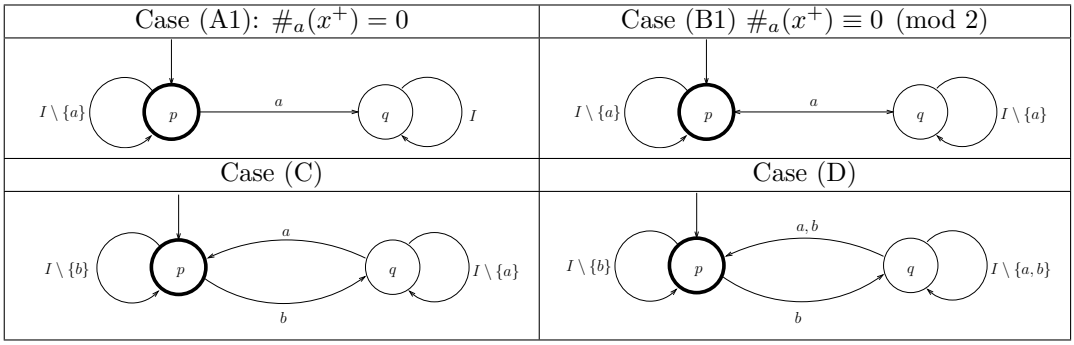

Table 5: Justifying the four YES answers of Algorithm 4.

> Notice that due to Observation 5, there is no other way of profiting from parity checks (without synchronizing letters, a case that occurs in the end pieces ω^+ and ω^-).

The given constructions justify all YES answers of our algorithm. Why does the algorithm catch all cases? Consider a YES-instance (x^+, x^-). Clearly, allowing for a solution means that $x^+ \neq x^-$ (see Step 1); in particular, at least one of the strings is non-empty. If only one is non-empty, say, x^+, then turning at least one letter from x^+ into a q-synchronizer tells x^+ and $x^- = \varepsilon$ apart. Otherwise, consider a hypothetical DFA A with initial state p and second state q and state transition function δ, separating x^+ and x^-. We can differentiate two cases for A: Either, (1) A acts on no letter as a synchronizer, or (2) A acts on some letter, say on a, as a synchronizer, i.e., δ_a is either a p- or a q-synchronizer.

Let us start considering Case (1). Then, according to Observation 3, for each letter a, δ_a is either the identity or a parity checker. As A distinguishes between x^+ and x^-, not all letters a could yield δ_a as the identity. Due to Observation 5, we can assume that A acts on all letters but one, say a, as the identity. As A distinguishes between x^+ and x^-, $\#_a(x^+) \not\equiv \#_a(x^-) \pmod 2$, which is the situation caught in Case B of our algorithm.

In Case (2), let us assume that, w.l.o.g., δ_a is a synchronizer in A. If a does not occur in x^+ or x^- at all, A could have the structure sketched in Case A. Otherwise, a occurs in x^+ and in x^-, and we could then consider the last occurrence of any letter in x^+ (that we again call a for simplicity) such that A has δ_a as a synchronizer. Hence, there exist decompositions $x^+ = \alpha^+ \cdot a \cdot \omega$, $x^- = \alpha^- \cdot a \cdot \omega$ for some $\alpha^+, \alpha^-, \omega \in I^*$. Two subcases may occur: (a) δ_a is a p-synchronizer, or (b) δ_a is a q-synchronizer. The arguments in (a) and (b) are completely symmetric, so let us focus on Subcase (a). So, after reading the either of the initial parts $\alpha^+ a$ and $\alpha^- a$, A will be back in state p. Hence, we can repeat our previous analysis, as A has to separate ω^+ from ω^-, as well, knowing that a does not occur in neither ω^+ nor ω^-. Cases (C) and (D) of our algorithm

are basically covering the previous Cases (A) and (B) on the shortened instance (ω^+, ω^-). This concludes the proof of the validity of our algorithm. □

5. Other related problems

The two main problems we investigated so far have quite a number of interesting variants that we want to mention in this section. This also offers further natural parameterizations. We focus on SW-variants, most of which are computationally harder than SW. The new hardness reductions are then mostly from the following problem.

Problem: DFA Intersection
Input: q DFAs $A_1, \ldots, A_q$ with the same input alphabet I, and an integer $m \geq 0$.
Question: Is there a string of length m that is accepted by all q automata?

This is related to the probably most famous PSPACE-complete problem dealing with DFAs, DFA Intersection Nonemptiness. With parameter q alone, and also for the combined parameter $(q, |I|)$, the problem becomes W[τ]-hard for any level τ of the W-hierarchy, even when $|I| = 2$, while it is "only" W[1]-hard for the combined parameter (q, m). These results were obtained by Wareham [73], and mentioned in the textbook [22] with the variation there asking for a string of length at least m.

Based on the assumption that some partial information on the current state of a DFA might be known, formalized by a set of states Q, the Q-Synchronizing Word problem was introduced, called Subsynity in [69], which we briefly conduct a multi-parameter analysis of in this section.

Problem: Q-Synchronizing Word (Q-SW)
Input: A DFA $A = (S, I, \delta, s_0, F)$, a set of states $Q \subseteq S$, and an integer k.
Question: Is there a word $x \in I^*$ of length at most k such that $|\delta^*(Q, x)| = 1$?

If such a word x exists, then we say that it *synchronizes* all states from Q, and call it a *Q-synchronizing word*. For this problem, we study the natural parameters $t = |S|$, $|I|$, k, and $|Q|$, and combinations of these. Our findings are summarized in Table 6.

Before we go on to justify these results, let us mention that even the question of the existence (without a length requirement) of a Q-synchronizing word is known to be PSPACE-complete. Hence we could even ask for the parameterized complexity of the existence variant, which would mean to focus on the parameters t, $|I|$ and $|Q|$. As the corresponding results would be very similar to the ones from Table 6, we refrain from explicitly stating them. From [73] and the reduction from DFA Intersection Nonemptiness given in [61] that shows PSPACE-hardness of this problem, we can rather immediately deduce the last two rows of Table 6.

Parameter	Parameterized Complexity
t	FPT with running time $O^*(2^t)$
$\|I\|$	PSPACE-complete for $\|I\| = 2$
k	W[2]-hard
k and $\|I\|$	FPT with running time $O^*(\|I\|^k)$
$\|Q\|$ and k	W[1]-hard
$\|Q\|$ and $\|I\|$	W[τ]-hard for all τ

Table 6: A summary of the results on Q-SYNCHRONIZING WORD.

Theorem 27. *Q-SYNCHRONIZING WORD, parameterized by $|Q|$ and k, is W[1]-hard.*
Q-SYNCHRONIZING WORD, parameterized by $|Q|$ and $|I|$, is W[τ]-hard for all τ.

The only technical problem is that in the parameterized analogue DFA INTERSECTION, the length parameter m is an exact bound, while the length parameter k is an upper bound. However, by adding a sequence of m "new" states starting from some new Q-state s_0, we can enforce the constructed DFA to have a word of length at least $m + 1$ as its shortest Q-synchronizing word. As a further technical detail, the reduction given in [61] will increase the word length by one, as well as the size of the input alphabet; also $|Q| = q + 1$.

In addition, Sandberg refers to [45] for constructing automata with exponentially long synchronizing words, even in the unary case. Conversely, our parameterized complexity results for parameters t, $|I|$, and k transfer from SYNCHRONIZING WORD to this more general setting. These are presented in the first four rows of Table 6. We should add a word on the result for parameter $|I|$. It is known that the question of the existence of a word accepted by q DFAs $A_1, \ldots, A_q$ is PSPACE-complete even on unary input alphabets. Conversely, the standard product automaton construction can be used to obtain an automaton $A = A_1 \times \cdots \times A_q$ such that $w \in L(A)$ if and only if $w \in L(A_i)$ for any $1 \leq i \leq q$. Hence, by a classical pumping-style argument (relying on the pigeon-hole principle), if there is a word $w \in L(A)$, then there is also a word in $L(A)$ of length at most equal to the number of states of A. Clearly, if A_i has t_i many states, then A has $T = t_1 \times \cdots \times t_q$ many states. T is not polynomial in the size of the input of the DFA INTERSECTION instance, which is instead polynomial in $t = t_1 + \cdots + t_q$. However, it is possible to write down T using some number of bits polynomial in t. Therefore, by using basically the same reduction as sketched above (apart from not having to add m "new" states starting from some new Q-state s_0), we can reduce the question whether the intersection of the languages of q given DFAs $A_1, \ldots, A_q$ on unary input alphabets is non-empty (which is PSPACE-complete) to the the following question: Given a state subset Q of a DFA and some integer k, does the DFA have some Q-synchronizing word of length at most k? In our reduction, we choose $k = T$ as argued before. For contrast, notice that the reduction given in [61] increases the size of the input alphabet, as it adds one more special symbol to be read in

the very end.

We now turn to related problems on Mealy machines, i.e., deterministic finite automata with an additional output function $\lambda : S \times I \to O$ (that can be easily extended to $\lambda : S \times I^* \to O^*$). We first present the necessary notions.

- A *homing sequence* for a Mealy machine $A = (S, I, O, \delta, \lambda, s_0, F)$ is a word $x \in I^*$ such that there exists a function f mapping $\lambda^*(s_0, x)$ to the state $\delta^*(s_0, x)$. In other words, the output on x uniquely determines the state that is reached. If $|O| = 1$, a homing sequence is nothing else but a synchronizing word.

- A *distinguishing sequence* for a Mealy machine $A = (S, I, O, \delta, \lambda, s_0, F)$ is a word $x \in I^*$ such that, for any states $s, s' \in S$, $\lambda^*(s, x) = \lambda^*(s', x)$ implies $s = s'$. In other words, the output on x uniquely determines the state that the machine started in. A *preset distinguishing sequence* is a word $x \in I^*$ such that, for any state $s' \in S$, $\lambda^*(s_0, x) = \lambda^*(s', x)$ implies $s_0 = s'$. In other words, the output on x verifies that the state that the machine started in has indeed been the initial state s_0.

While finding a homing sequence of length at most k is NP-complete (as with synchronizing words), even deciding whether a (preset) distinguishing sequence exists is PSPACE-complete, as shown in [45]. We refrain from giving detailed definitions of the problems Homing Sequence and (Preset) Distinguishing Sequence, both having a Mealy machine plus an integer k (an upper bound on the sequence length) as their inputs.

Almost all results that we presented for Synchronizing Word also work for Homing Sequence. More precisely, the hardness results simply translate, as the Mealy machine could have a unary output alphabet, which means that the output sequence does not reveal any structure of the automaton at all, as the length of the output word equals the length of the input word by definition of a Mealy machine. The (trivial) enumeration result concerning the parameter $(|I|, k)$ is also true for Mealy machines. The analogue of Theorem 10 is true as a classification result, but the running times of the known algorithms (see, e.g., [61]) are far worse, as those algorithms cycle through at most t sets of t-element sets (reflecting partitions and similar structures of the set of states). Therefore, the analogue of Theorem 12 is a less convincing result in this context. So, for this related question, there is even more room for improving the straightforward FPT algorithms.

In the PSPACE-hardness proofs of [45], the size of the output alphabet does not matter (possibly apart from the clearly simpler case when $|O| = 1$). For distinguishing sequences, the following parameter is of interest, called *initial uncertainty* (with respect to an input word w): $\pi(w)$ is a partition on the state set S defined by the output equivalence, i.e., the equivalence classes of s and s' coincide iff $\lambda(s, w) = \lambda(s', w)$. We might deduce from this as a parameter $IU_{\min} = \min\{|\pi(a)| \mid a \in I\}$ or $IU_{\max} = \max\{|\pi(a)| \mid a \in I\}$. Examining the proof in [45], we can deduce from the mentioned results of Wareham:

Corollary 28. (Preset) Distinguishing Sequence *is W[1]-hard when parameterized with* $IU_{\min}$ *(or alternatively* $IU_{\max}$*) and* k*.*

Q-Synchronizing Word *is W[*τ*]-hard for all* τ *when parameterized with* $IU_{\min}$ *(or alternatively* $IU_{\max}$*) and* $|I|$*. This is also true for the parameters* $IU_{\sum}$ *and* $|I|$*, where* $IU_{\sum} = \sum\{|\pi(a)| \mid a \in I\}$*.*

As in the previous case, the published reductions make it clear (or can be adapted so) that the addition of the parameter k does not harm the hardness results. Usually, only the existence of a (preset) distinguishing sequence is asked. Also, Table 6 nearly completely transfers to (Preset) Distinguishing Sequence with the more classical parameters; only, for the parameter t, the FPT result needs a bit more care, as all partitions of the state set have to be considered.

As a final remark to this section, let us mention that one could also consider variants and generalizations of Mealy machines can be considered, like Moore machines, subsequential machines, deterministic generalized sequential machines and so forth. Most of the results will transfer to such situations.

6. Conclusion and questions for future research

With this paper, we started the first steps in the multi-parameter analysis of several DFA (and Mealy machine) problems. We now indicate several lines of future research.

- Does Synchronizing Word belong to the class W[2] with respect to parameter k? The standard approach of constructing a multi-tape Turing machine seems to be problematic, as the Turing machine that guesses a synchronizing word and verifies it using one tape per DFA state would have a state transition table that is not polynomial in the input DFA, see [12]. For the model-checking approach (leading to the A-hierarchy; see [30]), we could suggest the following formula:

  $$\exists a_1, \ldots, a_k \in I \, \exists f \in S \, \forall s_0 \in S \exists s_1, \ldots, s_k \in S : \bigwedge_{0 \leq i < k} \delta(s_i, a_{i+1}) = s_{i+1} \wedge s_k = f$$

  Strictly speaking, this expresses the DFA property of having a synchronizing word of length equal to k, but it is not hard to adapt this to a formula that expresses the question whether a synchronizing word of length at most k exists. This puts this problem into $A[2]$, but not necessarily into $W[2]$. Is this problem hard for $A[2]$? Conversely, these problems might be hard for some higher level of the W hierarchy, which would be also quite an interesting result, as only few natural hard problems are known for these levels; see [13].

- Is DFA Consistency FPT when parameterized with c, or with both c and t?

- Does DFA Consistency have a polynomial kernel with parameter (t, c, ℓ)?

- For DFA Consistency, there are also natural refinements of the parameters we introduced above. For instance, $s^+ = ||X^+||$ and $s^- = ||X^-||$ would make perfect sense, as would ℓ^+, ℓ^-, c^+, and c^- that could be defined in analogy. The reasoning above leading to Corollary 22 (using prefix tree acceptors) shows that the parameter s^+ alone is sufficient to obtain an FPT membership result. By considering complemented automata, it is clear that also s^- alone would suffice. This gives even more possible combinations, for instance, what about the parameter (ℓ^+, c^-)? Hence, these refined parameterizations lead to quite a number of yet unsettled questions, including the quest for polynomial size kernels.

- One possible way out for hard problems is to consider sub-families of (in this case) regular languages or of DFAs; see [66]. In this context, it is interesting to observe that the optimization problem SW remains NP-hard for aperiodic regular languages, see [66].

- Alternatively, one could also to study other devices than DFAs that characterize the regular languages. For instance, what about posing our questions for nondeterministic finite automata or regular expressions? As Sandberg mentions [61], even deciding the existence of a synchronizing word for NFAs becomes PSPACE-complete.

- There are other natural variants of DFA Consistency. Angluin showed that Regular Expression Consistency is even hard for regular expressions of a very simple structure, without any nested Kleene stars [3], a class which sits very low in the famous star height hierarchy; see [19]. Also, the length of the regular expression, which is the most natural measure of descriptional complexity of regular expressions, is linearly dependent on the number of variables of the SAT instance she uses for reduction. The other natural parameter, the alphabet size, is kept down to two in that reduction. In view of the fact that for many applications, regular expressions are considered as important as DFAs, this could give an interesting line of research.

- What could be further natural parameters for problems on regular languages? Discovering these as possible sources of hardness could be a very fruitful line of research for both problem classes that we considered in this paper. Thoughts from the classical theory of Formal Languages could become very helpful in this quest, for instance, from Descriptional Complexity [39, 40] or also ideas coming from algebraic parameters (say, properties of the syntactic monoid); aperiodicity was already mentioned above.

- It might be interesting to explore other hard learning problems from the viewpoint of Parameterized Complexity; the problems most related to ours can be found in [42].

- Our reductions (for instance, the one from Hitting Set) might also inspire new ways of looking at the DFA problems that we presented. Continuing with the mentioned example reduction, there exists also a kind of dual problem, namely Max Cover (see [25]). In our context, this might mean that we want to know the largest set of states that we can synchronize by a sequence of a given (fixed) length. Alternatively, we might want to minimize the number of states that we fail to synchronize. Or, we want to minimize the number of states that a given DFA could still be in after having read a word of a certain length. By these simple thoughts, we already obtain three new rather natural parameters that one could look into when studying Synchronizing Word.

We conclude by noting the following.

Remark 2. In the conference version of this paper, we posed as the very first question whether Synchronizing Word has a polynomial kernel with parameter t? In the meantime, this question has been resolved in a negative sense in [70]. Unless an unlike collaps of the polynomial hierarchy occurs, this problem has no polynomial-size kernels. This contrasts their findings on a related road coloring problem.

Acknowledgements

We thank Dana Angluin, Colin de la Higuera, and Sicco Verwer for recovering the traces left in references [1, 38]. We thank the reviewer for the helpful comments.

[1] N. Abe and M. K. Warmuth. On the computational complexity of approximating distributions by probabilistic automata. *Machine Learning*, 9:205–260, 1992.

[2] N. Alon, D. Moshkovitz, and S. Safra. Algorithmic construction of sets for k-restrictions. *ACM Transactions on Algorithms*, 2(2):153–177, 2006.

[3] D. Angluin. On the complexity of minimum inference of regular sets. *Information and Control (now Information and Computation)*, 39:337–350, 1978.

[4] V. Arvind, J. Köbler, and W. Lindner. Parameterized learnability of juntas. *Theoretical Computer Science*, 410(47-49):4928–4936, 2009.

[5] A. Barecka and W. Charatonik. The parameterized complexity of chosen problems for finite automata on trees. In A. H. Dediu, S. Inenaga, and C. Martín-Vide, editors, *Language and Automata Theory and Applications - 5th International Conference, LATA*, volume 6638 of *LNCS*, pages 129–141. Springer, 2011.

[6] M. V. Berlinkov. Approximating the minimum length of synchronizing words is hard. In F. M. Ablayev and E. W. Mayr, editors, *Computer Science - Theory and Applications, 5th International Computer Science Symposium in Russia, CSR*, volume 6072 of *LNCS*, pages 37–47. Springer, 2010.

[7] M. V. Berlinkov. O pogreshnosti polinonominal'nogo vytchisleniya optimal'noj raskaski grafa v sinchronoziruemyj aftomat. *Prikladnaya Diskretnaya Matematika*, pages 49–72, 2011.

[8] H. Björklund and W. Martens. The tractability frontier for NFA minimization. *Journal of Computer and System Sciences*, 78(1):198–210, 2012.

[9] M. M. F. Bugalho and A. L. Oliveira. Inference of regular languages using state merging algorithms with search. *Pattern Recognition*, 38(9):1457–1467, 2005.

[10] C. Calabro, R. Impagliazzo, and R. Paturi. The complexity of satisfiability of small depth circuits. In J. Chen and F. V. Fomin, editors, *Parameterized and Exact Computation, 4th International Workshop, IWPEC*, volume 5917 of *LNCS*, pages 75–85. Springer, 2009.

[11] J. Černý. Poznámka k homogénnym experimentom s konečnými automatmi. *Matematicko-fyzikálny časopis*, 14(3):208–216, 1964.

[12] M. Cesati. The Turing way to parameterized complexity. *Journal of Computer and System Sciences*, 67:654–685, 2003.

[13] J. Chen and F. Zhang. On product covering in 3-tier supply chain models: Natural complete problems for W[3] and W[4]. *Theoretical Computer Science*, 363(3):278–288, 2006.

[14] Y.-F. Chen, A. Farzan, E. M. Clarke, Y.-K. Tsay, and B.-Y. Wang. Learning minimal separating DFA's for compositional verification. In S. Kowalewski and A. Philippou, editors, *Tools and Algorithms for the Construction and Analysis of Systems, 15th International Conference, TACAS*, volume 5505 of *LNCS*, pages 31–45. Springer, 2009.

[15] K. Chmiel and A. Roman. COMPAS - A computing package for synchronization. In M. Domaratzki and K. Salomaa, editors, *Implementation and Application of Automata - 15th International Conference, CIAA 2010*, volume 6482 of *LNCS*, pages 79–86. Springer, 2011.

[16] C. Costa Florêncio and H. Fernau. On families of categorial grammars of bounded value, their learnability and related complexity questions. *Theoretical Computer Science*, 452:21–38, 2012.

[17] C. Costa Florêncio and S. Verwer. Regular inference as vertex coloring. In N. H. Bshouty, G. Stoltz, N. Vayatis, and T. Zeugmann, editors, *Algorithmic Learning Theory ALT*, volume 7568 of *LNCS/LNAI*, pages 81–95. Springer, 2012.

[18] F. Coste and J. Nicolas. Regular inference as a graph coloring problem. In *Workshop on Grammatical Inference, Automata Induction, and Language Acquisition (ICML'97), Nashville, TN*, 1997.

[19] F. Dejean and M. P. Schützenberger. On a question of Eggan. *Information and Control (now Information and Computation)*, 9(1):23–25, 1966.

[20] E. D. Demaine, S. Eisenstat, J. Shallit, and D. A. Wilson. Remarks on separating words. In M. Holzer, M. Kutrib, and G. Pighizzini, editors, *Descriptional Complexity of Formal Systems, DCFS*, volume 6808 of *LNCS*, pages 147–157. Springer, 2011.

[21] R. G. Downey, P. A. Evans, and M. R. Fellows. Parameterized learning complexity. In *Proc. Sixth Annual ACM Conference on Computational Learning Theory, COLT*, pages 51–57. ACM Press, 1993.

[22] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1999.

[23] R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

[24] D. Eppstein. Reset sequences for monotonic automata. *SIAM Journal on Computing*, 19(3):500–510, 1990.

[25] U. Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45:634–652, 1998.

[26] M. R. Fellows and H. Fernau. Facility location problems: A parameterized view. *Discrete Applied Mathematics*, 159:1118–1130, 2011.

[27] M. R. Fellows, S. Gaspers, and F. A. Rosamond. Parameterizing by the number of numbers. *Theory of Computing Systems*, 50(4):675–693, 2012.

[28] M. R. Fellows, B. M. P. Jansen, and F. A. Rosamond. Towards fully multivariate algorithmics: Parameter ecology and the deconstruction of computational complexity. *European Journal of Combinatorics*, 34(3):541–566, 2013.

[29] H. Fernau, P. Heggernes, and Y. Villanger. A multivariate analysis of some DFA problems. In A.-H. Dediu, C. Martín-Vide, and B. Truthe, editors, *Language and Automata Theory and Applications, LATA*, volume 7810 of *LNCS*, pages 275–286. Springer, 2013.

[30] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer, 2006.

[31] F. V. Fomin and D. Kratsch. *Exact Exponential Algorithms*. Texts in Theoretical Computer Science. Springer, 2010.

[32] L. Fortnow and R. Santhanam. Infeasibility of instance compression and succinct PCPs for NP. In C. Dwork, editor, *ACM Symposium on Theory of Computing, STOC*, pages 133–142. ACM, 2008.

[33] M. R. Garey and D. S. Johnson. *Computers and Intractability.* New York: Freeman, 1979.

[34] M. Gerbush and B. Heeringa. Approximating minimum reset sequences. In M. Domaratzki and K. Salomaa, editors, *Implementation and Application of Automata – 15th International Conference, CIAA 2010*, volume 6482 of *LNCS*, pages 154–162. Springer, 2011.

[35] E. M. Gold. Language identification in the limit. *Information and Control (now Information and Computation)*, 10:447–474, 1967.

[36] E. M. Gold. Complexity of automaton identification from given data. *Information and Control (now Information and Computation)*, 37:302–320, 1978.

[37] M. Heule and S. Verwer. Exact DFA identification using SAT solvers. In J. M. Sempere and P. García, editors, *Grammatical Inference: Theoretical Results and Applications, 10th International Colloquium, ICGI*, volume 6339 of *LNCS*, pages 66–79. Springer, 2010.

[38] C. de la Higuera. *Grammatical inference. Learning automata and grammars.* Cambridge University Press, 2010.

[39] M. Holzer and M. Kutrib. Descriptional complexity — an introductory survey. In C. Martín-Vide, editor, *Scientific Applications of Language Methods*, volume 2 of *Mathematics, Computing, Language, and Life: Frontiers in Mathematical Linguistics and Language Theory*, pages 1–58. Imperial College Press, 2010.

[40] M. Holzer and M. Kutrib. Descriptional and computational complexity of finite automata - a survey. *Information and Computation*, 209(3):456–470, 2011.

[41] R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.

[42] M. Kearns and L. Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *Journal of the ACM*, 41:67–95, 1994.

[43] Z. Kohavi. *Switching and Finite Automata Theory.* McGraw-Hill, 1970.

[44] K. J. Lang, B. A. Pearlmutter, and R. A. Price. Results of the Abbadingo One DFA learning competition and a new evidence-driven state merging algorithm. In V. Honavar and G. Slutzki, editors, *Grammatical Inference, 4th International Colloquium, ICGI*, volume 1433 of *LNCS*, pages 1–12. Springer, 1998.

[45] D. Lee and M. Yannakakis. Testing finite state machines: State identification and verification. *IEEE Transactions on Computers*, 43:306–320, 1994.

[46] M. Li and U. V. Vazirani. On the learnability of finite automata. In D. Haussler and L. Pitt, editors, *Proceedings of the First Annual Workshop on Computational Learning Theory, COLT*, pages 359–370. ACM/MIT, 1988.

[47] D. Lokshtanov, D. Marx, and S. Saurabh. Lower bounds based on the Exponential Time Hypothesis. *EATCS Bulletin*, 105:41–72, 2011.

[48] A. Mateescu and A. Salomaa. Many-valued truth functions, Černý's conjecture and road coloring. *EATCS Bulletin*, 68:134–150, 1999.

[49] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006.

[50] R. Niedermeier. Reflections on multivariate algorithmics and problem parameterization. In J.-Y. Marion and T. Schwentick, editors, *27th International Symposium on Theoretical Aspects of Computer Science (STACS 2010)*, volume 5 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 17–32. Schloss Dagstuhl—Leibniz-Zentrum für Informatik, 2010.

[51] A. L. Oliveira and J. P. M. Silva. Efficient algorithms for the inference of minimum size DFAs. *Machine Learning*, 44(1/2):93–119, 2001.

[52] J. Olschewski and M. Ummels. The complexity of finding reset words in finite automata. In P. Hlinený and A. Kucera, editors, *Mathematical Foundations of Computer Science 2010, 35th International Symposium, MFCS*, volume 6281 of *LNCS*, pages 568–579. Springer, 2010.

[53] L. Pitt and M. K. Warmuth. The minimum consistent DFA problem cannot be approximated within any polynomial. *Journal of the ACM*, 40:95–142, 1993.

[54] I. T. Podolak, A. Roman, and D.z Jedrzejczyk. Application of hierarchical classifier to minimal synchronizing word problem. In L. Rutkowski, M. Korytkowski, R. Scherer, R. Tadeusiewicz, L. A. Zadeh, and J. M. Zurada, editors, *Artificial Intelligence and Soft Computing - 11th International Conference, ICAISC, Part I*, volume 7267 of *LNCS*, pages 421–429. Springer, 2012.

[55] M. Praveen and K. Lodaya. Parameterized complexity results for 1-safe Petri nets. In J.-P. Katoen and B. König, editors, *Concurrency Theory - 22nd International Conference, CONCUR*, volume 6901 of *LNCS*, pages 358–372. Springer, 2011.

[56] A. Roman. New algorithms for finding short reset sequences in synchronizing automata. *World Academy of Science, Engineering and Technology*, 7:573–577, 2007.

[57] A. Roman. Genetic algorithm for synchronization. In A. Horia Dediu, A.-M. Ionescu, and C. Martín-Vide, editors, *Language and Automata Theory and Applications, Third International Conference, LATA*, volume 5457 of *LNCS*, pages 684–695. Springer, 2009.

[58] A. Roman. The NP-completeness of the Road Coloring Problem. *Information Processing Letters*, 111(7):342–347, 2011.

[59] A. Roman. P-NP threshold for synchronizing road coloring. In A. Horia Dediu and C. Martín-Vide, editors, *Language and Automata Theory and Applications - 6th International Conference, LATA*, volume 7183 of *LNCS*, pages 480–489. Springer, 2012.

[60] I. K. Rystsov. Polynomial complete problems in automata theory. *Information Processing Letters*, 16(3):147–151, 1983.

[61] S. Sandberg. Homing and synchronizing sequences. In M. Broy, B. Jonsson, J.-P. Katoen, M. Leucker, and A. Pretschner, editors, *Model-Based Testing of Reactive Systems*, volume 3472 of *LNCS*, pages 5–33. Springer, 2005.

[62] F. Stephan, R. Yoshinaka, and T. Zeugmann. On the parameterised complexity of learning patterns. In E. Gelenbe, R. Lent, and G. Sakellari, editors, *Computer and Information Sciences II - 26th International Symposium on Computer and Information Sciences, ISCIS*, pages 277–281. Springer, 2011.

[63] A. N. Trahtman. An efficient algorithm finds noticeable trends and examples concerning the Černý conjecture. In R. Kralovic and P. Urzyczyn, editors, *Mathematical Foundations of Computer Science, MFCS*, volume 4162 of *LNCS*, pages 789–800. Springer, 2006.

[64] A. N. Trahtman. An algorithm for road coloring. In C. S. Iliopoulos and W. F. Smyth, editors, *Combinatorial Algorithms - 22nd International Workshop, IWOCA*, volume 7056 of *LNCS*, pages 349–360. Springer, 2011.

[65] A. N. Trahtman. Modifying the upper bound on the length of minimal synchronizing word. In O. Owe, M. Steffen, and J. A. Telle, editors, *Fundamentals of Computation Theory - 18th International Symposium, FCT*, volume 6914 of *LNCS*, pages 173–180. Springer, 2011.

[66] A. Trakhtman. The Černý conjecture for aperiodic automata. *Discrete Mathematics & Theoretical Computer Science*, 9(2):3–10, 2007.

[67] A. Trakhtman. The road coloring problem. *Israel Journal of Mathematics*, 172:51–60, 2009.

[68] M. V. Volkov. Synchronizing automata and the Černý conjecture. In C. Martín-Vide, F. Otto, and H. Fernau, editors, *Language and Automata Theory and Applications, Second International Conference, LATA*, volume 5196 of *LNCS*, pages 11–27. Springer, 2008.

[69] V. Vorel. Subset synchronization of transitive automata. In Z. Ésik and Z. Fülöp, editors, *Proceedings 14th International Conference on Automata and Formal Languages, AFL*, volume 151 of *EPTCS*, pages 370–381, 2014.

[70] V. Vorel and A. Roman. Parameterized complexity of synchronization and road coloring. Technical Report abs/1403.4749, ArXive, 2014.

[71] P. J. Walker. *Synchronizing Automata and a Conjecture of Černý*. PhD thesis, University of Manchester, UK, 2008.

[72] H. T. Wareham. *Systematic Parameterized Complexity Analysis in Computational Phonology*. PhD thesis, Department of Computer Science, University of Victoria, Canada, 1998.

[73] H. T. Wareham. The parameterized complexity of intersection and composition operations on sets of finite-state automata. In S. Yu and A. Păun, editors, *Implementation and Application of Automata, 5th CIAA 2000*, volume 2088 of *LNCS*, pages 302–310. Springer, 2001.