Synchronizing Words and Monoid Factorization: A Parameterized Perspective

Jens Bruchertseifer and Henning Fernau(✉)

Fachber. 4 – Abteilung Informatikwissenschaften, Universität Trier, 54286 Trier, Germany
{s4jebruc,fernau}@uni-trier.de

Abstract. The concept of a synchronizing word is a very important notion in the theory of finite automata. We consider the associated decision problem to decide if a given DFA possesses a synchronizing word of length at most k, where k is the standard parameter. We show that this problem DFA-SW is equivalent to the problem MONOID FACTORIZATION introduced by Cai, Chen, Downey and Fellows. Apart from the known W[2]-hardness results, we show that these problems belong to A[2], W[P] and WNL. This indicates that DFA-SW is not complete for any of these classes and hence, we suggest a new parameterized complexity class W[Sync] as a proper home for these (and more) problems.

Keywords: Synchronizing word · Deterministic Finite Automaton (DFA) · Parameterized complexity

1 Introduction

Černý's conjecture is arguably the most famous open combinatorial problem concerning deterministic finite automata (DFA), somehow dating back to [7]. Recently, a particular Special Issue was dedicated to this conjecture being around for more than five decades; see [29]. This Special Issue also contains an English translation of Černý's paper [8]. The key notion is that of a synchronizing word. A word x is called *synchronizing* for a DFA A, if there is a state s_f, also called the *synchronizing state* of A, such that if A reads x starting in any state, it will end up in s_f. The Černý conjecture states that every n-state DFA can be synchronized by a word of length $(n-1)^2$ if it can be synchronized at all [9]. Although this bound was proven for several classes of finite-state automata, the general case is still widely open. The currently best upper bound is cubic, and only very little progress has been made; see [19,24,26,27].

The notion of a synchronizing word is not only important from a mathematical perspective, offering a nice combinatorial question, but it is quite important in a number of application areas, simply because synchronization is an important concept for many applied areas: parallel and distributed programming, system and protocol testing, information coding, robotics, etc. Therefore, it is also interesting to compute a shortest synchronizing word. Unfortunately, as it was shown by Ryststov and Eppstein in [14,25], the corresponding decision problem DFA-SW (defined in the following) is NP-complete. Possible applications of this problem are explained in [21]. The problem has also been considered from the viewpoint of approximation [1] and parameterized complexity [4,15,17,23].

DFA-SW
Input: DFA A, $k \in \mathbb{N}$
Question: Is there a synchronizing word w for A with $|w| \leq k$?

We will continue to study this problem from the point of parameterized complexity. The standard parameter for this problem is the length upper bound k, which we assume to be the case without further mentioning in this paper. W.l.o.g., we assume that k is given in unary. It was shown in [4,15,23] that this problem is W[2]-hard, even when restricted to quite particular (and restricted) forms of finite automata. Also, other parameters have been studied, in particular, in [15]. Two decades ago, in [5], Cai, Chen, Downey and Fellows introduced the following algebraic problem.

Monoid Factorization (see [5])
Input: A finite set Q, a collection $F = \{f_0, f_1, \ldots, f_m\}$ of mappings $f_i : Q \to Q$, $k \in \mathbb{N}$
Question: Is there a selection of at most k mappings $f_{i_1}, \ldots, f_{i_{k'}}$, $k' \leq k$, with $i_j \in \{1, \ldots, m\}$ for $j = 1, \ldots, k'$, such that $f_0 = f_{i_1} \circ f_{i_2} \circ \cdots \circ f_{i_{k'}}$?

Again, k is the standard parameter which we will also consider (exclusively) in this paper. In [5], it was proven that Monoid Factorization is W[2]-hard. We prove in this paper that both problems are in fact equivalent in a parameterized sense. Furthermore, we exhibit three parameterized complexity classes to which both problems belong to, namely, A[2], W[P] and WNL. This indicates that DFA-SW is not complete for any of these classes and hence, we suggest a new parameterized complexity class W[Sync] as a proper home for these two parameterized problems (and more, as we will show).

Throughout this paper, we assume the reader to be familiar with some concepts from parameterized complexity. In particular, a *parameterized reduction* is a many-one reduction that consumes FPT-time (in our cases, it mostly uses only polynomial time) and translates a parameter value k to a parameter value of $f(k)$ (of the target problem), for some computable function f. A parameterized complexity class can be characterized by one (complete) problem, assuming the class is closed under parameterized reductions. Examples comprise the following classes; for the typical problems, the parameter will be always called k:

W[1] Given a nondeterministic single-tape Turing machine and $k \in \mathbb{N}$, does it accept the empty word within at most k steps?

W[2] Given a nondeterministic multi-tape Turing machine and $k \in \mathbb{N}$, does it accept the empty word within at most k steps?

A[2] Given an alternating single-tape Turing machine whose initial state is existential and that is allowed to switch only once into the set of universal states and $k \in \mathbb{N}$, does it accept the empty word within at most k steps?

WNL Given a nondeterministic single-tape Turing machine and some integer $\ell \geq 0$ in unary and $k \in \mathbb{N}$, does it accept the empty word within at most ℓ steps, visiting at most k tape cells?

W[P] Given a nondeterministic single-tape Turing machine and some integer $\ell \geq 0$ in unary and $k \in \mathbb{N}$, does it accept the empty word within at most ℓ steps, thereby making at most $k \leq \ell$ nondeterministic steps?

More details can be found in textbooks like [12,18]. The *Turing way* to these complexity classes is described also in [10,20]. Further interesting complexity classes (in our discussion) are: FPT, W[3], W[SAT], A[3], para-NP and XP. From the literature, the following relations are known:

- $\mathsf{FPT} \subseteq \mathsf{W}[1] \subseteq \mathsf{W}[2] \subseteq \mathsf{W}[3] \subseteq \cdots \subseteq \mathsf{W}[\mathsf{SAT}] \subseteq \mathsf{W}[\mathsf{P}] \subseteq (\mathsf{para\text{-}NP} \cap \mathsf{XP})$;
- $\mathsf{FPT} \subseteq \mathsf{W}[1] = \mathsf{A}[1] \subseteq \mathsf{W}[2] \subseteq \mathsf{A}[2] \subseteq \mathsf{A}[3] \subseteq \cdots \subseteq \mathsf{AW}[\mathsf{P}] \subseteq \mathsf{XP}$.

Each of the inclusions that we have explicitly written is conjectured but not known to be strict. Also, no non-trivial inter-relations are known between the A- and W-hierarchies, apart from $\mathsf{W}[t] \subseteq \mathsf{A}[t]$ for each t.

Guillemot defined WNL in [20] in the same way as we described it above. Interesting formal language problems complete for WNL include Bounded DFA-Intersection, given k DFAs, with parameter k, plus the length of the word that should be accepted by all k automata, or Longest Common Subsequence, parameterized by the number of given strings. WNL is situated above all levels of the W-hierarchy, because the last two mentioned problems are known to be hard for $\mathsf{W}[t]$ for any $t \geq 1$, see [3,30]. This proves the first part of the following theorem that we include also for the ease of reference.

Theorem 1. *$\bigcup_{t \geq 1} \mathsf{W}[t] \subseteq \mathsf{WNL} \subseteq (\mathsf{para\text{-}NP} \cap \mathsf{XP})$.*

Proof. Clearly, by a standard product automaton construction, Bounded DFA-Intersection can be tested in time $\mathcal{O}(n^k)$, where n is the maximum number of states of the input DFAs. Hence, WNL is included in XP.

Recall that membership of Bounded DFA-Intersection (parameterized by the number of automata) in WNL follows by guessing an input word letter-by-letter, keeping track of the DFAs by writing their k current states, plus a counter for the number of steps, on the tape of the Turing machine M. We can do so by using as many letters as there are states in the automata, plus q (which is given in unary). Alternatively, when counting the number of bits needed to write down the tape contents using the alphabet $\{0, 1\}$, this amounts in $\mathcal{O}(k \log(n))$ many bits, if n upper-bounds the size (number of bits) of (an encoding) of a Bounded DFA-Intersection instance. Assuming that M has s many states, then there are obviously no more than $s \cdot 2^{\mathcal{O}(k \log n)}$ many configurations of M. With the help of an additional counter, using $\log(s \cdot 2^{\mathcal{O}(k \log n)}) = \log(s) + \mathcal{O}(k \log n)$ many additional bits, we can ensure that such a nondeterministic Turing machine M' (simulating M) would need no more time than $s \cdot 2^{\mathcal{O}(k \log n)}$ when moving through the configuration graph, avoiding visiting configurations twice. This proves the claimed membership in para-NP. Hence, WNL is included in para-NP. □

As a final remark concerning this detour to parameterized complexity, observe that WNL is also closely linked to the class N[f poly, f log] of parameterized problems that can be solved nondeterministically (at the same time) obeying some time bound $f(k) \cdot n^c$ for some constant c and some space bound $f(k) \cdot \log(n)$, where f is some (computable) function, k is the parameter (value) and n gives the instance size, as discussed in [13]. Our reasoning also shows that Bounded DFA-Intersection (parameterized by the number of automata) lies in N[f poly, f log]. Hence, WNL can be seen as the closure of N[f poly, f log] under parameterized reductions. So, although one can argue that N[f poly, f log] (and also some other classes introduced by Elberfeld, Stockhusen and Tantau) is a better model of parameterized space complexity, WNL fits better into the landscape depicted in Fig. 1, being closed under parameterized reductions by its definition. Elberfeld, Stockhusen and Tantau [13] chose other types of reductions.

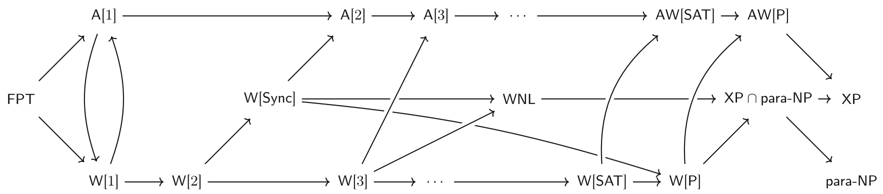

Fig. 1. Visualization of the complexity classes ('A → B' means 'A is contained in B')

2 Finding a Home for DFA-SW

As mentioned above, DFA-SW is known to be W[2]-hard. However, no complexity class was hitherto suggested to which DFA-SW belongs. In this section, we will describe three different memberships.

Theorem 2. DFA-SW *is contained in the classes WNL and W[P].*

Proof. Given a DFA A with state set Q and input alphabet Σ, where, w.l.o.g., $Q \cap \Sigma = \emptyset$, together with a bound k on the length of a synchronizing word, a Turing machine M is constructed that works as follows: (1) M writes a word of length at most k over the alphabet Σ on its tape, followed by some letter over the alphabet Q. (2) For each $q \in Q$ (this information can be hard-coded in the finite-state memory of M), M first moves its head to the left end of its tape and then starts reading the tape content from left to right. Each time a symbol $a \in \Sigma$ is read, M updates the current state it stores according to the transition function of A. Finally, M will read a symbol from Q, and it will only continue working if this symbol equals the current state stored in the finite memory of M. Notice that (2) works deterministically. (3) Only if M has completely processed the loop described in (2) (without abort), M will accept. This verifies that the guessed word over Σ is indeed synchronizing, always leading into the state that was also previously guessed. Hence, M will accept the empty word if and only if there is a possibility to guess a synchronizing word of length at most k. It is also clear that the Turing machine makes at most $(|Q|+1)(2k+1)$ many steps, visiting at most $k+1$ tape cells, thereby making at most $k+1$ guesses. □

We failed when trying to put DFA-SW into W[SAT]. By observing that the switch between phases (1) and (2) of the description of the Turing machine M in the previous proof can be also viewed as switching between existentially and universally quantified states, M can be also re-interpreted to show:

Theorem 3. DFA-SW *is contained in the class* $A[2]$.

3 How to Factor Monoids

We are now going to prove that Monoid Factorization is FPT-equivalent to DFA-SW.

Lemma 1. *There is a polynomial-time computable parameterized many-one reduction from* Monoid Factorization *to* DFA-SW.

Proof. Let $F = \{f_0, f_1, \ldots, f_m\}$ be a collection of mappings $f_i : Q \to Q$ and $k \in \mathbb{N}$. Define $\hat{Q} = Q \times Q \cup \{s_0, \ldots, s_k, s_{k+1}, f\}$. Let $\Sigma = \{a_1, \ldots, a_m, \sigma, \tau\}$ and define the transition function $\delta : \hat{Q} \times \Sigma \to \hat{Q}$ as follows.

$$\delta(p,x) = \begin{cases} (q_1, f_i(q_2)) & \text{if } p = (q_1,q_2), x = a_i \text{ for some } 1 \le i \le m \\ (q_1, q_1) & \text{if } p = (q_1,q_2), x = \sigma, \text{ or } x = \tau \text{ and } q_2 \ne f_0(q_1) \\ f & \text{if } p = (q_1,q_2), x = \tau \text{ and } q_2 = f_0(q_1) \\ s_0 & \text{if } p = s_0, x \ne \sigma \\ s_1 & \text{if } p = s_0, x = \sigma \\ s_0 & \text{if } p = s_i, x = \tau, i = 1, \ldots, k \\ s_{i+1} & \text{if } p = s_i, x \ne \tau, i = 1, \ldots, k \\ f & \text{if } p = s_{k+1}, x = \tau \\ s_{k+1} & \text{if } p = s_{k+1}, x \ne \tau \\ f & \text{if } p = f, x \in \Sigma \end{cases}$$

This describes the interesting aspects of the automaton A_F. We claim that (F,k) is a YES-instance of Monoid Factorization if and only if $(A_F, k+2)$ is a YES-instance of DFA-SW.

Namely, if (F,k) is a YES-instance of Monoid Factorization, then there exists a selection of at most k mappings $f_{i_1}, \ldots, f_{i_{k'}}$, $k' \le k$, with $i_j \in \{1, \ldots, m\}$ for $j = 1, \ldots, k'$, such that $f_0 = f_{i_1} \circ f_{i_2} \circ \cdots \circ f_{i_{k'}}$. Then, $w = \sigma^{k-k'+1} a_{i_1} \cdot a_{i_2} \cdots a_{i_{k'}} \tau$ synchronizes A_F. Clearly, w begins with $\sigma^{k-k'+1}$. When started in some (q_1, q_2), A_F will be in state (q_1, q_1) after digesting $\sigma^{k-k'+1}$. The word $a_{i_1} \cdot a_{i_2} \cdots a_{i_{k'}}$ will then drive A_F into some state (q_1, q_2'). Now, upon reading τ, A_F could only enter (the only) synchronizing state f if $q_2' = f_0(q_1)$ was true. If A_F starts reading w in any of the states $\{s_0, \ldots, s_k, s_{k+1}, f\}$, it is straightforward to check that A_F will be in state f thereafter.

Conversely, if w is any word of length at most $k+2$ that is synchronizing for A_F, then it must be of length exactly $k+2$, as this is the shortest path length from s_0 down to f, which is a sink state and must be hence the synchronizing state. This also enforces w to start with σ and to end with τ. Also, w cannot contain another occurrence of τ, as this would lead to s_0 again (from any of the states s_i) and hence prevent w from entering f, because the states s_i should be walked through one-by-one, hence counting up to $k+2$. Let us study the longest suffix $v\tau$ of w that satisfies $v \in \{a_1, \ldots, a_m\}^*$. By the structure of w that we analyzed before, we must have $w = u\sigma v\tau$, for some possibly empty word u such that $u\sigma$ starts with σ. In particular, $|v| \leq k$, as $|u| + |v| = k$. Hence, after reading the symbol σ preceding v, A_F will be in one of the states (q, q) or s_i (for some $|u| + 1 \leq i \leq k+1$) or f. Now, digesting v leads us into one of the states s_{k+1} or f or (q, p), with $p = (f_{i_1} \circ f_{i_2} \circ \cdots \circ f_{i_{k'}})(q)$, from which we can enter f only (after reading τ) if $f_0(q) = p$. This shows that, if $u = a_{i_1} \cdot a_{i_2} \cdots a_{i_{k'}}$, then $f_0 = f_{i_1} \circ f_{i_2} \circ \cdots \circ f_{i_{k'}}$. □

Lemma 2. *There is a polynomial-time computable parameterized reduction that produces, given some DFA A and some integer k as an instance of* DFA-SW*, an equivalent instance (A', k') of* DFA-SW *such that A' possesses a sink state (which is then also the unique possible synchronizing state).*

Proof. Consider the DFA $A = (Q, \Sigma, \delta, q_0, F)$. Without loss of generality, assume $\Sigma \cap Q = \emptyset$ and $\sigma \notin \Sigma \cup Q$. Let $\Sigma' = \Sigma \cup Q \cup \{\sigma\}$ be the input alphabet of the DFA A' that we are going to construct. Let $s_0, \ldots, s_k, f \notin Q$ be fresh states. Let $Q' = Q \cup \{s_0, \ldots, s_k, f\}$ be the states of A'. Define the transition function δ' as:

$$\delta'(p, x) = \begin{cases} \delta(p, x) & \text{if } p \in Q, x \in \Sigma \\ p & \text{if } p \in Q, x = \sigma \vee x \in Q \setminus \{p\} \\ f & \text{if } p \in Q, x = p \\ s_0 & \text{if } p = s_i, x \in Q, i = 0, \ldots, k-1 \\ s_{i+1} & \text{if } p = s_i, x \notin Q, i = 0, \ldots, k-1 \\ f & \text{if } p = s_k, x \in Q \\ s_k & \text{if } p = s_k, x \notin Q \\ f & \text{if } p = f, x \in \Sigma' \end{cases}$$

This describes the interesting aspects of the automaton A'. We claim that, letting $k' = k+1$, then A has a synchronizing word of length at most k if and only if A' has a synchronizing word of length (at most and exactly) k'.

Let $w \in \Sigma^*$ be a synchronizing word, leading A into state $q_f \in Q$, with $|w| \leq k$. Then it is easy to observe that the word $w' = \sigma^{k-|w|} w q_f$ leads A' into the sink state f, wherever A' starts. Hence, w' is a synchronizing word of length k' as claimed. Notice that due to the sequence of states $s_0, \ldots, s_k, f$, there cannot be any shorter synchronizing word in A'.

Conversely, let w' be a synchronizing word of length at most k' for A'. As f is a sink state, it must be the synchronizing state. Since in particular $\delta'^*(s_0, w') = f$, $|w'| = k' = k+1$, and for the same reason, $w' = w''q$ for some $w'' \in (\Sigma \cup \{\sigma\})^k$ and $q \in Q$. Observe that the special letter σ either loops (on $Q \cup \{f\}$) or advances as any other letter from Σ (on $Q' \setminus Q$). Therefore, if w' is synchronizing for A', then so is $\sigma^{k-|w|}wq$, where w is obtained from w'' by deleting all occurrences of σ, i.e., $w \in \Sigma^*$. As σ acts as the identity on Q, and because the final letter q indicates that, upon starting in some state from Q, the automaton must have reached state q (as w' is leading to the sink state f), we can see that w is indeed a synchronizing word for A; moreover, $|w| \leq k$. □

Theorem 4. Monoid Factorization *is (parameterized and polynomial-time) equivalent to* DFA-SW*.*

Proof. By Lemma 1, we can reduce Monoid Factorization to DFA-SW. Conversely, by Lemma 2, we need to consider only instances of DFA-SW that have a sink state. With some background knowledge on transition monoids, it is clear that by interpreting a given DFA $A = (Q, \Sigma, \delta, q_0, F)$ with sink state s_f as a collection F_A of $|\Sigma|$ many mappings $f_a : Q \to Q$, by setting $f_a(q) = \delta(q, a)$, we can solve a DFA synchronization problem given by (A, k) by solving the instance (F, k) of Monoid Factorization, where $F = \{f_0 = s_f\} \cup F_A$ and the aim is to represent the constant target map $f_0 = s_f$. □

This motivates us to suggest a new parameterized complexity class W[Sync] as the class of parameterized problems that can be reduced to DFA-SW (Fig. 1).

Corollary 1. Monoid Factorization *is W[Sync]-complete.*

4 More Problems Complete for or Contained in W[Sync]

Theorem 5. Bounded DFA-Intersection*, parameterized by the length of the commonly accepted string, is complete for W[Sync].*

Previously [30], only W[2]-hardness was known for this parameterized problem.

Proof. By Lemma 2, we need to consider only an instance $A = (Q, \Sigma, \delta, q_0, F)$ of DFA-SW with a sink state s_f. Observe that A has a synchronizing word of length at most k if and only if A has a synchronizing word of length exactly k, because wu is a synchronizing word if w is. Define $A_q = (Q, \Sigma, \delta, q, \{s_f\})$. Observe that $\bigcap_{q \in Q} L(A_q)$ contains some word $w \in \Sigma^k$ if and only if A has a synchronizing word of length exactly k.

Conversely, if $\{A_i \mid 1 \leq i \leq \ell\}$ is a collection of DFAs $A_i = (Q_i, \Sigma, \delta_i, q_{0,i}, F_i)$, then construct an equivalent instance of DFA-SW as follows. First, assume that the state sets Q_i are pairwise disjoint. Then, take two new letters a, b to form

$$\delta(p,x) = \begin{cases} \delta_i(p,x) & \text{if } p \in Q_i, x \in \Sigma \\ q_{0,i} & \text{if } p \in Q_i, x = \sigma \\ f & \text{if } p \in \left(\bigcup_{i=1}^{\ell} F_i\right), x = \tau \\ p & \text{if } p \in \left(\bigcup_{i=1}^{\ell} (Q_i \setminus F_i)\right), x = \tau \\ s_0 & \text{if } p = s_i, x = \tau, i = 0, \ldots, k \\ s_{i+1} & \text{if } p = s_i, x \neq \tau, i = 0, \ldots, k \\ f & \text{if } p = s_{k+1}, x = \tau \\ s_{k+1} & \text{if } p = s_{k+1}, x \neq \tau \\ f & \text{if } p = f, x \in \Sigma' \end{cases}$$

Fig. 2. Transition function δ of the constructed W[Sync] instance.

$\Sigma' = \Sigma \cup \{\sigma, \tau\}$. Let $Q' = \left(\bigcup_{i=1}^{\ell} Q_i\right) \cup \{s_0, \ldots, s_k, s_{k+1}, f\}$ be the state set of the DFA A that we construct. Define the transition function δ as in Fig. 2.

This describes the interesting aspects of the automaton A. We claim that, letting $k' = k + 2$, then $\bigcap_{i=1}^{\ell} L(A_i)$ contains some word $w \in \Sigma^k$ if and only if A has a synchronizing word of length (at most and exactly) k', namely $w' = \sigma w \tau$. More precisely, similar to the construction from Lemma 1, the states s_i force to consider a word from $\{\sigma\}\Sigma^k\{\tau\}$ if there should be a synchronizing word of length k' for A at all. One could move only from the part A_i of A to f when reading τ, which also forces to have been in the set of final states F_i before. Digesting σ as the first letter lets A_i start in the initial state $q_{0,i}$. □

We now discuss the well-known Longest Common Subsequence problem. The input consists of ℓ strings $x_1, \ldots, x_\ell$ over an alphabet Σ, and the task is to find a string $w \in \Sigma^k$ occurring in each of the x_i as a subsequence. As explained in [30], by building an automaton A_i for each x_i that accepts all subsequences of x_i, it is not hard to solve a Longest Common Subsequence instance by a Bounded DFA-Intersection instance, preserving our parameter. Hence:

Proposition 1. Longest Common Subsequence $\in$ *W*[*Sync*].

We do not know if Longest Common Subsequence is also hard for W[Sync]. We only know W[2]-hardness from [3], further membership results were unknown hitherto, so the previous proposition remedies this situation a bit.

One could also think of many ways to restrict the inputs of Bounded DFA-Intersection. For instance, observe that the automata constructed in the argument of Proposition 1 are all accepting finite languages. Is there a converse reduction from such a Bounded DFA-Intersection instance to some Longest Common Subsequence instance? Is this leading to another complexity class between W[2] and W[Sync]?

In [4], we discussed the restriction of DFAs to so-called TTSPL graphs for instances of DFA-SW. While we could prove W[2]-hardness also for such restricted instances, it is open if this leads to a problem that is still complete for W[Sync]. As shown by Möhring [22], there are quite close connections between

TTSP(L) graphs and so-called series-parallel partial orders. Without going into any details here, observe that the mappings $Q \to Q$ that can be associated to input letters are monotone with respect to the series-parallel partial order corresponding to the TTSPL automaton graph. Our earlier constructions show:

Corollary 2. DFA-SW, *restricted to DFAs with TTSPL automata graphs, is parameterized and polynomial-time equivalent to* Monoid Factorization, *restricted to collections of mappings F that are monotone with respect to a given series-parallel partial order on the finite ground set Q.*

This discussion also entails the (open) question concerning the complexity status of Monoid Factorization, restricted to collections of mappings F that are monotone with respect to a given partial order on the finite ground set Q.

5 Further Comments

The problems that we considered in this paper have quite a rich structure and many variations. We will comment on these variations in this section.

5.1 Variations on Monoid Factorization

Observe that it is important that the monoid used in Monoid Factorization is only implicitly given, not by a multiplication table. A variation could be:

Input: A finite set M, a binary operation $\circ$ given in the form of a multiplication table, such that $(M, \circ)$ forms a finite monoid, with neutral element $e \in M$, a target element $t \in M$, a finite subset $B \subseteq M$, $k \in \mathbb{N}$
Question: Is there a selection of at most k elements $b_1, \ldots, b_{k'}$, $k' \leq k$, from B, such that $t = b_1 \circ b_2 \circ \cdots \circ b_{k'}$?

However, an explicit representation of the multiplication table of $(Q^Q, \circ)$ (where Q^Q is the set of all mappings from Q to Q) would already take $\mathcal{O}^*(|Q|^{2|Q|})$ space and hence allow to construct an arc-labeled directed graph with a vertex for each mapping $Q \to Q$ and an arc labeled f_i from f to g if $f \circ f_i = g$, where f_i is from the explicit set of generators $F' = \{f_1, \ldots, f_m\}$. Now, the representability of f_0 with at most k mappings from F' can be solved by looking for a path of length at most k in the directed graph we just described, leading from the identity mapping Δ_Q to f_0. Hence, when the monoid is given in an explicit form, then the factorization problem can be solved in polynomial time. It might be interesting to study other implicitly given monoids with respect to the factorization question. Let us mention one more example. Assume that our implicitly given monoid operation is *set union*. Then, the corresponding factorization problem would take subsets $\{X_0, X_1, \ldots, X_m\}$ of a given finite set S as an input, and the question is to pick at most k sets from $\{X_1, \ldots, X_m\}$, say, $X_{i_1}, \ldots, X_{i_{k'}}$, where $k' \leq k$, such that $X_0 = \bigcup_{j=1}^{k'} X_{i_j}$. Obviously, this corresponds to Set Cover, which hence gives an example of a monoid factorization problem which, when parameterized by k, is complete for W[2]. It might be interesting to investigate further implicitly given monoids from this parameterized perspective. We only mention as a last example from the literature Permutation Group Factorization, which is known to be W[1]-hard but is lacking a precise classification; see [3,12].

5.2 Extension Variants

Following [6,16], we are now defining so-called extension problems, depending on the chosen partial order $\prec$ on Σ^*. Maybe surprisingly, the complexity status of these problems heavily depends on this choice.

Ext DFA-SW-$\prec$
Input: DFA A with input alphabet Σ, $u \in \Sigma^*$
Question: Is there a $w \in \Sigma^*$, $u \prec w$, such that w is minimal for the set of synchronizing words for A with respect to $\prec$?

We are focussing on the length-lexicographical ordering $\leq_{ll}$ and the subsequence ordering $|$ in the following. For further orderings, we refer to [16]. We consider $|u|$ to be the standard parameter.

Theorem 6. Ext DFA-SW-$\leq_{ll}$ *is contained in* co-WNL $\cap$ co-W[P] $\cap$ co-A[2], *but hard for* co-W[Sync].

Proof. For membership in co-WNL $\cap$ co-W[P] $\cap$ co-A[2], we can modify the proofs of Theorems 2 or 3, building a nondeterministic Turing machine M as follows, given A and u. As before, the machine can first guess a possible word $w \leq_{ll} u$ and verify if it is synchronizing. If such a word is found, then (A, u) is a NO-instance. The reduction itself checks if A is synchronizable at all; then we also have that if M does not find a synchronizing word $w \leq_{ll} u$, then (A, u) is a YES-instance, because as A is synchronizable, there must be a synchronizing word v, and according to the previous tests, $u \leq_{ll} v$ must hold.

For the hardness claim, consider a DFA A on input alphabet Σ, together with k, as an instance of DFA-SW. We can first check in polynomial time if A is synchronizable at all. If A is not synchronizable, then (A, k) (clearly) is a NO-instance of DFA-SW, so our reduction will produce some fixed NO-instance of Ext DFA-SW-$\leq_{ll}$. Hence, we now assume that A is synchronizable. Let $c \notin \Sigma$ be a fresh letter. Consider an arbitrary ordering $<$ on Σ, extended by $c < x$ for all $x \in \Sigma$ towards an ordering on $\hat{\Sigma} = \Sigma \cup \{c\}$. We are going to define the DFA $\hat{A}$ as an extension of A, working on the same state set Q. Let c simply act as the identity on Q. Hence, no word from c^* is synchronizing for $\hat{A}$. As A is synchronizable, $\hat{A}$ is also synchronizable. Consider $\hat{A}$ together with $u = c^{k+1}$ as an instance of Ext DFA-SW-$\leq_{ll}$. If $\hat{A}$ has a synchronizing word $w \in \Sigma^{\leq k}$, then clearly u is not extendible, as $|w| < |u|$. Otherwise, as $\hat{A}$ is synchronizable, $\hat{A}$ must have some synchronizing word w with $|w| \geq |u|$, and any synchronizing word of $\hat{A}$ is of length at least $|u|$. As u is the smallest of all words in $\hat{\Sigma}^*$ of length at least $|u|$, any synchronizing word will hence extend u. Hence, if $\hat{A}$ has no synchronizing word of length at most k, then u is extendible. $\square$

Theorem 27 in [16] converted an instance of Ext Hitting Set into an instance of Ext DFA-SW-|. With [2], this proves that Ext DFA-SW-|, is W[3]-hard, lifting it beyond another rarely mentioned complexity class; see [11]. This construction can be also adapted for TTSPL automata graphs.

5.3 Minimum Synchronizable Sub-automata

DFA-MSS (referring to a minimum synchronizable sub-automaton)
Input: DFA A with input alphabet Σ, $k \in \mathbb{N}$
Question: Is there a sub-alphabet $\hat{\Sigma} \subseteq \Sigma$, $|\hat{\Sigma}| \leq k$, such that the restriction of A to $\hat{\Sigma}$ is synchonizable, i.e., is there a synchronizing word over $\hat{\Sigma}$?

In [28], Türker and Yenegün asked to extract a synchronizable sub-automaton that is as small as possible, obtained by deleting letters from its specification. They formalized this idea as a weighted minimization problem. Here, it is sufficient to consider the unweighted variant (defined in the box). Their NP-hardness proof can be re-interpreted as a result on parameterized complexity.

Corollary 3. DFA-MSS *is W[2]-hard.*

Without proof, we mention the following membership result. However, membership in A[2] is open, nor to we know about WNL-hardness. It might be also the case that DFA-MSS is W[Sync]-hard.

Theorem 7. DFA-MSS *is contained in* $\mathsf{WNL} \cap \mathsf{W}[\mathsf{P}]$.

A Short Summary. We looked at various W[2]-hard problems where a proper classification is still missing. In particular, problems rooting in Formal Languages offer interesting sample problems. Many questions are still open about W[Sync].

References

1. Berlinkov, M.V.: Approximating the minimum length of synchronizing words is hard. Theory Comput. Syst. **54**(2), 211–223 (2014)
2. Bläsius, T., Friedrich, T., Lischeid, J., Meeks, K., Schirneck, M.: Efficiently enumerating hitting sets of hypergraphs arising in data profiling. In: Algorithm Engineering and Experiments (ALENEX), pp. 130–143. SIAM (2019)
3. Bodlaender, H., Downey, R.G., Fellows, M.R., Wareham, H.T.: The parameterized complexity of sequence alignment and consensus. Theor. Comput. Sci. **147**, 31–54 (1995)
4. Bruchertseifer, J., Fernau, H.: Synchronizing series-parallel automata with loops. In: Freund, R., Holzer, M., Sempere, J.M. (eds.) Eleventh Workshop on Non-Classical Models of Automata and Applications, NCMA, pp. 63–78. Österreichische Computer Gesellschaft (2019)
5. Cai, L., Chen, J., Downey, R., Fellows, M.: On the parameterized complexity of short computation and factorization. Arch. Math. Logic **36**, 321–337 (1997)

6. Casel, K., Fernau, H., Ghadikolaei, M.K., Monnot, J., Sikora, F.: On the complexity of solution extension of optimization problems. Technical report arXiv:1810.04553 [cs.CC], Cornell University, arXiv (2018)
7. Černý, J.: Poznámka k homogénnym experimentom s konečnými automatmi. Matematicko-fyzikálny časopis **14**(3), 208–216 (1964)
8. Černý, J.: A note on homogeneous experiments with finite automata. J. Automata Lang. Comb. **24**(2–4), 123–132 (2019)
9. Černý, J., Pirická, A., Rosenauerová, B.: On directable automata. Kybernetika **7**(4), 289–298 (1971)
10. Cesati, M.: The Turing way to parameterized complexity. J. Comput. Syst. Sci. **67**, 654–685 (2003)
11. Chen, J., Zhang, F.: On product covering in 3-tier supply chain models: Natural complete problems for W[3] and W[4]. Theor. Comput. Sci. **363**(3), 278–288 (2006)
12. Downey, R.G., Fellows, M.R.: Fundamentals of Parameterized Complexity. Texts in Computer Science. Springer, Heidelberg (2013). https://doi.org/10.1007/978-1-4471-5559-1
13. Elberfeld, M., Stockhusen, C., Tantau, T.: On the space and circuit complexity of parameterized problems: classes and completeness. Algorithmica **71**(3), 661–701 (2015)
14. Eppstein, D.: Reset sequences for monotonic automata. SIAM J. Comput. **19**(3), 500–510 (1990)
15. Fernau, H., Heggernes, P., Villanger, Y.: A multi-parameter analysis of hard problems on deterministic finite automata. J. Comput. Syst. Sci. **81**(4), 747–765 (2015)
16. Fernau, H., Hoffmann, S.: Extensions to minimal synchronizing words. J. Automata Lang. Comb. **24**, 287–307 (2019)
17. Fernau, H., Krebs, A.: Problems on finite automata and the exponential time hypothesis. Algorithms **10**(24), 1–25 (2017)
18. Flum, J., Grohe, M.: Parameterized Complexity Theory. Springer, Heidelberg (2006). https://doi.org/10.1007/3-540-29953-X
19. Frankl, P.: An extremal problem for two families of sets. Eur. J. Comb. **3**(2), 125–127 (1982)
20. Guillemot, S.: Parameterized complexity and approximability of the longest compatible sequence problem. Discret. Optim. **8**(1), 50–60 (2011)
21. Kisielewicz, A., Kowalski, J., Szykuła, M.: Computing the shortest reset words of synchronizing automata. J. Comb. Optim. **29**(1), 88–124 (2013). https://doi.org/10.1007/s10878-013-9682-0
22. Möhring, R.H.: Computationally tractable classes of ordered sets. In: Rival, I. (ed.) Algorithms and Order: Proceedings of the NATO Advanced Study Institute. NATO Science Series C, vol. 255, pp. 105–194. Springer, Heidelberg (1989). https://doi.org/10.1007/978-94-009-2639-4_4
23. Andres Montoya, J., Nolasco, C.: On the synchronization of planar automata. In: Klein, S.T., Martín-Vide, C., Shapira, D. (eds.) LATA 2018. LNCS, vol. 10792, pp. 93–104. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-77313-1_7
24. Pin, J.E.: On two combinatorial problems arising from automata theory. Ann. Discret. Math. **17**, 535–548 (1983)
25. Rystsov, I.K.: On minimizing the length of synchronizing words for finite automata. In: Theory of Designing of Computing Systems, pp. 75–82. Institute of Cybernetics of Ukrainian Acad. Sci. (1980). (in Russian)
26. Shitov, Y.: An improvement to a recent upper bound for synchronizing words of finite automata. J. Automata Lang. Combin. **24**(2–4), 367–373 (2019)

27. Szykuła, M.: Improving the upper bound on the length of the shortest reset word. In: Niedermeier, R., Vallée, B. (eds.) 35th Symposium on Theoretical Aspects of Computer Science, STACS. LIPIcs, vol. 96, pp. 56:1–56:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2018)
28. Türker, U.C., Yenigün, H.: Complexities of some problems related to synchronizing, non-synchronizing and monotonic automata. Int. J. Found. Comput. Sci. **26**(1), 99–122 (2015)
29. Volkov, M.V.: Preface: special issue on the Černý conjecture. J. Automata Lang. Comb. **24**(2–4), 119–121 (2019)
30. Todd Wareham, H.: The parameterized complexity of intersection and composition operations on sets of finite-state automata. In: Yu, S., Păun, A. (eds.) CIAA 2000. LNCS, vol. 2088, pp. 302–310. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44674-5_26