


PAPER

Synchronizing words and monoid factorization, yielding a new parameterized complexity class?

Henning Fernau*  and Jens Bruchertseifer

Theoretische Informatik, Abteilung Informatikwissenschaften, Fachbereich 4, Universität Trier, Trier, Germany

*Corresponding author. Email: fernau@uni-trier.de

(Received 8 January 2021; revised 8 February 2022; accepted 31 May 2022; first published online 28 September 2022)

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 $W[\text{Sync}]$ as a proper home for these (and more) problems. We present quite a number of problems that belong to $W[\text{Sync}]$ or are hard or complete for this new class.

Keywords: Synchronizing word; deterministic finite automaton (DFA); parameterized complexity; Monoid Factorization; $W[\text{Sync}]$

1. Introduction

Černý's conjecture is arguably the most famous open combinatorial problem concerning deterministic finite automata (DFA), somehow dating back to Černý (1964).¹ Recently, a particular Special Issue was dedicated to this conjecture being around for more than five decades; see Volkov (2019). This Special Issue also contains an English translation of Černý's paper, originally written in Slovak, see Černý (2019).

The key notion in Černý's conjecture 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 given by Černý et al. (1971) states that every n -state DFA can be synchronized by a word of length $(n - 1)^2$ if it can be synchronized at all. 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 Frankl (1982), Pin (1983), Shitov (2019), Szykuła (2018) in alphabetical, but incidentally also nearly in chronological order. Conversely, already Černý (1964) devised a family of DFAs that attain the mentioned quadratic bound. For instance, the 4-state automaton from Figure 3 has a smallest synchronizing word of length nine.

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 Rystsov (1980) and Eppstein (1990), the corresponding decision problem DFA-SW (defined in the following) is NP-complete. Possible applications of this problem are explained in Kisielewicz et al. (2015). The problem has also been considered from the viewpoint of approximation by Berlinkov (2014) and parameterized complexity by Bruchertseifer and Fernau (2021), Fernau et al. (2015), Fernau and Krebs (2017), Montoya and Nolasco (2018).

DFA-SYNCHRONIZING WORD (DFA-SW)

Input: DFA A , $k \in \mathbb{N}$

Problem: Is there a synchronizing word w for A with $|w| \leq k$?

It is important to note at this point that we are working with complete DFAs in our definition; that is, a DFA can be specified as $A = (Q, \Sigma, \delta, q_0, F)$, where Q is the finite-state alphabet, Σ is the finite input alphabet, $\delta : Q \times \Sigma \rightarrow Q$ is the complete transition function, and $q_0 \in Q$ and $F \subseteq Q$ specify the start state and the final state set that are actually inessential for the task of synchronization. While DFA-SW is NP-complete, the complexity picture actually changes if we allow undefined transitions (i.e., if we move over to partial DFAs); the corresponding synchronization problem then becomes PSPACE-complete; see Martyugin (2014). In fact, also based on the mentioned numerous applications of synchronizing automata, many variations have been studied. Here, we only mention some recent papers as Fernau et al. (2019, 2020), Fernau and Wolf (2020), Wolf (2020). However, we will mostly restrict our attention to the most classical variant of synchronizing automata in this paper.

We will continue to study the problem (DFA-SW) from the point of parameterized complexity. The standard parameter for this problem is the length upper bound k . Also with future definitions of parameterized problems, we will follow the convention that k always denotes the parameter. As the synchronizability of DFAs can be checked in polynomial time (see Sandberg 2005; Volkov 2008) and due to the polynomial upper bound on a shortest synchronizing word (if existent) as discussed above, we can assume that the parameter k is given in unary or binary, whatever is more convenient to us. It was shown in Bruchertseifer and Fernau (2021), Fernau et al. (2015), Montoya and Nolasco (2018) 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 Fernau et al. (2015), but they are of less interest to the present study where we focus on the standard parameter.

2. Organization of the Paper and Main Results

In Section 3, we introduce some basic notions of parameterized complexity. Observe that we pay special attention to some complexity classes like $A[2]$ or WNL that are less studied in the literature. Most problems that we look at are $W[2]$ -hard, but in our opinion, from a complexity theoretic viewpoint, an exact classification of parameterized problems should be at least attempted, beyond stating that they are $W[1]$ -hard and hence most likely outside FPT. In Section 4, we focus on studying (DFA-SW), a problem already defined in the introduction; here, our focus is on the length upper bound k as a parameter. As we can see in Section 5, there are several problems FPT-equivalent to (DFA-SW); this motivated us to introduce a new class of parameterized problems that we call $W[\text{Sync}]$. We prove that

$$W[2] \subseteq W[\text{Sync}] \subseteq WNL \cap W[P] \cap A[2]$$

as a main result of Section 5. In particular, MONOID FACTORIZATION, BOUNDED TRANSFORMATION RANK(r), and BOUNDED DFA-INTERSECTION are complete for $W[\text{Sync}]$. In Section 6, we describe problems that are both $W[2]$ -hard and belong to $W[\text{Sync}]$. Conversely, in

Section 7, we look into problems that are $W[\text{Sync}]$ -hard and belong to some of the classes WNL or $W[P]$ or $A[2]$. We also list problems (in Section 8) that behave similar to $DFA\text{-}SW$, being $W[2]$ -hard and belonging to some of the classes WNL or $W[P]$ or $A[2]$, but where the exact relation to $W[\text{Sync}]$ is unclear. In Section 9, we return to variations of $MONOID\ FACTORIZATION$ that often present a completely different complexity behavior. In all sections, we highlight open problems in the hope that this paper can serve as an incentive for further research. More general questions and further thoughts are collected in the concluding Section 10.

A preliminary version of this paper appeared in Bruchertseifer and Fernau (2020).

3. A Primer in Parameterized Complexity

In this paper, we will encounter quite a number of notions in parameterized complexity. We will assume some familiarity with basic notions of that theory on side of the reader in the following. 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 . One important property of this (as of all reasonable reduction notions) is transitivity: If there is a parameterized reduction from (Π_1, κ_1) to (Π_2, κ_2) and from (Π_2, κ_2) to (Π_3, κ_3) , then there is parameterized reduction from (Π_1, κ_1) to (Π_3, κ_3) . Here, κ_i denotes the parameterization (function) for clarity. 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 Downey and Fellows (2013), Flum and Grohe (2006). The *Turing way* to these complexity classes (i.e., using variants of Turing machines and defining parameterized problems on these machines) is described also in Cesati (2003), Guillemot (2011). However, choosing a “typical” problem Π and defining a (parameterized) complexity class as the set of all (parameterized) problems that are reducible (with a reasonable notion of reducibility) to Π clearly also works for problems not related to Turing machines. In this paper, we actually suggest a new parameterized complexity class $W[\text{Sync}]$ as the class of parameterized problems that can be (parameterized) reduced to $DFA\text{-}SW$, a class that is situated as drawn in Figure 1, that is, somewhere between $W[2]$ at its lower end and the intersection of WNL , $A[2]$ and $W[P]$ at its upper end. Most of the remainder of this section and the following one is devoted to proving this relation of $W[\text{Sync}]$ to $W[2]$, WNL , $A[2]$ and $W[P]$. Further interesting complexity classes (in our discussion) are as follows: FPT , $W[3]$, $W[\text{SAT}]$, $A[3]$, para-NP and XP . Let us briefly introduce the complexity classes FPT ,² XP and para-NP , also see Flum and Grohe (2006): If a parameterized

problem Π can be solved in time $\mathcal{O}^*(f(k))$ (for some function f) by a deterministic algorithm, Π belongs to FPT. If a parameterized problem Π can be solved in time $n^{f(k)}$ for instances (x, k) of size n for some function f , then $\Pi \in \text{XP}$. If there is a nondeterministic algorithm running in time $\mathcal{O}^*(f(k))$ (for some function f), then $\Pi \in \text{para-NP}$. For definitions of the other classes, we refer to the mentioned textbooks. From the literature, the following relations are known:

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

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 $\text{W}[t] \subseteq \text{A}[t]$ for each t .

Guillemot defined WNL in Guillemot (2011) 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. We will come back to these problems with different parameterizations below. WNL is situated above all levels of the W-hierarchy, because the last two mentioned problems are known to be hard for $\text{W}[t]$ for any $t \geq 1$, see Bodlaender et al. (1995), Wareham (2001). This proves the first part of the following theorem that we include also for the ease of reference.

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

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. \square

As a final remark concerning this detour to parameterized complexity, observe that WNL is also closely linked to the class $\text{N}[f \text{ poly}, f \log]$ 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 Elberfeld et al. (2015). Our reasoning also shows that BOUNDED DFA-INTERSECTION (parameterized by the number of automata) lies in $\text{N}[f \text{ poly}, f \log]$. Hence, WNL can be seen as the closure of $\text{N}[f \text{ poly}, f \log]$ under parameterized reductions. So, although one can argue that $\text{N}[f \text{ poly}, f \log]$ (and also some other classes introduced by Elberfeld et al. 2015) is a better model of parameterized space complexity, WNL fits better into the landscape of the W-hierarchy also depicted in Figure 1, being closed under parameterized reductions by its definition. Recall that parameterized reductions speak about time bounds and are hence not that well related to space complexities. Therefore, Elberfeld, Stockhusen, and Tantau considered other types of reductions in Elberfeld et al. (2015).

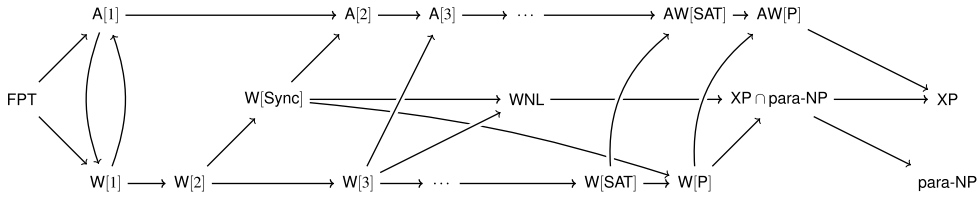


Figure 1. Visualization of the complexity classes (“A → B” means “A is contained in B”).

Because WNL is a less known class that is still quite central to our current discussion, let us mention a variation of DFA-SW in the following. For a subset of states $P \subseteq Q$ of a DFA $A = (Q, \Sigma, \delta, q_0, F)$, we say that $w \in \Sigma^*$ is *P-synchronizing* if, for all $p \in P$, $\delta^*(p, w) = s$ for some $s \in Q$. In other words, a word is synchronizing for A if it is Q -synchronizing.

DFA-SW-FROM-SUBSET-SB (where SB indicates a size bound)

Input: DFA A with state set Q as well as $k, m \in \mathbb{N}$ and subset $P \subseteq Q$ with $|P| = k$

Problem: Is there a P -synchronizing word w for A with $|w| \leq m$?

Notice that if m is given in binary and the restriction $|P| = k$ is missing, then this problem is even PSPACE-complete, because subset synchronization in itself (without the length bound) is already PSPACE-complete; see Fernau et al. (2019) and Rystsov (1983) for more discussions. However, if m is presented in unary, as implicit in the problem definition, because the input DFA comes with a list of its states and this list contains more elements than k , then the problem is “only” NP-complete, as it also follows from Montoya and Nolasco (2018), but this also follows when considering $P = Q$. However, Montoya and Nolasco focused on the parameterized complexity of DFA-SW-FROM-SUBSET-SB and could prove the following result.

Theorem 2. DFA-SW-FROM-SUBSET-SB, parameterized with the set size parameter k , is complete for the class WNL, even when restricted to planar automata graph instances.

4. 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 3. DFA-SW, parameterized with the length parameter k , is contained in the class WNL.

Notice that membership in WNL does not follow with the arguments presented by Montoya and Nolasco (2018) for a different parameterization of the related problem DFA-SUBSET-SBSW, because that argument makes explicit use of the fact that the given set P of states contains k elements.

Proof. We now describe a nondeterministic Turing machine that visits at most $f(k)$ many cells (ever), providing the required reduction. 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 from the alphabet Q ; this means that M starts with making (at most) $k + 1$ nondeterministic guesses.
- (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.³ \square

Unfortunately, we do not know if our problem is WNL-hard. Let us also remark that our reasoning in the previous result also proves that DFA-SW, parameterized with the length parameter k , is contained in $N[f \text{ poly}, f \log]$, a class introduced in Elberfeld et al. (2015) and also discussed above. We could also re-interpret the arguments of the previous proof to show that DFA-SW belongs to $W[P]$, based on the Turing machine characterization of $W[P]$.

We failed when trying to put DFA-SW into $W[SAT]$. In order to understand the difficulties, let us discuss a different strategy how to put DFA-SW into $W[P]$. Recall the following relationships:

$$\bigcup_{t \geq 1} W[t] \subseteq W[SAT] \subseteq W[P] \subseteq (XP \cap \text{para-NP}).$$

$W[SAT]$ is characterized (via parameterized reductions) by the following problem. Given an arbitrary Boolean formula φ , with X being its set of variables, does there exist an assignment $\alpha : X \rightarrow \{0, 1\}$ that satisfies φ , such that at most k variables are set to true (i.e., to 1)? Here, k is the parameter, also called the *weight* of the assignment. Similarly, $W[P]$ is characterized by the following problem. Given a Boolean circuit C , with X being its set of Boolean input variables, does there exist an assignment $\alpha : X \rightarrow \{0, 1\}$ of weight at most k that satisfies C ? Apart from the fact that circuits (graph-theoretically speaking, directed acyclic graphs) can represent Boolean functions more succinctly than formulas, another difference comes from the (related) fact that circuits can contain implicit variables (associated to inner logical gates) that might have to be made explicit in Boolean formulas, which means in particular that the weight of an assignment can change when moving from a Boolean circuit to an equivalent Boolean formula. This is the reason why we can prove in the following that DFA-SW, parameterized with the length parameter k , is contained in $W[P]$, but we do not know if we can adapt this proof toward showing membership in $W[SAT]$. As mentioned above, we cannot tell what the relations between WNL and $W[SAT]$ or $W[P]$ are, but assuming that $WNL \neq W[P]$ holds, the following construction also shows that DFA-SW, parameterized with the length parameter k , is neither WNL-hard nor $W[P]$ -hard.

Theorem 4. *DFA-SW, parameterized with the length parameter k , is contained in the class $W[P]$.*

Proof. We only give a sketch of the construction of a Boolean circuit that allows an assignment of weight k that outputs true if and only if the given DFA $A = (Q, \Sigma, \delta, q_0, F)$ has a synchronizing word of length at most k .

- We introduce an input gate $I_{a,i}$ for each $a \in \Sigma$ and $1 \leq i \leq k$. The meaning should be the following: Symbol a is at position i in the synchronizing word that is guessed via an assignment to these input gates. Hence, we check with a small auxiliary logical sub-circuit if (exactly) one of the inputs from $\{I_{a,i} \mid a \in \Sigma\}$ is set to one. In the following description, let w denote the word of length k that is meant by a satisfying assignment.

- Moreover, there are (internal) gates $S_{q,i,p}$ for $q, p \in Q$ and $0 \leq i \leq k$, meaning that each of these gates outputs true if A , when started to digest the input w within state q , will reach state p after i steps. In particular, we can initialize $S_{q,0,p}$ to output one (true) if and only if $q = p$. For $i > 0$, the gate $S_{q,i,p}$ outputs true if and only if both $I_{a,i}$ and $S_{q,i-1,r}$ output true and $\delta(r, a) = p$.
- Finally, the output gate O collects the information of all gates $S_{q,k,p}$; it outputs true if and only if there is one $p \in Q$ such that for all $q \in Q$, $S_{q,k,p}$ outputs true.

It should be clear that the circuit that we described can be constructed in polynomial time if k is given in unary (and in FPT-time otherwise, which is good enough for our purpose). \square

A reader who knows about the definition of the W -hierarchy in terms of weft will also recognize that this construction does not prove membership within any fixed level of that hierarchy, because the number of so-called large gates on paths from input gates to the output gate grows with k .

We are now going to prove membership in yet another parameterized complexity class, namely $A[2]$. Besides the mentioned W -hierarchy, there is another hierarchy of hard parameterized problems, which is the A -hierarchy. The best explanation of this hierarchy can be found in the textbook of Flum and Grohe (2006). It is known that $W[t] \subseteq A[t]$ for each level t of the hierarchies. In particular, $W[1] = A[1]$, but no other equality is known. Moreover, it is unknown if $A[2]$ is a subset of $W[P]$, so that the following membership of DFA-SW, parameterized with the length parameter k , can be seen as an indication that our problem is not complete for $A[2]$.

Recall that also $A[2]$ possesses a characterization in terms of Turing machines. Now, we are facing 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 when trying to accept the empty word in at most k steps, where k is the parameter. By observing that the switch between phases (1) and (2) of the description of the Turing machine M in the proof of Theorem 3 can be also viewed as switching between existentially and universally quantified states, M can be also re-interpreted to show:

Theorem 5. DFA-SW, parameterized with the length parameter k , is contained in the class $A[2]$.

The latter result also follows when combining the reductions of Lemmas 14 and 15 below, because parameterized reductions are transitive as a relation between parameterized problems.

Let us now state the position of $W[\text{Sync}]$ within the better known parameterized complexity classes more formally.

Corollary 1. $W[2] \subseteq W[\text{Sync}] \subseteq A[2] \cap WNL \cap W[P]$.

5. Further Problems Complete for $W[\text{Sync}]$

In the following problem statements, k is always the parameter that we are analyzing with respect to parameterized complexity. In each case, we prove a completeness result for $W[\text{Sync}]$.

5.1 DFAs with a sink state

Our first example is an easy modification of our basic problem DFA-SW. Recall that a state s of a DFA is a *sink state* if for all input letters a , we have $\delta(s, a) = s$. Observe that if a DFA A possesses a sink state s , then A is synchronizable if and only if s is reachable from each (other) state of A ; moreover, s is then the unique possible synchronizing state.

DFA-SYNCHRONIZING WORD WITH A SINK (DFA-SW-SINK)

Input: DFA A with a sink state, $k \in \mathbb{N}$

Problem: Is there a synchronizing word w for A with $|w| \leq k$?

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

Figure 2. How to define a transition function δ' of a DFA with a sink state f .

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.*

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. An example for this construction can be found below. Let $s_0, \dots, s_k, f \notin Q$ be fresh states. Let $Q' = Q \cup \{s_0, \dots, s_k, f\}$ be the states of A' . Define the transition function δ' as described in Figure 2. 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 $w' = \sigma^{k-|w|}wq_f$ 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, \dots, 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 , that is, $w \in \Sigma^*$. 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, $|w| \leq k$. This proves the correctness of the transformation. \square

We will give an example for this proof in the following. As an example automaton, we chose the famous 4-state DFA designed by Černý (drawn in Figure 3) to illustrate that shortest synchronizing words could be surprisingly long, in this case, of length nine. More precisely, $w = baaabaaab$ leads all four states to q_1 , so that wq_1 synchronizes the resulting automaton (in our construction) if $k \leq 9$. The resulting automaton can be found in Figure 4.

Remark 1. *Let us mention one more formal language fact about synchronizing words: there is a synchronizing word of length exactly k for a DFA A if and only if there is a synchronizing word of length at most k for A . This comes from the fact that if w is a synchronizing word, then also wu is synchronizing for any word u . Therefore, we can also employ the problem variation asking about the existence of a synchronizing word of length exactly k if it is more convenient.*

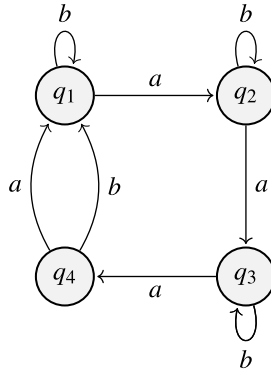


Figure 3. Černý's 4-state automaton from Černý (1964).

5.2 Synchronizing subsets

As mentioned above, the question if a given subset of states can be synchronized poses even a PSPACE-complete problem. When enriched with a length bound on the length of a subset-synchronizing word, we are back to an NP-complete problem, assuming that the length bound is given in a unary encoding. Motivated by these considerations, we consider the following parameterized problem.

DFA-SW-FROM-SUBSET
 Input: DFA A with state set Q , subset $P \subseteq Q$, $k \in \mathbb{N}$
 Problem: Is there a P -synchronizing word w for A with $|w| \leq k$?

Recall that we previously discussed this problem with the parameter $|P|$, calling it DFA-SW-FROM-SUBSET-SB for clarity.

Theorem 6. DFA-SW-FROM-SUBSET, parameterized by an upper bound k on the length of a subset-synchronizing word, is complete for $W[\text{Sync}]$.

Notice that our discussions from Remark 1 translate verbatim also to this problem variation.

Proof. As a Q -synchronizing word is just another name for a synchronizing word, it is clear that DFA-SW-FROM-SUBSET is $W[\text{Sync}]$ -hard. We prove membership in $W[\text{Sync}]$ in the following.

Consider a DFA $A = (Q, \Sigma, \delta, q_0, F)$, together with a state subset P and an integer k . We define a new DFA $A' = (Q', \Sigma', \delta', q_0, F)$ and an integer $k' = k + 1$ such that A possesses a P -synchronizing word of length k if and only if A' has a synchronizing word of length k' .

Let $\bar{P} = \{\bar{p} \mid p \in P\}$ consists of (barred) copies of the states in P and let $Q' = Q \cup \bar{P}$. Let $c \notin \Sigma$ be a fresh letter. Let $\Sigma' = \Sigma \cup \{c\}$. The transition function δ' equals δ when restricted to $Q \times \Sigma$. Now, fix some $p_{\text{fix}} \in P$. Furthermore, define $\delta'(q, c) = p_{\text{fix}}$ for $q \in Q \setminus P$ and $\delta'(\bar{p}, c) = \delta'(p, c) = p$ as well as $\delta'(\bar{p}, a) = \bar{p}$ for any $p \in P$ and $a \in \Sigma$.

Now, if w is a P -synchronizing word for A , then (by construction) cw is a synchronizing word for A' . Conversely, if v is a synchronizing word for A' , then it must contain at least one occurrence of c , because otherwise there is no way to synchronize the states from \bar{P} , as by construction, the states in \bar{P} cannot be synchronizing states. Assuming that v is of minimum length, we can deduce that v contains exactly one occurrence of c and this occurrence is at the very beginning of v . Now, after reading c and starting out from Q' , the DFA A' will be in any of the states of P , which means that if we write $v = cw$, then w is a P -synchronizing word for A . □

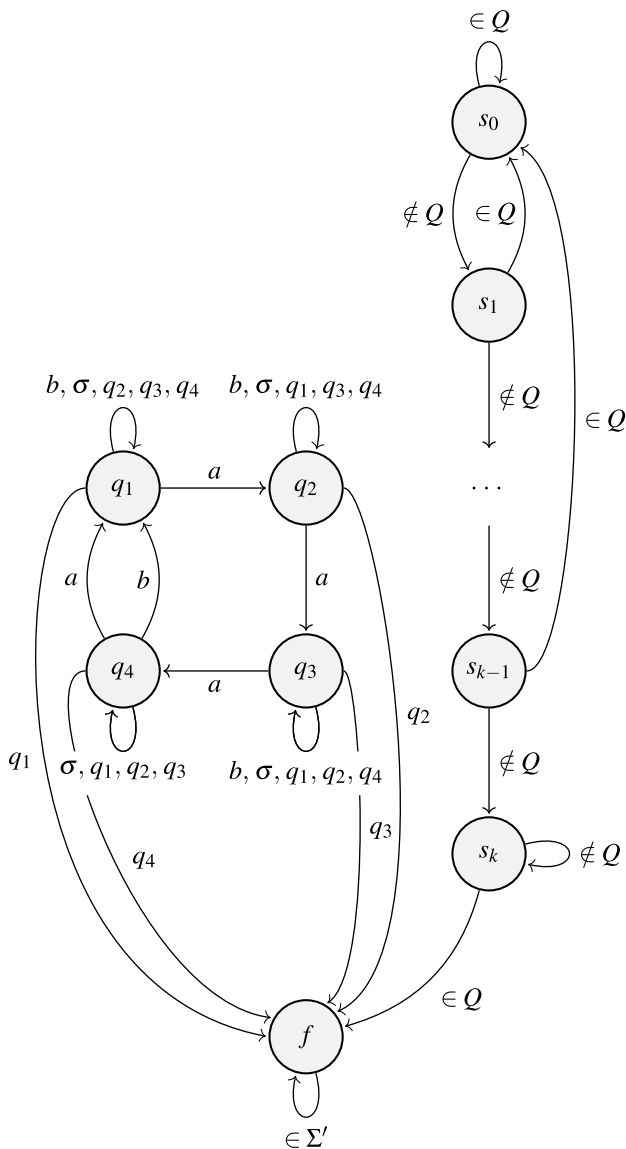


Figure 4. Černý's 4-state automaton would yield this \$(k + 6)\$-state automaton in our construction from Lemma 2.

5.3 Factoring monoids

We now present a problem originally introduced by Cai et al. (1997) in the context of parameterized complexity. As also discussed in Section 9, it is crucial that the monoid in question is implicitly given in this type of *generator problem*.

MONOID FACTORIZATION (see Cai et al. 1997)

Input: A finite set \$Q\$, a collection \$F = \{f_0, f_1, \dots, f_m\}\$ of mappings \$f_i : Q \to Q, k \in \mathbb{N}\$

Problem: Is there a selection of at most \$k\$ mappings \$f_{i_1}, \dots, f_{i_{k'}}\$, \$k' \le k\$, with \$i_j \in \{1, \dots, m\}\$ for \$j = 1, \dots, k'\$, such that \$f_0 = f_{i_1} \circ f_{i_2} \circ \dots \circ f_{i_{k'}}\$?

We can speak of k as (an upper bound on) the *generation length* of f_0 . This problem (without the generation length bound k) was introduced as GEN in the seminal work of Kozen (1977) and proven to be PSPACE-complete; it appeared as MS5 under the quite fitting name FINITE FUNCTION GENERATION in Garey and Johnson (1979, p. 280). With the generation length bound, MONOID FACTORIZATION, restricted to collections of bijections (i.e., permutations), was proven to be NP-complete in Even and Goldreich (1981); this restricted problem was termed MINIMUM GENERATOR SEQUENCE by Even and Goldreich. We will discuss further variations of MONOID FACTORIZATION concerning aspects of parameterized complexity in Section 9.

With some background knowledge on transition monoids, it is clear that by interpreting a given DFA $A = (Q, \Sigma, \delta, q_0, F)$ as a collection F_A of $|\Sigma|$ many mappings $f_a : Q \rightarrow Q$, by setting $f_a(q) = \delta(q, a)$, we can solve a DFA synchronization problem given by (A, k) by solving $|Q|$ many instances (F_q, k) of MONOID FACTORIZATION, where $F_q = \{f_0 = q\} \cup F_A$ and the aim is to represent the constant target map $f_0 = q$. The synchronization problem (A, k) is a YES-instance if some of the instances (F_q, k) are YES-instances. This shows that DFA-SW, parameterized with the length upper bound k , can be solved by a Turing reduction to MONOID FACTORIZATION, parameterized with the generation length upper bound k . Below, we will strengthen this toward presenting a many-one reduction.

Having a closer look at the proof of $W[2]$ -hardness of DFA-SW, parameterized with the length upper bound k (also discussed below in more details), which was presented in Fernau et al. (2015), we see that the resulting automaton has one sink state, and hence, this is the only possible synchronizing state. This means that the Turing reduction suggested in the previous paragraph can be indeed viewed as a many-one (Karp) reduction, as the constant target function is uniquely defined. Hence, this reasoning shows that MONOID FACTORIZATION, parameterized with the generation length upper bound k , is indeed $W[2]$ -hard. This result has been shown already in Cai et al. (1997) by a different reduction from DOMINATING SET.

Lemma 3. *There is a polynomial time computable parameterized many-one reduction from MONOID FACTORIZATION, parameterized with some generation length upper bound, to DFA-SW, parameterized with some length upper bound.*

Proof. Let $F = \{f_0, f_1, \dots, f_m\}$ be a collection of mappings $f_i : Q \rightarrow Q$ and $k \in \mathbb{N}$. Define $\hat{Q} = Q \times Q \cup \{s_0, \dots, s_k, s_{k+1}, f\}$. Let $\Sigma = \{a_1, \dots, a_m, \sigma, \tau\}$ and define the transition function $\delta : \hat{Q} \times \Sigma \rightarrow \hat{Q}$ as defined in Figure 5. This describes the interesting aspects of the automaton A_F . An illustrative example will be discussed below. 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 $f_{i_1}, \dots, f_{i_{k'}}$, $k' \leq k$, with $i_j \in \{1, \dots, m\}$ for $j = 1, \dots, k'$, such that $f_0 = f_{i_1} \circ f_{i_2} \circ \dots \circ f_{i_{k'}}$. Then, $w = \sigma^{k-k'+1} a_{i_1} \cdot a_{i_2} \dots a_{i_{k'}} \tau$ synchronizes the DFA 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 $a_{i_1} \cdot a_{i_2} \dots a_{i_{k'}}$ 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 $\{s_0, \dots, s_k, s_{k+1}, f\}$, 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 hence be 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 thereby 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 $v \in \{a_1, \dots, a_m\}^*$. By the structure of w that we analyzed before, we must have $w = u\sigma v\tau$,

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

Figure 5. Transition function δ of the constructed DFA-SW instance.

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 $p = (f_{i_1} \circ f_{i_2} \circ \dots \circ f_{i_{k'}})(q)$, from which we can enter f only (after reading τ) if $f_0(q) = p$. This shows that, if $u = a_{i_1} \cdot a_{i_2} \cdot \dots \cdot a_{i_{k'}}$, then $f_0 = f_{i_1} \circ f_{i_2} \circ \dots \circ f_{i_{k'}}$. \square

Let us give an illustrative example for the construction used above in Lemma 3. We are considering the following three mappings $f_i : Q \rightarrow Q$, with $Q = \{1, 2, 3, 4\}$.

f_i	1	2	3	4
f_0	1	1	1	1
f_1	2	3	4	1
f_2	1	2	3	1

This describes, together with some integer $k \geq 0$, our instance of MONOID FACTORIZATION. In the equivalent instance of DFA-SW, the mappings f_1 and f_2 are modeled as the action of letters, leading to a two-letter input alphabet, say, $\Sigma' = \{a, b\}$, which is enriched by two more symbols, σ, τ . In particular, σ and τ together model the target mapping f_0 (which is constant in our case), so that this construction, which can be found in Figure 6, results in another DFA capable of simulating the mentioned 4-state DFA designed by Černý (depicted in Figure 3) by an automaton with a sink state, yielding a somewhat more complicated construction (to this end) than the aforementioned one from Lemma 2 (see Figure 4).

Theorem 7. DFA-SW-SINK, parameterized by an upper bound k on the length of a synchronizing word, is complete for $W[\text{Sync}]$.

Proof. As each DFA-SW-SINK is trivially an instance of DFA-SW, the previous lemma shows the result. \square

We like to mention that there is another algebraic problem studied in the literature that looks quite similar to MONOID FACTORIZATION and that we define next:

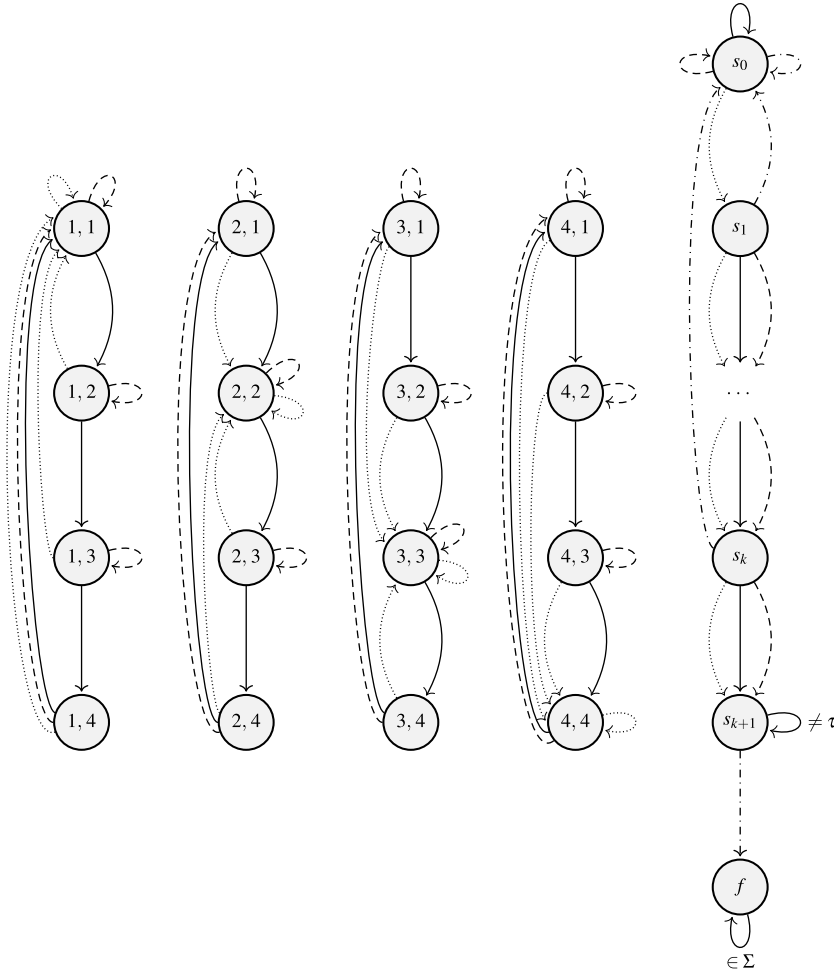


Figure 6. An example for our construction from Lemma 3. Notice the similarities of the mappings to the actions of letters in Černý automaton (Figure 3). For arcs without labeling, the following holds: Arcs carried out with a or b , respectively, in the Černý automaton (i.e., the mappings f_1 and f_2 , respectively) are solid or dashed, respectively. Arcs resulting from transitions with σ or τ , respectively, are dotted, or dotted and dashed, respectively. Note that the arcs labeled τ are omitted in all vertices (i, j) for the sake of readability. They would lead from (i, i) to f and from (i, j) to $(i, 1)$ when $j \neq i$.

BOUNDED TRANSFORMATION RANK(1) (see Goralčík and Koubek 1995)

Input: A finite set Q , a collection $F = \{f_1, \dots, f_m\}$ of mappings $f_i : Q \rightarrow Q, k \in \mathbb{N}$

Problem: Is there a selection of at most k mappings $f_{i_1}, \dots, f_{i_{k'}}$, $k' \leq k$, with $i_j \in \{1, \dots, m\}$ for $j = 1, \dots, k'$, such that $f_0 := f_{i_1} \circ f_{i_2} \circ \dots \circ f_{i_{k'}}$ satisfies $|f_0(Q)| = 1$ (i.e., rank equals 1)?

With the idea of transformation monoids in mind, it is rather straightforward to see that this problem can be viewed just as an algebraic reformulation of DFA-SW; this was also observed in Volkov (2008). We therefore only state the consequence that is interesting for our context.

Corollary 4. BOUNDED TRANSFORMATION RANK(1), parameterized by an upper bound k on the generation length, is complete for $W[\text{Sync}]$.

Goralčík and Koubek (1995) also considered the problem BOUNDED TRANSFORMATION RANK(r) for any $r \geq 1$.⁴ This means that the condition $|f_0(Q)| = 1$ is replaced by $|f_0(Q)| = r$ in the problem definition above. By adding further elements to Q on which the mappings all act as identities, one easily obtains the next result.

Corollary 5. For any $r \geq 1$, BOUNDED TRANSFORMATION RANK(r), parameterized by an upper bound k on the generation length, is hard for $W[\text{Sync}]$.

As r is part of the problem definition (and hence fixed), we can strengthen the previous two statements as follows:

Theorem 8. For any $r \geq 1$, BOUNDED TRANSFORMATION RANK(r), parameterized by an upper bound k on the generation length, is complete for $W[\text{Sync}]$.

Proof. We will show how to reduce BOUNDED TRANSFORMATION RANK(r) to BOUNDED TRANSFORMATION RANK(1), which completes the argument.

Given a set $Q = \{q_1, \dots, q_n\}$, a collection $F = \{f_1, \dots, f_m\}$ of mappings $f_i : Q \rightarrow Q$ and $k \in \mathbb{N}$ as an instance of BOUNDED TRANSFORMATION RANK(r), we construct an instance $(\hat{Q}, \hat{F}, \hat{k})$ of BOUNDED TRANSFORMATION RANK(1) as follows. Let $\hat{Q} = Q \cup \{s\} \cup [k] \cup \{c_1, \dots, c_r\}$, with pairwise disjoint unions, and $[k] = \{1, \dots, k\}$. We interpret f_i as acting on \hat{Q} by setting $f_i(s) = s$, $f_i(k) = c_1$ and $f_i(j) = j + 1$ for $j \in [k - 1]$, as well as $f_i(x) = 1$ if $x \in \{c_1, \dots, c_r\}$. We are now defining a collection \hat{F} of mappings from \hat{Q} to \hat{Q} that contains F in this sense. We add $r \cdot \mathcal{O}(|Q|^r)$ many mappings $g_{A,j} : \hat{Q} \rightarrow \hat{Q}$ into \hat{F} . Here, $A \subseteq Q$, $|A| = r$, $j \in [r] = \{1, \dots, r\}$. More precisely, $A = \{q_{\ell_1}, \dots, q_{\ell_r}\}$ with $\ell_j < \ell_{j+1}$ for $j \in [r - 1]$. For $j \in [r - 1]$, let $g_{A,j}(q_{\ell_j}) = g_{A,j}(c_j) = c_{j+1}$, but $g_{A,j}(c_i) = c_1$ for $i \in [r] \setminus \{j\}$. The case $j = r$ is covered by $g_{A,r}(q_{\ell_r}) = g_{A,r}(c_r) = s$. Moreover, $g_{A,j}(x) = g_{A,j}(i) = 1$ for any $j \in [r]$, $i \in [k]$, $x \in Q \setminus A$. In all other cases, $g_{A,j}(x) = x$. Furthermore, we add $k - 1$ increment mappings ι_j with $\iota_j(j) = j + 1$ if $j \in [k - 1]$ that act as the identity on all other elements of \hat{Q} . Finally, let $\hat{k} = k + r$ to complete our description of $(\hat{Q}, \hat{F}, \hat{k})$ constructed from (Q, F, k) . Observe that the transformation is polynomial, because r is a constant. In particular,

$$|\hat{F}| \leq m + r \cdot \binom{|Q|}{r} + k - 1.$$

If $f_{i_1}, \dots, f_{i_{k'}}$, $k' \leq k$, exist with $i_j \in \{1, \dots, m\}$ for $j = 1, \dots, k'$, such that $f_0 := f_{i_1} \circ \dots \circ f_{i_{k'}}$ satisfies $|f_0(Q)| = r$, then there is some set $A \subseteq Q$, $|A| = r$, such that $f_0(Q) = A$. Let $A = \{q_{\ell_1}, \dots, q_{\ell_r}\}$ with $\ell_j < \ell_{j+1}$ for $j \in [r - 1]$. Let $k_\delta = k - k'$. Composing first $\iota_1 \circ \dots \circ \iota_{k_\delta}$ with f_0 and then with $g_{A,1}, \dots, g_{A,r}$ gives a mapping \hat{f}_0 with $\hat{f}_0(\hat{Q}) = \{s\}$. As described, \hat{f}_0 can be expressed by composing $k + r = \hat{k}$ many mappings from \hat{F} .

If $h_1, \dots, h_{k'}$, $k' \leq \hat{k}$, exist with $h_j \in \hat{F}$ for $j = 1, \dots, k'$, such that $h_0 := h_1 \circ h_2 \circ \dots \circ h_{k'}$ satisfies $|h_0(\hat{Q})| = 1$, then by the structure of the mappings in \hat{F} , $h_0(\hat{Q}) = \{s\}$. Moreover, as executing mappings from F after executing some $g_{A,j}$ has no further effects but possibly deferring the overall procedure, we can assume that this is not case here. Assuming that the mapping composition yields h_0 , it is clear that the last mapping executed equals some $g_{A,r}$. As in particular $h_0(1) = s$, at least k other mappings must be from F or some increment mappings ι_j . As $h_0(c_1) = s$, we must have used one mapping from $\{g_{A,j} \mid A \subseteq Q\}$ for each $j \in [r]$. Due to the length bound $\hat{k} = k + r$ that is exactly met due to counting through $1, \dots, k$ and through c_1, \dots, c_r , it is clear that the maps from $\{g_{A,j} \mid A \subseteq Q\}$ were selected in order, starting with $g_{A,1}$, through $g_{A,r}$. Moreover, as

“wrong selections” are sent to 1 (which is unaffordable by the length bound \hat{k}), $A_1 \setminus \{q_{\ell_1}\} \subseteq A_2$, with q_{ℓ_1} being the first element of A_1 , $A_2 \setminus \{q_{\ell_2}\} \subseteq A_3$, with q_{ℓ_2} being the second element of A_2 , etc. means that finally a set $A = \{q_{\ell_1}, q_{\ell_2}, \dots\}$ is selected with $|A| = r$, a set with which we could have started alternatively. Therefore, we can assume $A = A_1 = A_2 = \dots = A_r$. Hence, we can assume that h_0 is composed as $h_0 = f \circ g$, where $f = h_1 \circ \dots \circ h_k$, such that h_1, \dots, h_{k_δ} are increments, $h_{k_\delta+1}, \dots, h_k \in F$ and $g = h_{k+1} \circ \dots \circ h_{k+r}$ with $h_{k+j} = g_{A,j}$ for some $A \subseteq Q$. Now, leaving out all mappings $g_{A,j}$ and all possibly contained increment mappings, we are left with a sequence of $k' \leq k$ many mappings from F that can be re-interpreted as acting on Q . As their composition f_0 , applied to Q , equals A , with $|A| = r$, the reverse implication is shown. \square

Finally, notice that due to the tight connections to automata theory, the previous proof can be also interpreted as a W[Sync]-completeness result for the following family of problems RANK(r) DFA-SYNCHRONIZING WORD (with $r \geq 1$): Given a DFA $A = (Q, \Sigma, \delta)$ and a number k , does there exist a word $w \in \Sigma^{\leq k}$ such that $|\delta(Q, w)| = r$? This is also true for the problem variation where we ask whether $|\delta(Q, w)| \leq r$ instead. Looking back at the previous proof, this would mean that it is not that important to really count through all elements of up to r to ensure the exact cardinality of the chosen set; smaller sets would suffice.

5.4 Intersecting regular languages

BOUNDED DFA-INTERSECTION (see Wareham 2001)

Input: A set \mathcal{A} of deterministic finite automata with the same input alphabet Σ , $k \in \mathbb{N}$

Problem: Is there a $w \in \Sigma^*$ of length k accepted by all automata in \mathcal{A} ?

Here, we took over the definition from Wareham (2001), but as we can also see from the discussion in Remark 1, in our case it is not that important if we ask for a word exactly or at most k .

Theorem 9. BOUNDED DFA-INTERSECTION, parameterized by an upper bound on the length of the commonly accepted string, is complete for W[Sync].

Previously, Wareham (2001) only proved W[2]-hardness 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 $A_q = (Q, \Sigma, \delta, q, \{s_f\})$ and consider the DFA collection $\mathcal{A} = \{A_q \mid q \in Q\}$. 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 .⁵

Conversely, if $\mathcal{A} = \{A_i \mid 1 \leq i \leq \ell\}$ 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 $\Sigma' = \Sigma \cup \{\sigma, \tau\}$. Let $Q' = \left(\bigcup_{i=1}^{\ell} Q_i\right) \cup \{s_0, \dots, s_k, s_{k+1}, f\}$ be the state set of the DFA A that we construct. Define the transition function δ as in Figure 7.

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 3, 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 ,

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

Figure 7. Transition function δ of the constructed DFA-SW instance.

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 could also discuss the variant when we ask for a word of length at most k to be accepted by all automata in \mathcal{A} . However, from the discussions of Remark 1, it immediately follows that also this variant is complete for $W[\text{Sync}]$. Other variations show up when we switch to nondeterministic finite automata or to regular expressions for representing regular languages in these intersection problems. In these cases (and without defining the problems formally), we can only state the following observations.

Corollary 6. *BOUNDED NFA-INTERSECTION, parameterized by an upper bound on the length of the commonly accepted string, is hard for $W[\text{Sync}]$.*

Conversely, it would be interesting to discuss subregular language families for this type of problem, with the question in mind if we (still) obtain $W[\text{Sync}]$ -complete problems or not. Notice that (for instance) in the $W[2]$ -hardness proof of Wareham (2001, Lemma 6) concerning BOUNDED DFA-INTERSECTION, only very simplistic automata have been used. This also means that the corresponding regular languages can be easily expressed by simple regular expressions, leading to the following result.

Corollary 7. *BOUNDED REGULAR-EXPRESSION-INTERSECTION, parameterized by the length of the commonly accepted string, is hard for $W[2]$.*

The exact position of BOUNDED NFA-INTERSECTION and of BOUNDED REGULAR-EXPRESSION-INTERSECTION, in particular with respect to $W[\text{Sync}]$, poses open questions. It would be also interesting to discuss further restrictions, leading to subclasses of NFA or of regular expressions, where it might be easier to prove membership in $W[\text{Sync}]$. It might be also an idea to quantify the degree of nondeterminism in these studies; we refer to Goldstine et al. (1992). In a sense, the preceding discussions are continued in the next sections, where we discuss other problems between $W[2]$ and $W[\text{Sync}]$ or hard for $W[\text{Sync}]$, where again the precise parameterized complexity status is unknown.

6. Problems between $W[2]$ and $W[\text{Sync}]$

One natural way to get problems below $W[\text{Sync}]$ is to consider only restricted variants of automata as admissible instances of DFA-SW. For instance, above we considered DFA-SW-SINK. However,

it is not always clear that we arrive at problem variants that are complete for $W[\text{Sync}]$. For instance, Bruchertseifer and Fernau (2021) showed that the further restriction to so-called TTSP(L) automata (where the automaton multi-graphs can be described as two-terminal series-parallel multi-graphs with loops) results in a variation of DFA-SW that is still $W[2]$ -hard, which we were able to concretize in the long version to the effect that it is “only” contained in $W[\text{Sync}]$. In actual fact, it is an open problem if this problem variant is contained in $W[2]$ or $W[\text{Sync}]$ -hard, or really something “in-between.” As shown by Möhring (1989), 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 $Q \rightarrow Q$ that can be associated to input letters are monotone with respect to the series-parallel partial order corresponding to the TTSP(L) automaton graph. Our earlier constructions show:

Corollary 8. *DFA-SW, restricted to DFAs with TTSP(L) automata graphs, is 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 might indicate that we have found yet another class of parameterized problems living inside the by now classical parameterized complexity classes. However, this is just another open problem. As surveyed in Martyugin (2009), there are quite a number of subregular families of DFAs whose length-bounded synchronization problem stays NP-complete. A concise study of the corresponding parameterized complexities is still lacking. These are good further candidates of problems that are possibly complete for $W[\text{Sync}]$, yielding lots of open problems.

But, there are further problems lying between $W[2]$ and $W[\text{Sync}]$, as we will see next.

6.1 Satisfying constraint satisfaction formulae

We consider constraint satisfaction formulae in conjunctive normal form, or CSP CNF for short, in the following. A CSP CNF formula φ on k variables x_1, \dots, x_k is given by a finite universe A , atomic sentences $x_i = a$ for $1 \leq i \leq k$ and $a \in A$ and a CNF built from these atomic sentences. Hence, for $A = \{0, 1\}$, this is equivalent to a classical Boolean CNF formula.

CSP CNF SATISFIABILITY

Input: A CSP CNF formula φ on k variables x_1, \dots, x_k given by a finite universe A , atomic sentences $x_i = a$ for $1 \leq i \leq k$ as well as $a \in A$ and a CNF built from these atomic sentences

Problem: Is φ satisfiable?

The Rystsov-Eppstein proof for NP-hardness of DFA SYNCHRONIZATION given by Rystsov (1980) and Eppstein (1990) can be generalized to cope with CSP CNF formulae, preserving the parameter. This observation already proves:

Lemma 9. $\text{CSP CNF SATISFIABILITY} \in W[\text{Sync}]$. □

What about the converse? We do not know, but we can prove instead the following hardness result.

Lemma 10. $\text{CSP CNF SATISFIABILITY}$ is $W[2]$ -hard.

Proof. We reduce from HITTING SET, a standard $W[2]$ -complete problem. Let U be a universe and let \mathcal{S} be a set system over U . Moreover, k is the parameter. The task is to choose k elements $u_1, \dots, u_k \in U$ such that for each $S \in \mathcal{S}$, $u_i \in S$ for some $1 \leq i \leq k$. To express this task as a CSP CNF formula with variables x_1, \dots, x_k , we simply add the clauses

$$\bigvee_{i=1}^k \bigvee_{s \in S} x_i = s$$

for each $S \in \mathcal{S}$. Now, it should be clear that we can choose at most k different elements $u_1, \dots, u_k \in U$ to cover all sets in the set system \mathcal{S} if and only if the resulting CSP CNF formula (over the universe U) is satisfiable. □

6.2 Long subsequences

We now discuss the well-known LONGEST COMMON SUBSEQUENCE problem.

LONGEST COMMON SUBSEQUENCE
 Input: A set of ℓ strings x_1, \dots, x_ℓ over an alphabet Σ
 Problem: Is there a string $w \in \Sigma^k$ occurring in each of the x_i as a subsequence?

As explained in Wareham (2001), 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 11. LONGEST COMMON SUBSEQUENCE $\in W[\text{Sync}]$.

Observe that we can also use the previous proposition to prove Lemma 9, because one can modify the proof of Theorem 3 in Bodlaender et al. (1995) to show the reduction CSP CNF SATISFIABILITY \leq_{FPT} LONGEST COMMON SUBSEQUENCE. Unfortunately, we do not know if LONGEST COMMON SUBSEQUENCE is also hard for $W[\text{Sync}]$. We only know $W[2]$ -hardness from Bodlaender et al. (1995); further membership results were unknown hitherto, so the previous proposition remedies this situation a bit.

6.3 Discussions

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 11 are all accepting finite languages. Is there a converse reduction from such a BOUNDED DFA-INTERSECTION instance to some LONGEST COMMON SUBSEQUENCE instance? Might this open question lead to another interesting complexity class between $W[2]$ and $W[\text{Sync}]$?

Let us also mention that Guillemot (2011) has shown that LONGEST COMMON SUBSEQUENCE, parameterized by the number of strings ℓ , is complete for WNL. Likewise, BOUNDED DFA-INTERSECTION, parameterized by the number of automata, is complete for WNL. Hence, whether or not two problems are FPT-equivalent clearly depends on the chosen parameterization.

Of course, having found two concrete problems between $W[2]$ and $W[\text{Sync}]$ leads to the natural open question if they are mutually reducible. As mentioned above, we only know that CSP CNF SATISFIABILITY \leq_{FPT} LONGEST COMMON SUBSEQUENCE. Further examples of problems below $W[\text{Sync}]$ can be obtained by restricting BOUNDED DFA-INTERSECTION to subclasses of DFA. Notice that also for several such restricted classes of DFA, the basic problem is still NP-hard; see Arrighi et al. (2021) for a recent study.

7. Problems Hard for W[Sync]

In the following problem statements, k is always the parameter that we are analyzing with respect to parameterized complexity. In each case, we prove a hardness result for W[Sync]. We remind the reader about the unknown status of BOUNDED NFA-INTERSECTION. We also discuss memberships in WNL, A[2], A[3], and W[P].

7.1 Intersecting regular languages again

Lemma 12. BOUNDED NFA-INTERSECTION, parameterized by the length of the commonly accepted string, is contained in WNL.

Proof. We only sketch this proof, as it parallels previous ones. Let \mathcal{A} be a collection of NFAs and $k \in \mathbb{N}$. We can design a nondeterministic one-tape Turing machine M (starting on the empty input) that first guesses a string w of length k and then verifies, for each automaton $A \in \mathcal{A}$, that A accepts w . Only if all these simulations succeed, M will accept. As required, M needs only space k . □

However, it is not clear to us how we could limit the number of guessing steps in the simulation of the previous proof; that is, it is an open question if BOUNDED NFA-INTERSECTION belongs to W[P]. Likewise, it is an open question if BOUNDED NFA-INTERSECTION belongs to A[2], because it is not clear how to make this simulation work with only one switch from existential to universal states. Without going into details here, let us mention that with one further switch from universal to existential states, an alternating Turing machine could be designed that simulates a given BOUNDED NFA-INTERSECTION instance in time $f(k)$, which gives (see Flum and Grohe 2006 for details) the following result.

Lemma 13. BOUNDED NFA-INTERSECTION, parameterized by the length of the commonly accepted string, is contained in A[3].

7.2 Extensions and orderings of words

Extension variants of SYNCHRONIZING WORD have been studied before, as in Fernau and Hoffmann (2019). The underlying problem, which depends on the choice of a partial order $<$ on the set of all words, is defined as follows:

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

Please note that the complexity varies fundamentally with the choice of the partial order. In the following, we will concentrate on the *length-lexicographical ordering* \leq_{ll} as the partial order, where $v \leq_{ll} w$ means that either $|v| < |w|$ or that $|v| = |w|$ and $v \leq_{lex} w$, where $v \leq_{lex} w$ refers to a lexicographical (total) order induced by a given total order on the alphabet. In Fernau and Hoffmann (2019) and in Bruchertseifer and Fernau (2021), we are also discussing other (natural) partial orders. It should be noted that in some cases, the extension variants are solvable in polynomial time, while other cases lead to NP- or co-NP-hard problems. In particular, there is a co-NP-hardness of EXT DFA-SW- \leq_{ll} . To avoid clumsy formulations, we will therefore consider the NP-hard problem CO EXT DFA-SW- \leq_{ll} instead that reverses YES with NO answers compared to EXT DFA-SW- \leq_{ll} .

Theorem 10. $\text{CO EXT DFA-SW-}\leq_{ll}$, parameterized by $|u|$, is contained in $\text{WNL} \cap \text{W}[P] \cap \text{A}[2]$, but one can reduce DFA-SW , parameterized by a length upper bound k , to $\text{CO EXT DFA-SW-}\leq_{ll}$, parameterized by $|u|$.

Proof. For membership $\text{CO EXT DFA-SW-}\leq_{ll} \in \text{WNL} \cap \text{W}[P] \cap \text{A}[2]$, we can modify the according proofs of Theorems 3, 4, or 5, constructing a nondeterministic Turing machine M as follows, given A and u . As in the previous construction, 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, which can be done in polynomial time according to Sandberg (2005) and Volkov (2008). 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 . Remind that according to the previous tests, $u \leq_{ll} v$ must hold.

We are now turning to the second claim. Consider a DFA A on the input alphabet Σ , together with a number 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 $\text{CO EXT DFA-SW-}\leq_{ll}$. 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 $x \in \Sigma$ toward 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 $\text{CO EXT DFA-SW-}\leq_{ll}$. If \hat{A} has a synchronizing word w of length at most 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

The natural open question is if $\text{CO EXT DFA-SW-}\leq_{ll}$ is in $\text{W}[\text{Sync}]$.

In Bruchertseifer and Fernau (2021), we also showed that $\text{EXT DFA-SW-}\leq_{|}$ is $\text{W}[3]$ -hard, where $|$ refers to the (scattered) subsequence ordering. But it is an open question if this extension problem belongs to $\text{W}[3]$. Notice that there are few natural parameterized problems higher up in the W -hierarchy; see Chen and Zhang (2006). However, the very idea of extension problems (not only applicable to formal language problems) seems to lead to such problems; we also refer to Bläsius et al. (2019), Casel et al. (2018).

7.3 Non-universality questions for NFAs

BOUNDED NFA NON-UNIVERSALITY

Input: A nondeterministic finite automaton $A = (Q, \Sigma, \delta, Q_0, F)$, $k \in \mathbb{N}$

Problem: Is there a word $w \in \Sigma^k$ that is not accepted by A ?

Lemma 14. **BOUNDED NFA NON-UNIVERSALITY** is $\text{W}[\text{Sync}]$ -hard.

Proof. We reduce from **BOUNDED DFA-INTERSECTION**. Let \mathcal{A} be a set of deterministic finite automata with the same input alphabet Σ . To be more precise, let $\mathcal{A} = \{A_1, \dots, A_n\}$ with $A_i = (Q_i, \Sigma, \delta_i, q_{i,0}, F_i)$ for $1 \leq i \leq n$. Without loss of generality, let $Q_i \cap Q_j \neq \emptyset$ imply that $i = j$. Define $Q = \bigcup_{i=1}^n Q_i$. Let $A = (Q, \Sigma, \delta, Q_0, F)$ be an NFA that is defined as follows. The set of initial states Q_0 equals $\{q_{i,0} \mid 1 \leq i \leq n\}$. The final states are $F = \bigcup_{i=1}^n (Q_i \setminus F_i)$. Interpreting the

function δ_i as a set of triples (q_i, a, p_i) , with $q_i, p_i \in Q_i$ and $a \in \Sigma$, we can define the relation δ as $\delta = \bigcup_{i=1}^n \delta_i$. Now, $w \in \Sigma^*$ is accepted by all DFAs A_i if and only if w is not accepted by A , because the state sets Q_i are pairwise disjoint. \square

Observe that the NFA A constructed in the preceding proof is very special, as its transition relation is actually a mapping. The only source of nondeterminism comes from the fact that A has n initial states. Alternatively, we could create a new single initial state q_0 and add transitions (q_0, a, p_i) whenever there is a transition $(q_{i,0}, q, p_i) \in \delta_i$. This way, we can get an NFA A' with the property that for any $w \in \Sigma^+$, w is accepted by all A_i if and only if w is not accepted by A' . Notice that now there is only one state (namely q_0) that can be seen as the source of nondeterminism of A' . In particular, this proves that the number of “nondeterministic states” is not a useful (additional) parameter. Moreover, if an NFA with a single initial state and no “nondeterministic state,” that is, a DFA, is given, the question of the existence of a word of length k that is not accepted by this automaton becomes easy (solvable in polynomial time) by a product automaton construction, reducing it to the question of testing a single DFA for non-emptiness.

Lemma 15. BOUNDED NFA NON-UNIVERSALITY belongs to $A[2]$.

Proof. We describe the work of an alternating single-tape Turing machine M that can be produced from a given NFA $A = (Q, \Sigma, \delta, Q_0, F)$ and $k \in \mathbb{N}$ in polynomial time, such that M accepts the empty word in $2k$ steps if and only if there is a word $w \in \Sigma^k$ that is not accepted by A . In the following, we describe how M works, and this way we also give sufficient details on how to construct M . The state set of M consists of $P = \{p_0, \dots, p_k\}$ and Q (the state set of the given NFA), where $P \setminus \{p_k\}$ are existential states and $Q \cup \{p_k\}$ are universal states. The state p_0 is the initial state of M . The set of final states of M is $Q \setminus F$.

- First, M writes k letters from Σ on its tape, moving its head from left to right. In this phase, M traverses the states p_0, \dots, p_k in order. It ignores the tape contents (which should be empty anyways in the beginning).
- After entering p_k , M reads the tape contents backwards, until it reaches the left end of the tape (again). More precisely, when being in state p_k , M ignores its tape contents, and stays where it is, but moves from state p_k (universally) to any of the states of Q_0 .
- When being in a state q from Q , M does the following:
 - It reads a tape symbol a .
 - If $(q, a, p) \in \delta$, M moves to state p .
 - The tape head moves one step to the left.
- When M detects the right border of the tape, it stops working and accepts if it is in a state from $Q \setminus F$.

As the states of $Q \cup \{p_k\}$ are universal, all possible computation paths of A are checked if they do not accept the guessed word $w \in \Sigma^k$. Therefore, the described reduction works. \square

Notice that there are several aspects of the reduction described in the previous proof that might need small adaptations if we use a different concrete model of an alternating Turing machine. For instance, we (implicitly) assumed a Turing machine model where the tape is only (potentially) infinite to the right, but bounded to the left, and the left border could be detected. If we assume a tape that is (potentially) infinite to both sides, then we might need two more steps (of the Turing machine) to implement a left border by first printing a special border symbol and reading it in the end. But these details are inessential for our complexity result.

It is unclear to us (and hence an open question) if BOUNDED NFA NON-UNIVERSALITY belongs to WNL or to $W[P]$, because in both cases, we face a model of nondeterministic Turing

machine (if we follow the Turing way paved by Cesati 2003) that cannot cope with checking all possible nondeterministic ramifications of the given NFA, in particular, because the step bounds are given in unary. This leads to the intuition that BOUNDED NFA NON-UNIVERSALITY is indeed “harder” than DFA-SW. However, we do not see either why BOUNDED NFA NON-UNIVERSALITY should be hard for WNL or for W[P]. For instance, in order to prove WNL-hardness, one could start with a BOUNDED DFA-INTERSECTION instance (\mathcal{A}, k) , now parameterized by $|\mathcal{A}|$. However, in any classical construction linking BOUNDED DFA-INTERSECTION to BOUNDED NFA NON-UNIVERSALITY, there is no obvious connection between $|\mathcal{A}|$ and the string length parameter of BOUNDED NFA NON-UNIVERSALITY.

8. Problems that Feel Similar to Synchronizing DFAs

In this section, we like to collect problems that are $W[2]$ -hard and belong at least to some of the classes $A[2]$, $W[P]$, or WNL, but where the relation to $W[\text{Sync}]$ is unknown. Alas, we can present only one concrete problem, again related to synchronization.

Türker and Yenigün (2015) asked to extract a synchronizable sub-automaton that is as small as possible, obtained by deleting letters from its specification. This notion of a sub-automaton is of particular interest to us, as we are dealing with completely specified deterministic automata, and DFAs are not closed under most other notions of sub-automaton one might come up with, because they tend to produce incomplete automata. Türker and Yenigün formalized this idea as a weighted minimization problem. For our purposes, it is sufficient to consider the following unweighted variant:

DFA-MSS (referring to a minimum synchronizable sub-automaton)

Input: DFA A with input alphabet Σ , $k \in \mathbb{N}$

Problem: Is there a sub-alphabet $\hat{\Sigma} \subseteq \Sigma$, $|\hat{\Sigma}| \leq k$, such that the restriction of A to $\hat{\Sigma}$ is synchronizable?

Interestingly, Türker and Yenigün used nearly the same reduction as Fernau, Heggernes, and Villanger in Fernau et al. (2015) for a different purpose to prove the following result.

Theorem 11. *DFA-MSS is NP-complete.*

From that reduction, we can observe the following.

Corollary 16. *DFA-MSS, parameterized by the upper bound k on the size of the selected alphabet, is $W[2]$ -hard.*

As with the other $W[2]$ -hard problems considered in this paper, membership in $W[2]$ is open. Also in this case, we can prove membership in WNL, although this is not completely trivial this time.

Theorem 12. *DFA-MSS is contained in $WNL \cap W[P]$.*

Proof. Let (A, k) be an instance of DFA-MSS. Notice that in a first preprocessing step, we can eliminate letters a' that act the same on the state set Q as another letter a that we decide to keep. Such a rule can be implemented to run in polynomial time, as we simply loop over all pairs of letters, so that we can now assume to face an automaton A with input alphabet Σ and state set Q such that $|\Sigma| \leq |Q|^{|Q|}$. Moreover, we can assume that Σ contains more than $\log(|Q|)$ many symbols, as otherwise we can test all subsets of Σ (for synchronizability) in polynomial time. Finally, we can test in polynomial time if A itself is synchronizable at all, as proven in Sandberg (2005) and

Volkov (2008). Recall that the algorithm checking synchronizability tests if any pair of states can be synchronized within at most $|Q|$ steps. We adapt this strategy in the following.

Now, we hard-wire the DFA into a Turing machine M as described in the following. In particular, this means that this Turing machine can keep track of pairs of states, say, (q, p) and update this information toward (q', p') in its finite memory (alternatively, on the tape, using state letters) upon reading a symbol $a \in \Sigma$, such that q' is reached from q (in the given DFA) and p' is reached from p upon reading a . We also assume a fixed linear ordering $<$ on the state set Q , and moreover, we assume that $Q \cap \Sigma = \emptyset$. Let q_a and q_b be the smallest and second-to-smallest states in Q .

The Turing machine M first guesses a sub-alphabet Σ' by writing the corresponding k input letters on its tape. Moreover, it writes down q_a next to q_b . After reading (and memorizing) the current state pair (q, p) , initially $(q, p) = (q_a, q_b)$, the machine (nondeterministically) reads one of the k guessed input letters on its tape and transfers to (q', p') in its internal memory. It continues doing so until it reaches a pair (r, r) for some $r \in Q$. If it reaches such a pair, it is verified that the pair (q, p) written on the tape can be synchronized. It continues by incrementing p on the tape (according to the linear order $<$) and then testing the synchronizability of the new pair of states on the tape with respect to the guessed sub-alphabet. M loops until the largest element of Q (with respect to $<$) is reached. Then, it would increment the first component q of the pair and set the second component of the pair of states to the smallest state larger than the first component's state; then, again the inner loop is entered. Finally, when reaching the largest state in the first component, the procedure terminates, this way verifying that the automaton is indeed synchronizable when restricted to the guessed alphabet.

Obviously, the machine M uses only $k + c$ space for some constant c , depending upon details of the implementation. This proves that DFA-MSS belongs to WNL.

Toward proving membership in $W[P]$, recall that two states (q, q') are synchronizable if and only if there is a path in the following directed auxiliary graph from (q, q') to the target vertex t : $V = Q \times Q \cup \{t\}$ is the set of vertices. There is an arc from (q, q') to (p, p') if there is some letter a such that the given DFA makes transitions from q to p and from q' to p' upon reading the letter a . Moreover, there are arcs from any vertex (q, q) to t . In our case, after having guessed the sub-alphabet, this information can be used to construct a sub-graph of the graph just described and write its description down on the tape (in polynomial time). Now, we can use, for example, Dijkstra's algorithm to decide if there is such a path in this sub-graph, deterministically looping through all pairs of states. This describes a Turing machine M' that first makes k guesses and then works deterministically a polynomial number of steps and hence proves membership in $W[P]$. \square

The reader might have expected in the previous proof that logarithmic space would be needed by a Turing machine that (basically) verifies reachability in a graph with $|Q|^2$ many vertices. Notice that this is implicit in the model, because the Turing machine writes down letters of an arbitrary alphabet on its tape, not just bits. Breaking down to the bit level, one observes that indeed $2 \log_2(\lceil |Q| \rceil) + k \log_2(\lceil |\Sigma| \rceil)$ many bits are necessary to write down the letters used by the Turing machine in the previous construction.

Although we found similar parameterized complexity results for DFA-MSS as for DFA-SW, we are not aware of further close links between both problems. In particular, it remains an open question how to solve one problem with the help of the other (respecting our choice of parameters). In particular, the first Turing machine constructed in the previous proof (or straightforward re-interpretations) does not show membership in $W[P]$, because the number of nondeterministic (guess) steps is not bounded in k (so that we had to design a second Turing machine that worked in a different way), nor in $A[2]$, since after guessing k letters, it would enter a universal state (to ensure that all pairs of states are tested for synchronizability), but then the synchronizing word should be guessed again, that is, now this machine would enter again some existential states. Even then, the number of steps is not bounded by a function in k , so that this does not prove membership

in A[3].⁶ Furthering this question, we do not know of any fixed level of the A-hierarchy to which this problem belongs.

9. Further Comments and Discussions 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:

MONOID FACTORIZATION (Variation)

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 $B \subseteq M$, $k \in \mathbb{N}$

Problem: Is there a selection of at most k elements $b_1, \dots, b_{k'}$, $k' \leq k$, from B , such that $t = b_1 \circ b_2 \circ \dots \circ b_{k'}$?

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 $Q \rightarrow Q$ 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 $F' = \{f_1, \dots, f_m\}$. 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 (even in NL, see the discussion in Barrington et al. 2001). 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 $\{X_0, X_1, \dots, X_m\}$ of a given finite set S as an input, and the question is to pick at most k sets from $\{X_1, \dots, X_m\}$, say, $X_{i_1}, \dots, X_{i_{k'}}$, 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 Bodlaender et al. (1995), Downey and Fellows (2013). In the classical complexity context, we refer to Even and Goldreich (1981) (where the problem has the name MINIMUM GENERATOR SEQUENCE) and Jerrum (1985).

Let us also mention that from the 1980s onwards, Martin Beaudry and his colleagues led a whole research agenda, looking at what they called the MEMBERSHIP PROBLEM, which meant to look at special cases of MONOID FACTORIZATION; in many cases, they still obtain NP-hardness. We only mention here monoids with threshold (at least) two and point to Beaudry (1988, 1994).⁷ It should be noted that the question whether or not an upper bound on the length of the factorization is given makes a difference for various special cases if it comes to classical complexity. None of these monoid classes has been studied from the angle of parameterized complexity. Further problems of this type are discussed in Böhler et al. (2005). To the best of our knowledge, none of these has been examined from the viewpoint of parameterized complexity and hence give a number of open questions. To mention one concrete problem of this type: Let $\{c_1, \dots, c_m\}$ be a collection of clauses, with variables $\{x_1, \dots, x_n\}$, and let $k \in \mathbb{N}$. The question is if the empty clause can be derived within the resolution proof system within k resolution steps.

Let us finally remark that the discussion of implicit or explicit representations also extends to finite automata. Namely, it is well-known that the question if a given DFA accepts a word of length k is easy to check, while DFA-SW is NP-complete. Yet, we can build a DFA A' from a DFA A with the classical power automaton construction, see Sandberg (2005), such that A' accepts a word

of length k if and only if A possesses a synchronizing word of length k . In other words, $L(A')$ can be viewed as an implicitly presented regular language. This type of presentation tends to make problems harder.

10. Open Questions and Concluding Thoughts

Throughout the paper, we already highlighted several concrete open questions. Let us now draw the reader's attention to three more general questions.

- Are there any relationships between $W[\text{SAT}]$ and $W[\text{Sync}]$?
- Are there further problems in or hard for $W[\text{Sync}]$? We explicitly mention the intersection problems from Sections 5.4 and 7 again here.
- Is there any relation between the class WNL and the A -hierarchy?

There is some evidence that $W[\text{Sync}]$ is different from $A[2]$. Namely, if $W[\text{Sync}] = A[2]$, then $A[2] \subseteq \text{para-NP}$, because $W[\text{Sync}] \subseteq \text{para-NP}$. As shown by Haan (2016), then all Σ_2^P -problems would be as "easy" as SAT from an algorithmic point of view, something which contradicts at least the practical experience with this type of problems.

It is also interesting to observe that most of our problems come from the area of formal languages, in most cases, from automata theory. This area has been a bit neglected from the perspective of parameterized complexity, see Fernau (2019) for discussing further questions in this regard. Also, the class WNL mostly hosts formal language problems, as already exhibited by Guillemot (2011). It would be interesting to see further problems situated in WNL or in $W[\text{Sync}]$ coming from other areas. As problems related to string problems can be found in computational biology and also in computational social choice, we expect these to be good candidate areas to look into in the future.

Acknowledgements. We are grateful for discussions on the topic of this paper with several colleagues. Further developments concerning the parameterized complexity of automata problems can be found in Fernau et al. (2021).

Notes

- 1 We refer to discussions in the mentioned Special Issue about the real origins of that conjecture that in fact cannot be found in Černý's paper.
- 2 Appropriate for parameterized algorithms is using the \mathcal{O}^* -notation; for instance, a problem solvable in time $\mathcal{O}^*(f(k))$ refers to an algorithm running in time $\mathcal{O}(f(k)p(n))$, where n is the input size, p is some polynomial, f is some arbitrary (computable) function and k is the so-called parameter, some secondary measurement of the input.
- 3 Here, some definitorial details might change the constants, e.g., how 'fast' can a Turing machine detect that it is at the tape end and turn back?
- 4 They did not consider the length bound restriction, but they did so for the restriction to idempotent target mappings; we drop the idempotency condition in our discussions to make our reasoning a bit less technical. Notice that even without the length bound, TRANSFORMATION RANK(r) is NP-hard for any $r \geq 2$, see Goralčík and Koubek (1995).
- 5 We refer to Remark 1 for a discussion on asking for synchronizing words of length exactly or at most k .
- 6 For a formal treatment of the class $A[3]$, we refer to Flum and Grohe (2006). It is known that $W[3] \cup A[2] \subseteq A[3]$.
- 7 To $f : X \rightarrow X$, one can associate integers $t \geq 0$ and $p > 0$ (threshold and period), which are the smallest numbers such that $f^{t+p} = f^t$. If $t = 0$, then f is a permutation.

References

- Arrighi, E., Fernau, H., Hoffmann, S., Holzer, M., Jecker, I., de Oliveira Oliveira, M. and Wolf, P. (2021). On the complexity of intersection non-emptiness for star-free language classes. In: Bojanczyk, M. and Chekuri, C. (eds.) *41st IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS*, LIPIcs, vol. 213, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 34:1–34:15.

- Barrington, D. A. M., Kadau, P., Lange, K. and McKenzie, P. (2001). On the complexity of some problems on groups input as multiplication tables. *Journal of Computer and System Sciences* **63** (2) 186–200.
- Beaudry, M. (1988). Membership testing in commutative transformation semigroups. *Information and Computation (formerly Information and Control)* **79** (1) 84–93.
- Beaudry, M. (1994). Membership testing in threshold one transformation monoids. *Information and Computation (formerly Information and Control)* **113** (1) 1–25.
- Berlinkov, M. V. (2014). Approximating the minimum length of synchronizing words is hard. *Theory of Computing Systems* **54** (2) 211–223.
- Bläsius, T., Friedrich, T., Lischeid, J., Meeks, K. and Schirneck, M. (2019). Efficiently enumerating hitting sets of hypergraphs arising in data profiling. In: *Algorithm Engineering and Experiments (ALENEX)*, SIAM, 130–143.
- Bodlaender, H., Downey, R. G., Fellows, M. R. and Wareham, H. T. (1995). The parameterized complexity of sequence alignment and consensus. *Theoretical Computer Science* **147** 31–54.
- Böhler, E., Glaßer, C., Schwarz, B. and Wagner, K. W. (2005). Generation problems. *Theoretical Computer Science* **345** (2–3) 260–295.
- Bruchertseifer, J. and Fernau, H. (2020). Synchronizing words and monoid factorization: A parameterized perspective. In: Chen, J., Feng, Q. and Xu, J. (eds.) *Theory and Applications of Models of Computation, 16th International Conference, TAMC*, LNCS, vol. 12337, Springer, 352–364.
- Bruchertseifer, J. and Fernau, H. (2021). Synchronizing series-parallel deterministic automata with loops and related problems. *RAIRO Informatique théorique et Applications/Theoretical Informatics and Applications* **55** (7) 1–24.
- Cai, L., Chen, J., Downey, R. and Fellows, M. (1997). On the parameterized complexity of short computation and factorization. *Archive for Mathematical Logic* **36** 321–337.
- Casel, K., Fernau, H., Ghadikolaei, M. K., Monnot, J. and Sikora, F. (2018). On the complexity of solution extension of optimization problems. CoRR, abs/1810.04553.
- Černý, J., Pirická, A. and Rosenauerová, B. (1971). On directable automata. *Kybernetika* **7** (4) 289–298.
- Černý, J. (1964). Poznámka k homogénnym experimentom s konečnými automatmi. *Matematicko-fyzikálny časopis* **14** (3) 208–216.
- Černý, J. (2019). A note on homogeneous experiments with finite automata. *Journal of Automata, Languages and Combinatorics* **24** (2–4) 123–132.
- Cesati, M. (2003). The Turing way to parameterized complexity. *Journal of Computer and System Sciences* **67** 654–685.
- Chen, J. and Zhang, F. (2006). 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.
- de Haan, R. (2016). *Parameterized Complexity in the Polynomial Hierarchy*. Phd thesis, Faculty of Informatics at the Technische Universität Wien, Austria.
- Downey, R. G. and Fellows, M. R. (2013). *Fundamentals of Parameterized Complexity*, Texts in Computer Science, Springer, Cham, Switzerland.
- Elberfeld, M., Stockhusen, C. and Tantau, T. (2015). On the space and circuit complexity of parameterized problems: Classes and completeness. *Algorithmica* **71** (3) 661–701.
- Eppstein, D. (1990). Reset sequences for monotonic automata. *SIAM Journal on Computing* **19** (3) 500–510.
- Even, S. and Goldreich, O. (1981). The minimum-length generator sequence problem is NP-hard. *Journal of Algorithms* **2** (3) 311–313.
- Fernau, H. (2019). Modern aspects of complexity within formal languages. In: Martín-Vide, C., Okhotin, A. and Shapira, D. (eds.) *Language and Automata Theory and Applications - 13th International Conference, LATA*, LNCS, vol. 11417, Springer, 3–30.
- Fernau, H., Gusev, V. V., Hoffmann, S., Holzer, M., Volkov, M. V. and Wolf, P. (2019). Computational complexity of synchronization under regular constraints. In: Rosmanith, P., Heggeres, P. and Katoen, J.-P. (eds.) *44th International Symposium on Mathematical Foundations of Computer Science, MFCS*, Leibniz International Proceedings in Informatics (LIPIcs), vol. 138, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 63:1–63:14.
- Fernau, H., Heggeres, P. and Villanger, Y. (2015). A multi-parameter analysis of hard problems on deterministic finite automata. *Journal of Computer and System Sciences* **81** (4) 747–765.
- Fernau, H. and Hoffmann, S. (2019). Extensions to minimal synchronizing words. *Journal of Automata, Languages and Combinatorics* **24** 287–307.
- Fernau, H., Hoffmann, S. and Wehar, M. (2021). Finite automata intersection non-emptiness: Parameterized complexity revisited. CoRR, abs/2108.05244.
- Fernau, H. and Krebs, A. (2017). Problems on finite automata and the exponential time hypothesis. *Algorithms* **10** 24:1–25.
- Fernau, H. and Wolf, P. (2020). Synchronization of deterministic visibly push-down automata. In: Saxena, N. and Simon, S. (eds.) *40th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS*, LIPIcs, vol. 182, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 45:1–45:15.
- Fernau, H., Wolf, P. and Yamakami, T. (2020). Synchronizing deterministic push-down automata can be really hard. In: Esparza, J. and Král', D. (eds.) *45th International Symposium on Mathematical Foundations of Computer Science, MFCS*, LIPIcs, vol. 170, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 33:1–33:15.

- Flum, J. and Grohe, M. (2006). *Parameterized Complexity Theory*, Springer, Heidelberg, Germany.
- Frankl, P. (1982). An extremal problem for two families of sets. *European Journal of Combinatorics* **3** (2) 125–127.
- Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability*, New York, Freeman.
- Goldstine, J., Leung, H. and Wotschke, D. (1992). On the relation between ambiguity and nondeterminism in finite automata. *Information and Computation* **100** 261–270.
- Goralčík, P. and Koubek, V. (1995). Rank problems for composite transformations. *International Journal of Algebra and Computation* **5** (3) 309–316.
- Guillemot, S. (2011). Parameterized complexity and approximability of the longest compatible sequence problem. *Discrete Optimization* **8** (1) 50–60.
- Jerrum, M. (1985). The complexity of finding minimum-length generator sequences. *Theoretical Computer Science* **36** 265–289.
- Kisielewicz, A., Kowalski, J. and Szykuła, M. (2015). Computing the shortest reset words of synchronizing automata. *Journal of Combinatorial Optimization* **29** (1) 88–124.
- Kozen, D. (1977). Lower bounds for natural proof systems. In: *18th Annual Symposium on Foundations of Computer Science*, FOCS, IEEE Computer Society, 254–266.
- Martyugin, P. (2009). Complexity of problems concerning reset words for some partial cases of automata. *Acta Cybernetica* **19** (2) 517–536.
- Martyugin, P. V. (2014). Computational complexity of certain problems related to carefully synchronizing words for partial automata and directing words for nondeterministic automata. *Theory of Computing Systems* **54** (2) 293–304.
- Möhring, R. H. (1989). Computationally tractable classes of ordered sets. In: Rival, I. (ed.) *Algorithms and Order: Proceedings of the NATO Advanced Study Institute on Algorithms and Order*, NATO Science Series C, vol. 255, 105–194, Springer.
- Montoya, J. A. and Nolasco, C. (2018). On the synchronization of planar automata. In: Klein, S. T., Martín-Vide, C. and Shapira, D. (eds.) *Language and Automata Theory and Applications - 12th International Conference, LATA*, LNCS, vol. 10792, Springer, 93–104.
- Pin, J. E. (1983). On two combinatorial problems arising from automata theory. *Annals of Discrete Mathematics* **17** 535–548.
- Rystsov, I. K. (1980). On minimizing the length of synchronizing words for finite automata. In: *Theory of Designing of Computing Systems*, Institute of Cybernetics of Ukrainian Academy of Science, 75–82. (in Russian).
- Rystsov, I. K. (1983). Polynomial complete problems in automata theory. *Information Processing Letters* **16** (3) 147–151.
- Sandberg, S. (2005). Homing and synchronizing sequences. In: Broy, M., Jonsson, B., Katoen, J.-P., Leucker, M. and Pretschner, A. (eds.) *Model-Based Testing of Reactive Systems*, LNCS, vol. 3472, Springer, 5–33.
- Shitov, Y. (2019). An improvement to a recent upper bound for synchronizing words of finite automata. *Journal of Automata, Languages and Combinatorics* **24** (2–4) 367–373.
- Szykuła, M. (2018). Improving the upper bound on the length of the shortest reset word. In: Niedermeier, R. and Vallée, B. (eds.) *35th Symposium on Theoretical Aspects of Computer Science, STACS*, LIPIcs, vol. 96, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 56:1–56:13.
- Türker, U. C. and Yenigün, H. (2015). Complexities of some problems related to synchronizing, non-synchronizing and monotonic automata. *International Journal of Foundations of Computer Science* **26** (1) 99–122.
- Volkov, M. V. (2008). Synchronizing automata and the Černý conjecture. In: Martín-Vide, C., Otto, F., and Fernau, H. (eds.) *Language and Automata Theory and Applications, Second International Conference, LATA*, LNCS, vol. 5196, Springer, 11–27.
- Volkov, M. V. (2019). Preface: Special issue on the Černý conjecture. *Journal of Automata, Languages and Combinatorics* **24** (2–4) 119–121.
- Wareham, H. T. (2001). The parameterized complexity of intersection and composition operations on sets of finite-state automata. In: Yu, S. and Păun, A. (eds.) *Implementation and Application of Automata, 5th CIAA 2000*, LNCS, vol. 2088, Springer, 302–310.
- Wolf, P. (2020). Synchronization under dynamic constraints. In: Saxena, N. and Simon, S. (eds.) *40th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS*, LIPIcs, vol. 182, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 58:1–58:14.

Cite this article: Fernau H and Bruchertseifer J (2022). Synchronizing words and monoid factorization, yielding a new parameterized complexity class? *Mathematical Structures in Computer Science* **32**, 189–215. <https://doi.org/10.1017/S0960129522000184>