Synchronization Under Dynamic Constraints

Petra Wolf
Universität Trier, Fachbereich IV, Informatikwissenschaften, Germany
`https://www.wolfp.net/`
wolfp@informatik.uni-trier.de

Abstract

We introduce a new natural variant of the synchronization problem. Our aim is to model different constraints on the order in which a potential synchronizing word might traverse through the states. We discuss how a word can induce a state-order and examine the computational complexity of different variants of the problem whether an automaton can be synchronized with a word of which the induced order agrees with a given relation. While most of the problems are PSPACE-complete we also observe NP-complete variants and variants solvable in polynomial time. One of them is the careful synchronization problem for partial weakly acyclic automata (which are partial automata whose states can be ordered such that no transition leads to a smaller state), which is shown to be solvable in time $\mathcal{O}(k^2n^2)$ where n is the size of the state set and k is the alphabet-size. The algorithm even computes a synchronizing word as a witness. This is quite surprising as the careful synchronization problem uses to be a hard problem for most classes of automata. We will also observe a drop in the complexity if we track the orders of states on several paths simultaneously instead of tracking the set of active states. Further, we give upper bounds on the length of a synchronizing word depending on the size of the input relation and show that (despite the partiality) the bound of the Černý conjecture also holds for partial weakly acyclic automata.

2012 ACM Subject Classification Theory of computation → Formal languages and automata theory; Theory of computation → Problems, reductions and completeness

Keywords and phrases Synchronizing automaton, Černý conjecture, Reset sequence, Dynamic constraints, Computational complexity

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2020.58

Related Version A full version is available at `https://arxiv.org/abs/1910.01935`.

Funding *Petra Wolf*: DFG project FE 560/9-1.

Acknowledgements I thank all colleges who commented on drafts of this work, esp. Henning Fernau.

1 Introduction

We call $A = (Q, \Sigma, \delta)$ a deterministic partial (semi-) automaton (DPA) if Q is a finite set of states, Σ is a finite alphabet, and $\delta\colon Q \times \Sigma \to Q$ is a (potentially partial) transition function. If δ is defined for every element in $Q \times \Sigma$, we call A a deterministic complete (semi-) automaton (DCA). Clearly, every DCA is also a DPA. We do not specify any start and final states as we are only interested in the transition of states. A DCA $A = (Q, \Sigma, \delta)$ is *synchronizing* if there exists a word $w \in \Sigma^*$ such that w takes every state to the same state. In that case, we call w a *synchronizing word* for A. If we are only interested in synchronizing a subset of states $S \subseteq Q$ we refer to the problem as *subset synchronization*.

One of the oldest applications of the intensively studied topic of synchronizing automata is the problem of designing parts orienters, which are robots or machines that get an object in an (due to a lack of expensive sensors) unknown orientation and transform it into a defined orientation [2]. In his pioneering work, Natarajan [17] modeled the parts orienters as deterministic complete automata where a state corresponds to a possible orientation of a part and a transition of some letter a from state q corresponds to applying the modifier

© Petra Wolf;
licensed under Creative Commons License CC-BY

corresponding to a to a part in orientation q. He proved that the synchronization problem is solvable in polynomial time for – what is later called – the class of *orientable automata* [20] if the cyclic order respected by the automaton is part of the input. Many different classes of automata have since been studied regarding their synchronization behavior. We refer to [27, 4, 1] for an overview. The original motivation of designing a parts orienter was revisited in [26] where Türker and Yenigün modeled the design of an assembly line, which again brings a part from an unknown orientation into a known orientation, where different modifiers have different costs. What has not been considered so far is that different modifiers can have different impact on the parts and as we do not know the current orientation we might want to restrict the chronology of applied modifiers. For example, if the part is a box with a fold-out lid, turning it upside-down will cause the lid to open. In order to close the lid one might need another modifier such as a low bar which brushes the lid and closes it again. To specify that a parts orienter should deliver the box facing upward with a closed lid one needs to encode something like: "When the box is in the state *facing down*, it later needs to be in the state *lid closed*". But this does not stop us from opening the lid again, so we need to be more precise and encode: "After **the last time** the box was in the state *facing down*, it needs to visit the state *lid closed* at least once". We will implement these conditions in our model of a parts orienter by enhancing a given DCA with a relation R. We will then consider different ways of how a synchronizing word implies an order on the states and ask whether there exists a synchronizing word whose implied state-order agrees with the input-relation R. The case-example above will be covered by the first two introduced orders. The third considered order relates to the following scenario: Let us again picture the box with the lid in mind, but this time the box initially contains some water. We would like to have the box in a specific orientation with the lid open but the water should not be shed during orientating. We have a modifier that opens the lid and a modifier which rotates the box. Clearly we do not want the box to face downwards after the lid has been opened. So, we encode: "As soon as the state *lid open* has been reached, the state *facing downwards* should never be entered again".

For every type of dynamic constraint (which we will also call *order*), we investigate the computational complexity of the problem whether a given automaton admits a synchronizing word that transitions the states of the automaton in an order that is conform with a given relation. Thereby, we distinguish between tracking all active states simultaneously and tracking each state individually. We observe different complexities for different ordering concepts and get a good understanding of which ordering constraints yield tractable synchronization problems and which do not. The complexity of the problem also depends on how detailed we describe the allowed sequence of states.

2 Related Work

The problem of checking whether a synchronizing word exists for a DCA $A = (Q, \Sigma, \delta)$ can be solved in time $\mathcal{O}(|Q|^2|\Sigma|)$, when no synchronizing word is computed, and in time $\mathcal{O}(|Q|^3)$ when a witnessing synchronizing word is demanded [11, 27]. In comparison, if we only ask for a subset of states $S \subseteq Q$ to be synchronized, the problem becomes PSPACE-complete for general DCAs [22]. These two problems have been investigated for several smaller classes of automata involving orders on states. Here, we want to mention the class of *oriented* automata whose states can be arranged in a cyclic order which is preserved by all transitions, which have been studied among others in [17, 11, 2, 21, 27]. If the order is linear instead of cyclic, we get the class of *monotone* automata which has been studied in [2, 21]. An automaton is called *aperiodic* [4] if there is a non-negative integer k such that for any word w and any state q it holds that $\delta(q, w^k) = \delta(q, w^{k+1})$. An automaton is called *weakly acyclic* [20] if there exists an ordering of the states $q_1, q_2, \ldots, q_n$ such that if $\delta(q_i, a) = q_j$ for some letter $a \in \Sigma$, then $i \leq j$. In other words, all cycles in a WAA are self-loops. In Section 3 we will consider partial WAAs. The class of WAAs forms a proper subclass of the class of aperiodic automata. Each synchronizing aperiodic automaton admits a synchronizing word of length at most $n(n-1)/2$ [25], whereas synchronizing WAAs admit synchronizing words of linear lengths [20]. Asking whether an aperiodic automaton admits a synchronizing word of length at most k is an NP-complete task [27] as it is for general DCAs [18, 11]. The subset synchronization problem for WAAs, and hence for aperiodic automata, is NP-complete [20].

Going from complete automata to partial automata normally brings a jump in complexity. For example, the so called *careful synchronization* problem for DPAs asks for synchronizing a partial automata such that the synchronizing word w is defined on all states. The problem is PSPACE-complete for DPAs with a binary alphabet [15]. It is even PSPACE-complete for DPAs with a binary alphabet if δ is undefined for only one pair in $Q \times \Sigma$ [16]. The length of a shortest carefully synchronizing word $c(n)$, for a DPA with $|Q| = n$, differs with $\Omega(3^{\frac{n}{3}}) \leq c(n) \leq \mathcal{O}(4^{\frac{n}{3}} \cdot n^2)$ [16] significantly from the cubic upper-bound for complete automata. Also for the smaller class of monotone partial automata with an unbounded alphabet size, an exponential lower bound on the length of a shortest carefully synchronizing word is known, while for fixed alphabet sizes of 2 and 3 only a polynomial lower bound is obtained [21]. The careful synchronization problem is NP-hard for partial monotone automata over a four-letter alphabet [26, 21]. It is also NP-hard for aperiodic partial automata over a three-letter alphabet [20]. In contrast we show in Section 3 that the careful synchronization problem is decidable in polynomial time for partial WAAs.

In [20, 21] several hardness and inapproximability results are obtained for WAAs, which can be transferred into our setting as depicted in Section 3. We will also observe W[1]-hardness results from the reductions given in [20]. So far, only little is known (see for example [13, 28, 5]) about the parameterized complexity of all the different synchronization variants considered in the literature.

While synchronizing an automaton under a given order, the set of available (or allowed) transitions per state may depend on the previously visited states on all paths. This dynamic can also be observed in weighted and timed automata [10]. More static constraints given by a second automaton have been discussed in [12]. Due to space limitations missing proofs can be found in the long version of this work [29].

3 Problem Definitions

A deterministic semi-automaton $A = (Q, \Sigma, \delta)$ that might either be partial or complete is called an *automaton*. The transition function δ is generalized to words in the usual way. It is further generalized to sets of states $S \subseteq Q$ as $\delta(S, w) := \{\delta(q, w) \mid q \in S\}$. We sometimes refer to $\delta(S, w)$ as $S.w$. We call a state q *active* regarding a word w if $q \in Q.w$. If for some $w \in \Sigma^*$, $|Q.w| = 1$ we call $q \in Q.w$ a *synchronizing state*. We denote by $|S|$ the size of the set S. With $[i..j]$ we refer to the set $\{k \in \mathbb{N} \mid i \leq k \leq j\}$. For a word w over some alphabet Σ, we denote by $|w|$ the length of w, by $w[i]$ the i^{th} symbol of w (or the empty word ϵ if $i = 0$) and by $w[i..j]$ the factor of w from symbol i to symbol j. For each state q, we call the sequence of active states $q.w[i]$ for $0 \leq i \leq |w|$ the *path* induced by w starting at q. We expect the reader to be familiar with basic concepts in complexity theory, approximation theory and parameterized complexity theory [9, 24, 3].

We are now presenting different orders $\lessdot_w$ which describe how a word traverses an automaton. We describe how a word implies each of the three presented orders. The first two orders relate the last visits of the states to each other, while the third type of order relates the first visits. We will then combine the order with an automaton A and a relation $R \subseteq Q^2$ given in the input and ask whether there exists a synchronizing word for A such that the implied order of the word agrees with the relation R. An order $\lessdot_w$ agrees with a relation $R \subseteq Q^2$ if and only if for all pairs $(p, q) \in R$ it holds that $p \lessdot_w q$, i.e., $R \subseteq \lessdot_w$.

For any of the below defined orders $\lessdot_w \subseteq Q \times Q$, we define the problem of *synchronization under order* and *subset synchronization under order* as:

▶ **Definition 1** (SYNC-UNDER-$\lessdot_w$). *Given a DCA $A = (Q, \Sigma, \delta)$ and a relation $R \subseteq Q^2$. Does there exist a word $w \in \Sigma^*$ such that $|Q.w| = 1$ and $R \subseteq \lessdot_w$?*

▶ **Definition 2** (SUBSET-SYNC-UNDER-$\lessdot_w$). *Given a DCA $A = (Q, \Sigma, \delta)$, $S \subseteq Q$, and a relation $R \subseteq Q^2$. Is there a word $w \in \Sigma^*$ with $|S.w| = 1$ and $R \subseteq \lessdot_w$?*

It is reasonable to distinguish whether the order should include the initial configuration of the automaton or if it should only describe the consequences of the chosen transitions. In the former case, we refer to the problem as SYNC-UNDER-*0*-$\lessdot_w$ (starting at $w[0]$), in the latter case as SYNC-UNDER-*1*-$\lessdot_w$ (starting at $w[1]$), and if the result holds for both variants, we simply refer to is as SYNC-UNDER-$\lessdot_w$. Examples for positive and negative instances of the problem synchronization under order for some discussed variants are illustrated in Figure 1. Let $\text{first}(q, w, S)$ be the function returning the minimum of positions at which the state q appears as an active state over all paths induced by w starting at some state in S. Accordingly, let $\text{last}(q, w, S)$ return the maximum of those positions. Note that $\text{first}(q, w, S) = 0$ for all states $q \in S$ and > 0 for $q \in Q \backslash S$. If q does not appear on a path induced by w on S, then set $\text{first}(q, w, S) := |w| + 1$ and $\text{last}(q, w, S) := -1$. In the SYNC-UNDER-*1*-$\lessdot_w$ problem variant, the occurrence of a state at position 0 is ignored (i.e., if q occurs only at position 0 while reading w on S, then $\text{last}(q, w, S) = -1$). In the following definitions let $A = (Q, \Sigma, \delta)$ be a DCA and let $p, q \in Q$. The following relations $\lessdot_w$ are defined for every word $w \in \Sigma^*$.

▶ **Definition 3** (Order $l < l$ on sets). $p \propto^{l<l}_{w@s} q :\Leftrightarrow \text{last}(p, w, Q) < \text{last}(q, w, Q)$.

▶ **Definition 4** (Order $l \leq l$ on sets). $p \propto^{l\leq l}_{w@s} q :\Leftrightarrow \text{last}(p, w, Q) \leq \text{last}(q, w, Q)$.

The second order differs from the first in the sense that q does not have to appear finally without p, instead they can disappear simultaneously. Further, note that in comparison with order $\propto^{l<l}_{w@s}$, for a pair (p, q) in order $\propto^{l\leq l}_{w@s}$ it is not demanded that q is active after reading w up to some position $i > 0$. This will make a difference when we later consider the orders on isolated paths rather than on the transition of the whole state set. It can easily be verified that for any word $w \in \Sigma^*$ and any automaton $A = (Q, \Sigma, \delta)$ the order $\propto^{l<l}_{w@s}$ is a proper subset of $\propto^{l\leq l}_{w@s}$. For the order $\propto^{l\leq l}_{w@s}$, it makes no difference whether we take the initial configuration into account since states can disappear simultaneously.

So far, we only introduced orders which consider the set of active states as a whole. It did not matter which active state belongs to which path and a state on a path τ could stand in a relation with a state on some other path ρ. But, in most scenarios the fact that we start with the active state set Q only models the lack of knowledge about the *actual* current state. In practice only one state q is active and hence any constraints on the ordering of transitioned states should apply to the path starting at q. Therefore, we are introducing variants of order 1 and 2 which are defined on paths rather than on series of state sets.

▶ **Definition 5** (Order $l < l$ on paths). $p \propto^{l<l}_{w@p} q :\Leftrightarrow \forall r \in Q\colon\ \text{last}(p, w, \{r\}) < \text{last}(q, w, \{r\})$.

▶ **Definition 6** (Order $l \leq l$ on paths). $p \propto_{w@p}^{l\leq l} q :\Leftrightarrow \forall r \in Q\colon \operatorname{last}(p, w, \{r\}) \leq \operatorname{last}(q, w, \{r\})$.

The orders $\propto_{w@p}^{l<l}$ and $\propto_{w@p}^{l\leq l}$ significantly differ since the synchronization problem (starting at position 1) for $\propto_{w@p}^{l<l}$ is in NP while it is PSPACE-complete for $\propto_{w@p}^{l\leq l}$.

While the previously defined orders are bringing "positive" constraints to the future transitions of a word, in the sense that the visit of a state p will demand for a later visit of the state q (as opening the lid demands closing the lid later in our introductory example), we will now introduce an order which yields "negative" constraints. The third kind of order demands for a pair of states (p, q) that the (first) visit of the state q forbids any future visits of the state p (like do not turn the box after opening the lid). This stands in contrast to the previous orders where we could made up for a "forbidden" visit of the state p by visiting q again. The order $l < f$ will only be considered on paths since when we consider the state set Q, every pair in R would already be violated in position 0.

▶ **Definition 7** (Order $l < f$ on paths). $p \propto_{w@p}^{l<f} q :\Leftrightarrow \forall r \in Q : \operatorname{last}(p, w, \{r\}) < \operatorname{first}(q, w, \{r\})$.

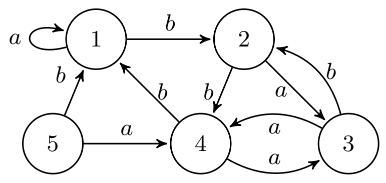
Note that $\propto_{w@p}^{l<f}$ is not transitive; e.g., for $R = \{(1, 2), (2, 3)\}$ we are allowed to go from 3 to 1 as long as we have not transitioned from 1 to 2 yet. For the order $l < f$, we will also consider the special case of R being a strict total order (irreflexive, asymmetric, transitive, and total).

▶ **Definition 8** (SYNC-UNDER-TOTAL-$\propto_{w@p}^{l<f}$). *Given a DCA $A = (Q, \Sigma, \delta)$, a strict and total order $R \subseteq Q^2$. Is there a word $w \in \Sigma^*$ with $|Q.w| = 1$ and $R \subseteq \propto_{w@p}^{l<f}$?*

The orders on path could also be stated as LTL formulas of some kind which need to be satisfied on every path induced by a synchronizing word w and our hardness results transfer to the more general problem whether a given DCA can be synchronized by a word such that every path induced by w satisfies a given LTL formula. The orders on sets could be translated into LTL formulas which need to be satisfied on the path in the power-set-automaton starting in the state representing Q. Despite the similarity of the chosen orders and their translated LTL formulas we need different constructions for the considered orders as the presented attempts mostly do not transfer to the other problems. Therefore, it is not to be expected that a general construction for restricted LTL formulas can be obtained. Our aim is to focus on restricting the order in which states appear and disappear on a path in the automaton or on a path in the power-set-automaton (remember the introductory example). Hence, we have chosen the stated definitions in order to investigate the complexity of problems where the LTL formula is always of the same type, i.e., comparing only the last or first appearances of states on a path. We leave it to future research to investigate other types of LTL formulas. In order to express synchronizability of Kripke structures, an extension to CTL has been introduced in [8]. Note that synchronization of Kripke structures is more similar to D3-directing words [14] for unary NFAs as in contrast to general DFAs the labels of the transitions are omitted in Kripke structures.

▶ **Definition 9** (CAREFUL SYNC (PSPACE-complete [15])). *Given a DPA $A = (Q, \Sigma, \delta)$. Is there a word $w \in \Sigma^*$, s.t. $|Q.w| = 1$ and w is defined on all $q \in Q$?*

▶ **Definition 10** (VERTEX COVER (NP-complete [24])). *Given a graph $G = (V, E)$ and an integer $k \leq |V|$. Is there a vertex cover $V' \subseteq V$ of size $|V'| \leq k$? A vertex cover is a set of states that contains at least one vertex incident to every edge.*



	1	2	3	4	5
b	2	4	2	1	1
a	3	3	3	1	1
a	4	4	4	1	1
b	1	1	1	2	2
b	2	2	2	4	4
a	3	3	3	3	3

	✓	✗
$\propto^{l<l}_{w@s}$	$(1,2)$	$(2,4)$
$\propto^{l\leq l}_{w@s}$	$(2,4)$	$(2,1)$
$\propto^{l<l}_{w@p}$	$(1,2)$	$(5,5)$
$\propto^{l\leq l}_{w@p}$	$(5,5)$	$(2,4)$
$\propto^{l<f}_{w@p}$	$(5,2)$	$(4,3)$

Figure 1 DCA A (left) with all paths induced by $w = baabba$ (middle) and relations R consisting of single pairs forming a positive, resp. negative, instance for versions of Sync-Under-$\lessdot_w$ (right).

Table 1 Overview of the complexity for synchronization (on the left), and subset synchronization under order (on the right) for relations $\propto^{l<l}_{w@s}$, $\propto^{l<l}_{w@p}$, $\propto^{l\leq l}_{w@s}$, $\propto^{l\leq l}_{w@p}$, and $\propto^{l<f}_{w@p}$ (tot. is short for total).

		Synchronization				*Subset Synchronization*		
Order		$l<l$	$l\leq l$	$l<f$	$l{<}f$-tot	$l</\leq l$	$l<f$	$l{<}f$-tot
Set	*0*	PSPACE-c	PSPACE-c	–	–	PSPACE-c	–	–
	1	PSPACE-c	PSPACE-c	–	–	PSPACE-c	–	–
Path	*0*	in NP	NP-hard	PSPACE-c	P	PSPACE-c	PSPACE-c	NP-c
	1	in NP	PSPACE-c	PSPACE-c	NP-c	PSPACE-c	PSPACE-c	NP-c

4 Main Results

We now investigate the complexity of the introduced problems. An overview on the obtained results is given in Table 1.

▶ **Theorem 11.** *For all orders* $\lessdot \in \{\propto^{l<l}_{w@s}, \propto^{l<l}_{w@p}, \propto^{l\leq l}_{w@s}, \propto^{l\leq l}_{w@p}, \propto^{l<f}_{w@p}\}$*, the problem* Sync-Under-$\lessdot$ *is contained in PSPACE. Further, it is FPT with parameter* $p = |Q|$.

Proof sketch. For a DCA $A = (Q, \Sigma, \delta)$ and $R \subseteq Q^2$, we can enrich the powerset-automaton $\mathcal{P}(A)$ of A with the information about the set of active pairs in R in every state. Here, a pair in R is active during the transition of a word if it constrains which states might be, or need to be visited in the future. For instance, in the example in Figure 1 concerning the order $\propto^{l<l}_{w@s}$, the pair $(2,4)$ is active while reading the prefix ba, since the state 2 has appeared as an active state while the state 4 has not appeared without 2 as an active state yet. It is not active after reading baa, since now 4 is active without 2 and hence the pair $(2,4)$ is satisfied and does not demand for further state visits. The pair becomes active again after reading $baab$ since again 2 became active demanding for the state 4 to become active without 2 again.

To store the information of active pairs in R, we copy each state in $\mathcal{P}(A)$ $2^{|R|}$ times for the orders $\propto^{l<l}_{w@s}$, and $\propto^{l\leq l}_{w@s}$, and $|Q|2^{|R|}$ times for the orders $\propto^{l<l}_{w@p}, \propto^{l\leq l}_{w@p}$, and $\propto^{l<f}_{w@p}$, yielding an automaton $\mathcal{P}(A)$ of size at most $2^{|Q|}|Q|2^{|R|} = 2^{\mathcal{O}(|Q|^2)}$. As each state contains only up to $|Q|+1$ bit-strings of length up to $|Q|^2$ a state of $\mathcal{P}(A)$ can be stored in polynomial space. Hence, reachability tests can be performed in $\mathcal{P}(A)$ in polynomial space and in time $2^{\mathcal{O}(|Q|^2)}$. ◀

▶ **Theorem 12.** Sync-Under-$\propto^{l\leq l}_{w@s}$ *is PSPACE-complete, even for* $|R| = 1$ *and* $|\Sigma| = 2$.

Proof. We reduce from the PSPACE-complete Careful Sync problem for DPAs [15]. Let $A = (Q, \Sigma, \delta)$ be a DPA. We construct from A a DCA $A' = (Q' = Q \cup \{q_\ominus, r\}, \Sigma, \delta')$ with $q_\ominus, r \notin Q$. For every pair $q \in Q, \sigma \in \Sigma$ for which $\delta(q, \sigma)$ is undefined, we define the transition $\delta'(q, \sigma) = q_\ominus$. On all other pairs δ' agrees with δ. Further, for some arbitrary state $t \in Q$ and for all $\gamma \in \Sigma$ we set $\delta'(q_\ominus, \gamma) = \delta'(t, \gamma)$ (note that this can be $q_\ominus$ itself) and $\delta'(r, \gamma) = \delta'(t, \gamma)$. We set the relation R to $R := \{(q_\ominus, r)\}$.

Assume there exists a word $w \in \Sigma^*$, $|w| = n$ that synchronizes A without using an undefined transition. Then, $\delta(q, w[1])$ is defined for all states $q \in Q$. The letter $w[1]$ acts on A' in the following way: (1) $\delta'(r, w[1]) = \delta'(q_\ominus, w[1]) = \delta(t, w[1])$ which is defined by assumption; (2) $\delta'(Q, w[1]) \subseteq Q$ since $\delta(q, w[1])$ is defined for all states $q \in Q$. The combination of (1)-(2) yields $\delta'(Q', w[1]) \subseteq Q$. We further constructed δ' such that $\delta'(Q', w[1]) = \delta(Q, w[1])$. Since $\delta(q, w[2..n])$ is defined by assumption for every $q \in \delta(Q, w[1])$, δ' agrees with δ on $w[2..n]$ for every $q \in \delta(Q, w[1])$. This means especially that while reading $w[2..n]$ in A' on the states in $\delta'(Q', w[1])$ the state $q_\ominus$ is not reached and that $\delta'(Q', w) = \delta(Q, w)$. Therefore, w also synchronizes the automaton A'. The state $q_\ominus$ is only active in the start configuration where no letter of w is read yet and is not active anymore while reading w. The same holds for r, hence $R = \{(q_\ominus, r)\} \subseteq \propto^{l \leq l}_{w@s}$.

For the other direction, assume there exists a word $w \in \Sigma^*$, $|w| = n$ that synchronizes A' with $(q_\ominus, r) \in \propto^{l \leq l}_{w@s}$. Then, w can be partitioned into $w = uv$ with $u, v \in \Sigma^*$ where r is not active while reading the factor v in w. The only position of w in which r is active due to the definition of δ' is before any letter of w is read. Hence, we can set $u = \epsilon$ and $v = w$. As $(q_\ominus, r) \in \propto^{l \leq l}_{w@s}$ it holds for all $i \in [1..n]$ that $q_\ominus \notin \delta'(Q', v[1..i])$. Hence, $\delta'(q, v)$ is defined for every state $q \in Q$. Since δ' and δ agree on the definition range of δ it follows that v also synchronizes the state set Q in A without using an undefined transition. ◀

▶ Remark 13. The construction works for both variants (with and without 0) of the problem. It can further be adapted for the order $\propto^{l<l}_{w@s}$ (both variants) by introducing a copy $\hat{q}$ of every state in $Q \cup \{r\}$ and setting $\delta'(\hat{q}, \sigma) = q$ for every $\sigma \in \Sigma$, $q \in Q \cup \{r\}$. For all other transitions, we follow the above construction. We keep $R := \{(q_\ominus, r)\}$. Since r is left after $w[2]$ for any word $w \in \Sigma^*$ with $|w| \geq 2$ in order to satisfy R the state $q_\ominus$ needs to be left with $w[1]$ such that afterwards r is active without $q_\ominus$. Note that $q_\ominus$ has not been copied.

▶ **Corollary 14.** *SYNC-UNDER-$\propto^{l<l}_{w@s}$ is PSPACE-complete even for $|R| = 1$ and $|\Sigma| = 2$.*

▶ Remark 15. The reduction presented in the proof of Theorem 12 can also be applied to show the PSPACE-completeness of SYNC-UNDER-*1*-$\propto^{l \leq l}_{w@p}$. Since the state r cannot be reached from any other state, the state $q_\ominus$ needs to be left with the first letter of any synchronizing word and must not become active again on any path. The rest of the argument follows the proof of Theorem 12. Note that the construction only works for SYNC-UNDER-*1*-$\propto^{l \leq l}_{w@p}$. If we consider SYNC-UNDER-*0*-$\propto^{l \leq l}_{w@p}$ the problem might become easier. But it is at least NP-hard.

▶ **Theorem 16.** *The problem SYNC-UNDER-0-$\propto^{l \leq l}_{w@p}$ is NP-hard.*

Proof. We give a reduction from VERTEX COVER. We refer to Figure 2 for a schematic illustration. Let $G(V, E)$ be a graph and let $k \in \mathbb{N}$. We construct from G a DCA $A = (Q, \Sigma, \delta)$ in the following way. We set $\Sigma = V \cup \{p\}$ for some $p \notin V$. We start with $Q = \{f, r, s\}$ where s is a sink state, meaning $\delta(s, \sigma) = s$ for all $\sigma \in \Sigma$, f will be the "false way" and r will be the "right way". We set $\delta(r, p) = \delta(f, p) = s$ and $\delta(r, v) = r$, $\delta(f, v) = f$ for all other $v \in \Sigma$. For every edge $e_{ij} \in E$ connecting some vertices $v_i, v_j \in V$, we create two states e_{ij} and $\hat{e}_{ij}$ and set $\delta(e_{ij}, v_i) = \delta(e_{ij}, v_j) = \hat{e}_{ij}$, $\delta(e_{ij}, p) = f$. For all other letters, we stay in e_{ij}. For the state $\hat{e}_{ij}$, we stay in $\hat{e}_{ij}$ for all letters except p. For p, we set $\delta(\hat{e}_{ij}, p) = s$. We further create for $1 \leq i \leq k+2$ the states q_i with the transitions $\delta(q_i, v) = q_{i+1}$ for $i \leq k+1$ and $v \in V$, $\delta(q_i, p) = q_i$ for $i \leq k$, and $\delta(q_{k+1}, p) = r$, $\delta(q_{k+2}, p) = s$, $\delta(q_{k+2}, v) = q_{k+2}$ for $v \in V$. We set $R := (q_1, r) \cup \{(e_{ij}, \hat{e}_{ij}) \mid e_{ij} \in E\}$.

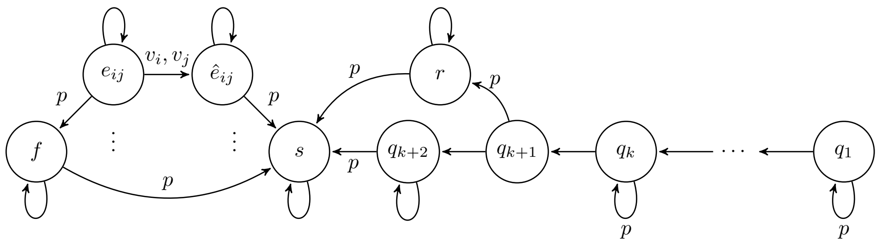


Figure 2 Schematic illustration of the reduction from Vertex cover (see Theorem 16). For each state, the transition without a label represents all letters which are not explicitly listed as an outgoing transition from that state.

If there exists a vertex cover of size $k' < k$ for G, then there also exists a vertex cover of size k for G. Therefore, assume V' is a vertex cover for G of size k. Then, the word wpp where w is any non-repeating listing of the vertices in V' is a synchronizing word for A with $R \subseteq \propto^{l \le l}_{wpp@p}$. Since q_1 cannot be reached from any other state, the pair $(q_1, r) \in R$ is trivially satisfied for each path starting in any state other than q_1. Hence, we only have to track the appearances of q_1 and r on the path starting in q_1. Since w lists the states in the vertex cover V' it holds that $|w| = k$ and hence $q_1.w = q_{k+1}$. Further, $q_1.wp = r$ and $q_1.wpp = s$. Hence, the pair (q_1, r) is satisfied on the path starting in q_1 as well as on all paths. It remains to show that wpp is indeed a synchronizing word and that all pairs in R of the form $(e_{ij}, \hat{e}_{ij})$ are satisfied. For every state e_{ij} representing an edge e_{ij}, the state $\hat{e}_{ij}$ is reached if we read a letter corresponding to a vertex incident to it. Since V' is a vertex cover, the word w contains for each edge e_{ij} at least one vertex incident to it. Hence, for each edge $e_{ij}.w = \hat{e}_{ij}$ and $e_{ij}.wpp = s$. Since each state e_{ij} is not reachable from any other state it follows that all pairs $(e_{ij}, \hat{e}_{ij})$ are satisfied by wpp on all paths. It is easy to see that for all other states $q \in Q$ it holds that $q.wpp = s$.

For the other direction, assume there exists a synchronizing word w for A with $R \subseteq \propto^{l \le l}_{w@p}$. By the construction of A the word w must contain some letters p. Partition w into $w = upv$ where p does not appear in u. Since $R \subseteq \propto^{l \le l}_{w@p}$ the pair (q_1, r) in R enforces $|u| \le k$ since otherwise the only path on which q_1 appears (namely the one starting in q_1) will not contain the state r as for any longer prefix u it holds that $q_1.u = q_{k+2}$ and r is not reachable from q_{k+2}. The other pairs of the form $(e_{ij}, \hat{e}_{ij}) \in R$ enforces that u encodes a vertex cover for G. Assume this is not the case, then there is some state e_{ij} for which $e_{ij}.u = e_{ij}$. But then, $e_{ij}.up = f$ and from f the state $\hat{e}_{ij}$ is not reachable, hence the pair $(e_{ij}, \hat{e}_{ij})$ is not satisfied on the path starting in e_{ij}. Therefore, u encodes a vertex cover of size at most k. ◀

If we consider $\propto^{l<l}_{w@p}$, the two variants of the order (with and without position $i = 0$) do not differ since for a pair (p, q), regardless of whether p is reached, the state q must be reached on every path. Hence, whenever we leave q we must be able to return to it, so it does not matter if we consider starting in q or not. In comparison with Sync-Under-*1*-$\propto^{l \le l}_{w@p}$, the problem Sync-Under-$\propto^{l<l}_{w@p}$ is solvable in polynomial time using non-determinism.

▶ **Theorem 17.** *The problem* Sync-Under-$\propto^{l<l}_{w@p}$ *is in NP.*

Proof. Recall that in the problem Sync-Under-$\propto^{l<l}_{w@p}$, for every pair of states $(p, q) \in R$ and every state $r \in Q$, it is demanded that q appears somewhere on a path induced by the sought synchronizing word w, starting in r. Hence, a precondition for the existence of w is that for every pair $(p_i, q_i) \in R$ the states q_i must be reachable *from any state* in Q. More

precisely, under the order $\propto^{l<l}_{w@p}$ only the last appearance of each state on a path is taken into account. Hence, a prohibited visit of a state can later be compensated by revisiting all related states in the correct order. Thus, it is sufficient to first synchronize all pairs of states and then transition the remaining state through all related states in the demanded order. ◀

▶ Remark 18. The NP-hardness proof for SYNC-UNDER-*0*-$\propto^{l\leq l}_{w@p}$ in Theorem 16 and the NP-membership proof for SYNC-UNDER-$\propto^{l<l}_{w@p}$ in Theorem 17 do not work for the respectively other problem since concerning $\propto^{l<l}_{w@p}$ the larger states need to be reached on *every* path and not only on a path containing the corresponding smaller state as it is the case concerning $\propto^{l\leq l}_{w@p}$.

▶ **Theorem 19.** *The problem* SYNC-UNDER-*0*-$\propto^{l<f}_{w@p}$ *is PSPACE-complete.*

Proof sketch. We reduce from CAREFUL SYNC. As in the proof of Theorem 12 we take every undefined transition $\delta(q, \sigma)$ to the new state $q_\ominus$. We further enrich the alphabet by a letter c and use c to take $q_\ominus$ into Q. We use the relation R and extra states r, s to enforce that c is the first letter of any synchronizing word, and that afterwards $q_\ominus$ is not reached again. ◀

▶ Remark 20. In the presented way, the reduction relies on taking the initial configuration at position $i = 0$ into account but we can adapt the construction to prove PSPACE-completeness of SYNC-UNDER-*1*-$\propto^{l<f}_{w@p}$ by copying every state in Q and the state r. Denote a copy of a state q with q'. Then, we set $\delta'(q', \sigma) = q$ for any copied state including r'. Note that the copied states are not reachable from any state. Now, after the first transition $w[1]$ (which can be arbitrary), we have a similar situation as previously considered for $w[0]$. The state r is active and forces the next letter to be the letter c; all states in Q are active; reading the letter c will cause all states q_σ to be left and never be reached again.

In the above reduction from CAREFUL SYNC the size of R depends on $|Q|$. Hence, the question whether SYNC-UNDER-$\propto^{l<f}_{w@p}$ is PSPACE-hard for $|R| = 1$ is an interesting topic for further research. We will now see that when R is a strict and total order on Q, the problem of synchronizing under $\propto^{l<f}_{w@p}$ (a.k.a. SYNC-UNDER-TOTAL-$\propto^{l<f}_{w@p}$) becomes tractable.

▶ **Theorem 21.** *Let $A = (Q, \Sigma, \delta)$, R be an instance of* SYNC-UNDER-TOTAL-$\propto^{l<f}_{w@p}$*. A shortest synchronizing word w for A with $R \subseteq \propto^{l<f}_{w@p}$ has length $|w| \leq \frac{|Q|(|Q|-1)}{2} + 1$.*

Note that this length bound is smaller than the bound of the Černý conjecture [6, 7] for $|Q| > 3$. The same bound can be obtained for SUBSET-SYNC-UNDER-TOTAL-$\propto^{l<f}_{w@p}$. We will now prove that the problem SYNC-UNDER-TOTAL-*0*-$\propto^{l<f}_{w@p}$ is equivalent – concerning polynomial time many-one-reductions (depicted by $\equiv_p$) – to the problem of carefully synchronizing a partial weakly acyclic automaton (PWAA) (a PWAA is a WAA where δ might be only partially defined). The obtained length bound also holds for PWAAs, which is only a quadratic increase w.r.t. the linear length bound in the complete case [20]. This is quite surprising as in general shortest carefully synchronizing words have an exponential lower bound [16]. Further, we show that careful synchronization for PWAAs is in P while the problem is PSPACE-complete for general DPAs even if only one transition is undefined [16].

▶ **Theorem 22.** SYNC-UNDER-TOTAL-*0*-$\propto^{l<f}_{w@p}$ $\equiv_p$ CAREFUL SYNC *of PWAAs.*

Proof. We prove this statement by reducing the two problems to each other. Let $A = (Q, \Sigma, \delta)$, $R \subseteq Q^2$ be an instance of SYNC-UNDER-TOTAL-*0*-$\propto^{l<f}_{w@p}$. Since R is a strict total order on Q, we can order the states according to R. We construct from A the PWAA $A' = (Q, \Sigma, \delta')$ by removing all transitions in δ which are leading backwards in the order. Clearly, A' is carefully synchronizable if and only if A is synchronizable with respect to R.

For the other reduction, assume $A = (Q, \Sigma, \delta)$ is a PWAA. Then, we can order the states in Q such that no transition leads to a smaller state. We are constructing from A the DCA $A' = (Q \cup \{q_<\}, \Sigma, \delta')$ and insert $q_<$ as the smallest state in the state ordering. Then, we define in δ' all transitions (q, σ) for $q \in Q, \sigma \in \Sigma$ which are undefined in δ as $\delta'(q, \sigma) = q_<$. We take the state $q_<$ with every symbol to the maximal state in the order. Note that the maximal state needs to be the synchronizing state if one exists. We set $R = \{(p, q) \mid p < q \text{ in the state ordering of } Q \text{ in } A\} \cup \{(q_<, q) \mid q \in Q\}$. Every undefined transition (p, σ) in A is not allowed in A' at any time, since otherwise the pair $(q_<, p) \in R$ would be violated. The state $q_<$ itself can reach the synchronizing state with any transition. Hence, A' is synchronizable with respect to R if and only if A is carefully synchronizable. ◀

▶ **Theorem 23.** *Let $A = (Q, \Sigma, \delta)$, $R \subseteq Q^2$ with $n := |Q|$. Let $Q_1 \subseteq Q$ be such that R restricted to $Q_1 \times Q_1$ is a strict and total order. Let $p = |Q| - |Q_1|$. For* Sync-Under-$\propto^{l<f}_{w@p}$*: If A is synchronizable by a shortest word w with $R \subseteq \propto^{l<f}_{w@p}$, then: $|w| \leq (\frac{n(n-1)}{2} + 1) \cdot 2^p$.*

We now present an $\mathcal{O}(|\Sigma|^2|Q|^2)$ algorithm for Sync-Under-Total-*0*-$\propto^{l<f}_{w@p}$. The idea is the following: We start on all states as the set of active states and pick a letter, which is defined on all active states and maps at least one active state to a larger state in the order R. We collect the sequence u of applied letters and after each step, we apply the whole sequence u on the set of active states. This is possible as we already know that u is defined on Q. We thereby ensure that a state which has become inactive after some iteration never becomes active again after an iteration step and hence Σ_{def} grows in each step and never shrinks. While a greedy algorithm which does not store u runs in $\mathcal{O}(|\Sigma||Q|^3)$, with this trick we get a running time of $\mathcal{O}(|\Sigma|^2|Q|^2)$. As in practice $|Q| \gg |\Sigma|$ this is a remarkable improvement. Note that we can store u compactly by only keeping the map induced by the current u and storing the sequence of letters σ from which we can restore the value of u in each iteration.

▶ **Theorem 24.** Sync-Under-Total-*0*-$\propto^{l<f}_{w@p}$ *is solvable in quadratic time.*

Proof. Let $A = (Q, \Sigma, \delta)$ be a DCA, and let $R \subseteq Q^2$ be a strict and total order on Q. Figure 3 describes an algorithm that decides in time $\mathcal{O}(|\Sigma|^2|Q|^2)$ whether A is synchronizable with respect to R under the order $\propto^{l<f}_{w@p}$ (including position $i = 0$) on paths. Despite the simplicity of the algorithm its correctness is not trivial and is proven in the following lemmas. ◀

▶ **Lemma 25.** *The algorithm in Figure 3 terminates on every input $A = (Q, \Sigma, \delta)$ with $m := |\Sigma|$, $n := Q$, strict and total order $R \subseteq |Q|^2$ in time $\mathcal{O}(m^2n^2)$.*

Proof. Step 1 can be performed in time $\mathcal{O}(n \log n)$ using the Quicksort-algorithm. Step 2 to Step 5 take time $\mathcal{O}(mn)$ each. The procedure `explore` takes time $\mathcal{O}(mn^2)$. The number of iterations in Step 6 is bounded by $|\Sigma_{\text{part}}|$ as Σ_{def} is applied exhaustively on Q_{act} and by invariant (2) of Lemma 26 we have $Q'_{\text{act}} \subseteq Q_{\text{act}}$, This yields a total run-time of $\mathcal{O}(m^2n^2)$. ◀

▶ **Lemma 26.** *If the algorithm in Figure 3 returns true on the input $A = (Q, \Sigma, \delta)$, strict and total order $R \subseteq |Q|^2$, then A, R is a yes instance of* Sync-Under-Total-*0*-$\propto^{l<f}_{w@p}$.

Proof. For the procedure `explore`, the following invariant holds: Let u_{old} be the word u before the execution of `explore` and let u_{new} be the one after the execution of `explore`. Then, it holds for all executions of `explore` that (1) $Q.u_{\text{new}}$ is defined, (2) $Q.u_{\text{new}} \subseteq Q.u_{\text{old}}$, and (3) $Q.u_{\text{new}}u_{\text{new}} = Q.u_{\text{new}}$. We prove the invariant by induction. First, note that the word u computed by `explore` in Step 5 is defined on all states in Q since it only consists of letters which are defined on all states. Since we go through the states in order during the

Step 1: Order all states in Q according to the order R. Since R is strict and total the states can be ordered in an array $\{q_1, q_2, \dots, q_n\}$.
Step 2: Delete in the automaton A all transitions which are leading backwards in the state-ordering. If this produces a state with no outgoing arc, abort; return false.
Step 3: Let q_n be the maximal state according to the order R. Delete all transitions in A which are labeled with letters $\sigma \in \Sigma$ for which $q_n.\sigma$ is undefined. If this produces a state with no outgoing transition, abort and return false.
Step 4: Partition the alphabet Σ into Σ_{def}, consisting of all letters $\sigma \in \Sigma$ for which $q.\sigma$ is defined for all states $q \in Q$, and $\Sigma_{\text{par}} := \Sigma \backslash \Sigma_{\text{def}}$. If $\Sigma_{\text{def}} = \emptyset$ abort; return false.
Step 5: Compute $\texttt{explore}(Q, Q, \Sigma_{\text{def}}, \epsilon)$ which returns Q_{act} and $u \in \Sigma^*_{\text{def}}$. The returned set of active states will equate $Q_{\text{trap}} = \{q \in Q \mid q.\Sigma_{\text{def}} = q\}$.
Step 6: Set $\Sigma_{\text{def}} := \Sigma_{\text{def}} \cup \{\sigma \in \Sigma_{\text{par}} \mid q.\sigma$ is defined for all $q \in Q_{\text{act}}\}$.
Compute $\texttt{explore}(Q, Q_{\text{act}}, \Sigma_{\text{def}}, u)$ which returns Q'_{act} and $u' \in \Sigma^*_{\text{def}}$.
Set $Q_{\text{act}} := Q'_{\text{act}}$, $u := u'$, $\Sigma_{\text{par}} := \Sigma \backslash \Sigma_{\text{def}}$.
Repeat this step until Q_{act} does not change anymore ($\equiv$ to Σ_{def} does not change anymore). Then, if $Q_{\text{act}} = \{q_n\}$ return true, otherwise return false.
Procedure `explore`: Input: Ordered state set Q, set of active states Q_{act}, alphabet Σ_{exp} to be explored, word u with $Q.u = Q_{\text{act}}$.
Initialize a new word $u' := u$.
Go through the active states in order. For the current state q, test if any $\sigma \in \Sigma_{\text{def}}$ leads to a larger state, if so, perform the transition σu on all active states and update the set of active states Q_{act}. Concatenate u' with σu. Continue with the next larger active state (note that this can be $q.\sigma u$). If q_n is reached, return u', and the current set of active states Q_{act}.

Figure 3 Poly-time algorithm for SYNC-UNDER-TOTAL-*0*-$\propto^{l<f}_{w@p}$ on $A = (Q, \Sigma, \delta)$, $R \subseteq Q^2$.

execution of `explore` and we only proceed with the next larger state if (1) we where able to leave the current one towards a larger state or if (2) the current state cannot be left with any of the letters in Σ_{def}, it holds that $Q.uu = Q.u$. Also, trivially $Q.u \subseteq Q$.

Next, consider some later execution of `explore`. The new word computed by `explore` is of the form $u_{\text{new}} := u_{\text{old}}\sigma_1 u_{\text{old}}\sigma_2 u_{\text{old}} \dots \sigma_i u_{\text{old}}$ for some $0 \leq i \leq |Q|$. The induction hypothesis tells us that (1) $Q.u_{\text{old}}$ is defined. Since $Q.u_{\text{old}}\sigma_1$ is defined (since $\sigma_1 \in \Sigma_{\text{def}}$) and $Q.u_{\text{old}}\sigma_1 \subseteq Q$ it holds that $Q.u_{\text{old}}\sigma_1 u_{\text{old}}$ is defined. Further, since u_{old} brings all states to the set $Q.u_{\text{old}}$ it also brings a subset of Q to a subset of $Q.u_{\text{old}}$. Using the induction hypothesis (3) we get by an induction on i that $Q.u_{\text{new}}$ is defined and $Q.u_{\text{new}} \subseteq Q.u_{\text{old}}$. Since in the execution of `explore` we only proceed with the next larger state if we exhaustively checked all possible transitions for the current state and since $Q.u_{\text{old}}u_{\text{old}} = Q.u_{\text{old}}$ it follows that $Q.u_{\text{new}}u_{\text{new}} = Q.u_{\text{new}}$.

If the algorithm in the proof of Theorem 24 terminates and returns yes, it also returns a synchronizing word u. By the invariant proven above, we know that $Q.u$ is defined. This means that u never causes a transition of a larger state to a smaller state and hence $\propto^{l<f}_{u@p}$ agrees with R. During the execution of the algorithm we track the set of active states Q_{act} (starting with Q) and only return true if Q_{act} contains only the in R largest state q_n. Since R is a total order, every $q \in Q$ is smaller than q_n and hence q_n cannot be left. Therefore, q_n needs to be the single synchronizing state of A and u is a synchronizing word for A. ◀

▶ **Lemma 27.** *If the algorithm in Figure 3 returns false on input $A = (Q, \Sigma, \delta)$, strict and total order $R \subseteq |Q|^2$, then A is not synchronizable with respect to R under the order $\propto^{l<f}_{w@p}$.*

Proof. The algorithm returns false in the following cases.

(1) All outgoing transitions of some state q are deleted in Step 2. In that case, every transition of q leads to a smaller state. As this would violate the order R, we cannot perform any of those transitions. Hence, q cannot be left. (The case that $q = q_n$ is treated in (2).)

(2) Since q_n is the largest state, it cannot be left. Hence, q_n will be active the whole time. Therefore, any transition which is not defined for q_n cannot be taken at all since q_n is active during the whole synchronizing process. Hence, we can delete these transitions globally. If this creates a state which cannot be left anymore, this state cannot be synchronized.

(3) The execution of `explore` returns two identical sets of active states Q_{act} in a row. Let Σ_{def} be the explored alphabet of the last execution of `explore`. Then, Σ_{def} contains all letters σ from Σ for which $q.\sigma$ is defined on all states $q \in Q_{\text{act}}$ and none of them leads some state in Q_{act} to a larger state. Since the relation R forbids cycles, for all $\sigma \in \Sigma_{\text{def}}$ and all $q \in Q_{\text{act}}$ $q.\sigma = q$ and hence this set cannot be left when all states of the set are active simultaneously. Since all states are active at the beginning of the algorithm, also all states in Q_{act} are active and since this set cannot be left with any transition which does not cause an undefined transition for all states in the set, the state set cannot be synchronized at all. ◀

▶ **Corollary 28.** *The careful synchronization problem for PWAA is in P.*

If we allow one unrestricted transition first (Sync-Under-Total-*1*-$\propto^{l<f}_{w@p}$) the problem is related to the subset synchronization problem of complete WAAs which is NP-complete [20]. Together with the quadratic length bound of a synchronizing word of Sync-Under-Total-*1*-$\propto^{l<f}_{w@p}$ (which implies membership of Sync-Under-Total-*1*-$\propto^{l<f}_{w@p}$ in NP), we get:

▶ **Theorem 29.** *The problem* Sync-Under-Total-*1*-$\propto^{l<f}_{w@p}$ *is NP-complete.*

Proof sketch. We reduce from the NP-complete problem: Given a complete weakly acyclic automaton $A = (Q, \Sigma, \delta)$ and a subset $S \subseteq Q$, does there exist word $w \in \Sigma^*$ such that $|S.w| = 1$? We construct from A an automaton $A' = (Q', \Sigma \cup \{c\}, \delta')$ with $c \notin \Sigma$ in the following way. We start with $Q' = Q$. W.l.o.g., assume $|S| \geq 2$. For each state $q \in S$, we add a copy $\hat{q}$ to Q'. Further, we add the states $q_<$ and $q_>$. Let $q_1, q_2, \ldots, q_n$ be an ordering of the states in Q such that δ follows this ordering. The transition function δ' agrees with δ on all states in Q and letters in Σ. For a copied state $\hat{q}$, we set $\delta'(\hat{q}, \sigma) = \hat{q}$ for all $\sigma \in \Sigma$ and $\delta'(\hat{q}, c) = q$. For every state $q \in Q$, we set $\delta'(q, c) = q_<$. Let q_s be some state in S. Then for all $\sigma \in \Sigma$ we set $\delta'(q_<, \sigma) = \delta(q_s, \sigma)$, $\delta'(q_<, c) = q_s$ and $\delta'(q_>, \sigma) = q_>$, $\delta'(q_>, c) = q_s$. Then, we set $R = \{(q_i, q_j) \mid i < j\}$ for all states in Q. Further, for every copied state $\hat{q_k}$ we extend R by the sets: $\{(\hat{q_k}, q_k)\}$, $\{(q_i, \hat{q_k}), (\hat{q_k}, q_j) \mid i < k, k < j\}$, and $\{(\hat{q_i}, \hat{q_k}), (\hat{q_k}, \hat{q_j}) \mid i < k < j\}$ for all copied states $\hat{q_i}, \hat{q_j}$. For the states $q_<, q_>$, we add $\{(q_<, q) \mid q \neq q_< \in Q'\}$ and $\{(q, q_>) \mid q \neq q_> \in Q'\}$ to R. ◀

▶ **Theorem 30.** Subset-Sync-Under-Total-$\propto^{l<f}_{w@p}$ *is NP-complete.*

▶ **Theorem 31.** *The following subset synchronization problems are PSPACE-complete for both -0- and -1-:* Subset-Sync-Under-$\propto^{l<l}_{w@s}$, -$\propto^{l<l}_{w@p}$, -$\propto^{l\leq l}_{w@s}$, -$\propto^{l\leq l}_{w@p}$, -$\propto^{l<f}_{w@p}$.

Several other results can be transferred from [20] to the corresponding version of the Sync-Under-Total-*0*-$\propto^{l<f}_{w@p}$ problem, such as inapproximability of the problems of finding a shortest synchronizing word; a synchronizing set of maximal size (here also W[1]-hardness can be observed); or determining the rank of a given set. Further, by the observation (in [20]) that, in the construction given in [18, 11] the automata are WAAs, we immediately get NP-hardness for finding a shortest synchronizing word for all of our orders (for order $l < l$ and $l \leq l$ set $R = \emptyset$).

5 Conclusion

We discussed ideas how constraints for the design of assembly lines caused by the physical deformation of a part can be described in terms of synchronization problems. For that, we considered several ways how a word can imply an order of states in Q. We considered the complexity of synchronizing an automaton under different variants of orders and observed that the complexity of considering an order on the set of active states may differ from considering the order on each single path. Although we were able to get a good understanding of the complexity of synchronization under the considered orders, some questions remained open: We only know that Sync-Under-$\propto^{l<l}_{w@p}$ is contained in NP but it is open whether the problem is NP-complete or if it can be solved in polynomial time. Conversely, for Sync-Under-*0*-$\propto^{l\leq l}_{w@p}$ the problem is NP-hard but its precise complexity is unknown. It would be quite surprising to observe membership in NP here since it would separate the complexity of this problem from the closely related problem Sync-Under-*1*-$\propto^{l\leq l}_{w@p}$. Further, it remains open whether for the other orders a drop in the complexity can be observed, when R is strict and total, as it is the case for $\propto^{l<f}_{w@p}$.

References

1 Journal of Automata, Languages and Combinatorics – Essays on the Černý Conjecture. `https://www.jalc.de/issues/2019/issue_24_2-4/content.html`. Accessed: 10/1/2020.

2 Dimitry S. Ananichev and Mikhail V. Volkov. Synchronizing monotonic automata. *Theor. Comput. Sci.*, 327(3):225–239, 2004.

3 Giorgio Ausiello, M. Protasi, A. Marchetti-Spaccamela, G. Gambosi, P. Crescenzi, and V. Kann. *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*. Springer-Verlag, Berlin, Heidelberg, 1st edition, 1999.

4 Marie-Pierre Béal and Dominique Perrin. *Synchronised Automata*, page 213–240. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2016.

5 Jens Bruchertseifer and Henning Fernau. Synchronizing series-parallel automata with loops. In Rudolf Freund, Markus Holzer, and José M. Sempere, editors, *Eleventh Workshop on Non-Classical Models of Automata and Applications, NCMA 2019, Valencia, Spain, July 2-3, 2019.*, pages 63–78. Österreichische Computer Gesellschaft, 2019.

6 Ján Černý. Poznámka k homogénnym eksperimentom s konečnými automatami. *Matematicko-fyzikalny Časopis Slovensk*, 14(3):208–215, 1964.

7 Ján Cerný. A note on homogeneous experiments with finite automata. *Journal of Automata, Languages and Combinatorics*, 24(2-4):123–132, 2019.

8 Krishnendu Chatterjee and Laurent Doyen. Computation tree logic for synchronization properties. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, pages 98:1–98:14, 2016.

9 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

10 Laurent Doyen, Line Juhl, Kim Guldstrand Larsen, Nicolas Markey, and Mahsa Shirmohammadi. Synchronizing words for weighted and timed automata. In Venkatesh Raman and S. P. Suresh, editors, *34th International Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2014, December 15-17, 2014, New Delhi, India*, volume 29 of *LIPIcs*, pages 121–132. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2014.

11 David Eppstein. Reset sequences for monotonic automata. *SIAM J. Comput.*, 19(3):500–510, 1990.

12 Henning Fernau, Vladimir V. Gusev, Stefan Hoffmann, Markus Holzer, Mikhail V. Volkov, and Petra Wolf. Computational complexity of synchronization under regular constraints. In Peter

Rossmanith, Pinar Heggernes, and Joost-Pieter Katoen, editors, *44th International Symposium on Mathematical Foundations of Computer Science, MFCS 2019, August 26-30, 2019, Aachen, Germany*, volume 138 of *LIPIcs*, pages 63:1–63:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

13 Henning Fernau, Pinar Heggernes, and Yngve Villanger. A multi-parameter analysis of hard problems on deterministic finite automata. *J. Comput. Syst. Sci.*, 81(4):747–765, 2015.

14 Balázs Imreh and Magnus Steinby. Directable nondeterministic automata. *Acta Cybern.*, 14(1):105–115, 1999.

15 Pavel Martyugin. Computational complexity of certain problems related to carefully synchronizing words for partial automata and directing words for nondeterministic automata. *Theory Comput. Syst.*, 54(2):293–304, 2014.

16 Pavel V. Martyugin. Synchronization of automata with one undefined or ambiguous transition. In Nelma Moreira and Rogério Reis, editors, *Implementation and Application of Automata - 17th International Conference, CIAA 2012, Porto, Portugal, July 17-20, 2012. Proceedings*, volume 7381 of *Lecture Notes in Computer Science*, pages 278–288. Springer, 2012.

17 Balas K. Natarajan. An algorithmic approach to the automated design of parts orienters. In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, pages 132–142. IEEE Computer Society, 1986.

18 I. K. Rystsov. On minimizing the length of synchronizing words for finite automata. In *Theory of Designing of Computing Systems*, pages 75–82. Institute of Cybernetics of Ukrainian Acad. Sci., 1980. (in Russian).

19 Igor K. Rystsov. Polynomial complete problems in automata theory. *Inf. Process. Lett.*, 16(3):147–151, 1983.

20 Andrew Ryzhikov. Synchronization problems in automata without non-trivial cycles. *Theor. Comput. Sci.*, 787:77–88, 2019.

21 Andrew Ryzhikov and Anton Shemyakov. Subset synchronization in monotonic automata. *Fundam. Inform.*, 162(2-3):205–221, 2018.

22 Sven Sandberg. Homing and synchronizing sequences. In Manfred Broy, Bengt Jonsson, Joost-Pieter Katoen, Martin Leucker, and Alexander Pretschner, editors, *Model-Based Testing of Reactive Systems, Advanced Lectures [The volume is the outcome of a research seminar that was held in Schloss Dagstuhl in January 2004]*, volume 3472 of *Lecture Notes in Computer Science*, pages 5–33. Springer, 2004.

23 Walter J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *J. Comput. Syst. Sci.*, 4(2):177–192, 1970.

24 Michael Sipser. *Introduction to the Theory of Computation.* PWS Publishing Company, 1997.

25 Avraham Trakhtman. The Černý conjecture for aperiodic automata. *Discrete Mathematics & Theoretical Computer Science*, 9(2), 2007.

26 Uraz Cengiz Türker and Hüsnü Yenigün. Complexities of some problems related to synchronizing, non-synchronizing and monotonic automata. *Int. J. Found. Comput. Sci.*, 26(1):99–122, 2015.

27 Mikhail V. Volkov. Synchronizing automata and the Černý conjecture. In Carlos Martín-Vide, Friedrich Otto, and Henning Fernau, editors, *Language and Automata Theory and Applications, Second International Conference, LATA 2008, Tarragona, Spain, March 13-19, 2008. Revised Papers*, volume 5196 of *Lecture Notes in Computer Science*, pages 11–27. Springer, 2008.

28 Vojtech Vorel and Adam Roman. Parameterized complexity of synchronization and road coloring. *Discrete Mathematics & Theoretical Computer Science*, 17(1):283–306, 2015.

29 Petra Wolf. Synchronization under dynamic constraints. *CoRR*, abs/1910.01935, 2019. `arXiv:1910.01935`.