Synchronizing words under LTL constraints

Nathalie Bertrand, Hugo Francon, Nicolas Markey

Synchronizing words under LTL constraints

Nathalie Bertrand, Hugo Francon, Nicolas Markey

IRISA – Inria, CNRS, Univ. Rennes 1 – France

Abstract

Synchronizing a (deterministic, finite-state) automaton is the problem of finding a sequence of actions to be played in the automaton in order to end up in the same state independently of the starting state. We consider synchronization with LTL constraints on the executions leading to synchronization, extending the results of [Petra Wolf. Synchronization under dynamic constraints. FSTTCS'20] by showing that the problem is PSPACE-complete for LTL as well as for restricted fragments (involving only modality F or G), while it is NP-complete for constraints expressed using only modality X.

1. Introduction

Synchronizing an automaton. A synchronizing word for a deterministic finite-state automaton $\mathcal{A}$ is a finite word that always takes the automaton to the same state, independently of the state of $\mathcal{A}$ it starts from. When the automaton represents the behaviours of a computer system, a synchronizing word can be seen as a sequence of actions to perform on the system so as to take it to the same state (in a sense, to regain control) if we initially don't know which state it is in.

Example. *Consider the coffee-vending machine whose behaviour is depicted in Fig. 1: it asks the user how much sugar they want, and whether they want milk; the user may insert coins at any time to get coffee with the last choices that have been recorded.*

Unfortunately, the display is broken. You do not know what state the machine is in, but you want a coffee with milk and no sugar. By noticing that the sequence $\langle$`no`, `yes`, `no`$\rangle$ *is a synchronizing word, always ending up in the* milk *state, you will get a beverage*

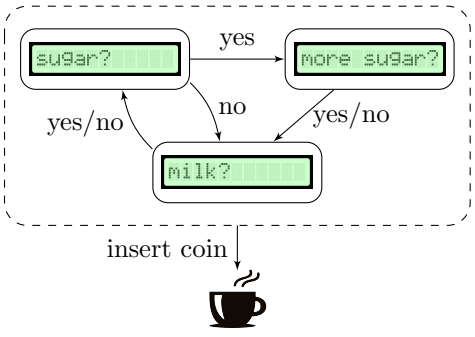

Figure 1: A coffee-vending machine

at your taste (by appending $\langle \texttt{yes}, \texttt{no} \rangle$ *to the previous sequence to be sure you get milk and no sugar).*

The problem of regaining control on a device dates back to Moore [Moo56] and Ginsburg [Gin58], and was popularized in the early 1960's by Černý [Čer64], who proposed a polynomial-time algorithm for deciding the existence of a synchronizing word. Černý's algorithm works by iteratively synchronizing states by pairs, based on the following fact: an automaton has a synchronizing word if, and only if, for any two states s and s', there is a word w which, when read from s and from s' ends up in the same state. This way, the existence of a synchronizing word can even be decided in non-deterministic logarithmic space, and is actually NLOGSPACE-complete. Černý's result also entails that, if there is a synchronizing word for a given automaton, then there is one of size $O(n^3)$ (where n is the number of states of the automaton). Černý conjectured that the upper bound on the size of synchronizing words is $(n-1)^2$; this is a long-standing open problem in automata theory. Pin and Frankl proved in the early 1980's that an upper bound is $(n^3 - n)/6$ [Fra82, Pin83], and recently Szykuła and Shitov improved this bound to $0.1654n^3 + o(n^3)$ [Szy18, Shi19]. Additionally, the probability that an n-state deterministic complete automaton (chosen uniformly at random) has a synchronizing word of size $O(n \log^3(n))$ tends to 1 as n tends to $+\infty$ [Nic19].

Another, more naive algorithm for deciding the existence of a synchronizing word consists in building the powerset automaton and looking for a path to a singleton state; however, the resulting automaton has exponential size, and this algorithm thus runs in polynomial space. It can be used to solve the *subset*-synchronization-word problem, where the aim is to synchronize a subset of the set of states of the automaton; somewhat surprisingly, this problem can be proved to be PSPACE-complete [Rys83]; as a side result, shortest synchronizing words may be of exponential size for subset synchronization. We refer to classical surveys [Vol08, San05] for more background and results about synchronizing words.

Synchronizability in other classes of automata. Synchronizability has been studied in a variety of contexts since then: for timed automata and integer-weighted automata under energy constraints, synchronizability is PSPACE-complete [DJL+14]. The problem of synchronizing probabilistic automata is a bit more complex (conceptually): probabilistic automata generate infinite sequences of distributions over states; synchronizing a probabilistic automaton may then amount to reaching a distribution where the probability mass ultimately accumulates in a single state. The existence of a synchronizing word in that setting is PSPACE-complete [DMS11, DMS12]; several variants and extensions of this result are presented in [DMS19]. Synchronizability has also been studied in the context of register automata, where again the problem has been proven to be PSPACE-complete [QS19].

Our contributions. Recently, Wolf considered the problem of finding synchronizing words with additional *dynamic constraints*: this requires that all the

paths generated by the considered synchronizing word from all the states of $\mathcal{A}$ satisfy some constraints on the order of the states being visited; an example of such a constraint is that the last occurrence of state p always precedes the last occurrence of some other state q. Deciding the existence of such synchronizing words is PSPACE-complete in most cases [Wol20].

In the present paper, we extend the results of Wolf by showing that the existence of synchronizing words with additional LTL constraints on all paths is also PSPACE-complete. Membership in PSPACE relies on the translation of the LTL constraint into a finite-state automaton, and on an algorithm similar to the powerset automaton described above. Hardness in PSPACE is actually proved for very simple, fixed formulas involving only LTL modalities F or G. When X is the only allowed modality, we show that the problem is NP-complete.

2. Definitions

2.1. Linear-time Temporal Logic (LTL)

The linear-time temporal logic (LTL) [Pnu77] over the set AP of atomic propositions is the set of formulas built along the following grammar:

$$\mathsf{LTL} \ni \phi, \psi ::= p \mid \phi \vee \psi \mid \neg\phi \mid \mathsf{X}\phi \mid \phi \,\mathsf{U}\, \psi$$

where p ranges over $\mathsf{AP} \uplus \{\mathsf{true}\}$. The size of an LTL formula ϕ, denoted with $|\phi|$, is the size of its derivation tree.

A finite trace over AP is a sequence $\rho = (v_i)_{0 \le i \le n}$ with $n \ge 0$ and $v_i \subseteq \mathsf{AP}$ for all $0 \le i \le n$. We may equivalently see each v_i as a mapping $\mathsf{AP} \to \{0, 1\}$. The size of ρ, denoted with $|\rho|$, is $n + 1$.

For our purposes, we interpret LTL formulas over finite traces. For any finite trace $\rho = (v_i)_{0 \le i \le n}$ and any index $0 \le j \le n$, we inductively define:

$$\begin{array}{lcl}
\rho, j \models \mathsf{true} & & \\
\rho, j \models p & \iff & v_j(p) = 1 \\
\rho, j \models \phi \vee \psi & \iff & \rho, j \models \phi \text{ or } \rho, j \models \psi \\
\rho, j \models \neg\phi & \iff & \neg(\rho, j \models \phi) \\
\rho, j \models \mathsf{X}\phi & \iff & j < n \text{ and } \rho, j+1 \models \phi \\
\rho, j \models \phi \,\mathsf{U}\, \psi & \iff & \exists 0 \le k < n - j.\ \big(\rho, j+k \models \psi \text{ and} \\
 & & \qquad \forall 0 \le l < k.\ \rho, j+l \models \phi\big).
\end{array}$$

Intuitively, X means "next" while U stands for "until". We write $\rho \models \phi$ as a shorthand for $\rho, 0 \models \phi$.

Example. *Formula $\phi \,\mathsf{U}\, \psi$ demands that ψ be true at some point along the path. A weaker modality, called* weak until*, relaxes this condition by allowing ψ to never hold true. This modality can be defined as $\phi \,\mathsf{W}\, \psi \equiv \mathsf{G}\phi \vee \phi \,\mathsf{U}\, \psi$.*

We extend the syntax of LTL by defining false as the negation of true, conjunction as the dual of disjunction, and with the following extra constructs:

- $\phi \,\mathsf{R}\, \psi \equiv \neg(\neg\phi \,\mathsf{U}\, \neg\psi)$ is the dual of U: this modality is read "release". Its semantics is given by

$$\rho, j \models \phi \,\mathsf{R}\, \psi \qquad \Longleftrightarrow \qquad \forall 0 \leq k < n - j.\ \big(\rho, j+k \models \psi \text{ or } \exists 0 \leq l < k.\ \rho, j+l \models \phi\big).$$

  It can be checked that $\phi \,\mathsf{R}\, \psi$ is equivalent to $\psi \,\mathsf{W}\, (\phi \wedge \psi)$ in our setting.

- $\widetilde{\mathsf{X}}\phi \equiv \neg\mathsf{X}\neg\phi$ is the dual of X: it holds true at position j of ρ if either j is the last position in ρ, or ϕ holds at position $j+1$;

- $\mathsf{F}\phi \equiv \mathsf{true}\,\mathsf{U}\,\phi$; then $\rho, j \models \mathsf{F}\phi$ if there exists $0 \leq k$ such that $\rho, j+k \models \phi$;

- $\mathsf{G}\phi \equiv \neg\mathsf{F}\neg\phi$; then $\rho, j \models \mathsf{G}\phi$ if for all $0 \leq k \leq |\rho| - j$, it holds $\rho, j+k \models \phi$. Notice that $\mathsf{G}\phi \equiv \mathsf{false}\,\mathsf{R}\,\phi$.

Thanks to conjunction, $\widetilde{\mathsf{X}}$ and R, any LTL formula can be turned into *negation normal form*, where negations are only allowed to apply to atomic propositions. Said differently, LTL can be defined as

$$\mathsf{LTL} \ni \phi, \psi ::= p \mid \neg p \mid \phi \vee \psi \mid \phi \wedge \psi \mid \mathsf{X}\phi \mid \widetilde{\mathsf{X}}\phi \mid \phi \,\mathsf{U}\, \psi \mid \phi \,\mathsf{R}\, \psi. \tag{1}$$

In the sequel, we will consider some sublogics of LTL defined by restricting the set of allowed modalities. For a subset $\mathsf{M} \subseteq \{\mathsf{X}, \widetilde{\mathsf{X}}, \mathsf{F}, \mathsf{G}, \mathsf{U}, \mathsf{R}\}$, we write $\mathsf{L}^+(\mathsf{M})$ for the sublogic obtained by restricting the grammar of Eq. (1) to only allow the modalities in M [SC85]. In the sequel we will encounter the sublogics $\mathsf{L}^+(\mathsf{F})$, $\mathsf{L}^+(\mathsf{G})$ and $\mathsf{L}^+(\mathsf{X})$. Notice that in those sublogics, negation is only allowed in front of atomic propositions, so that e.g. modality F cannot be used in $\mathsf{L}^+(\mathsf{G})$.

2.2. Synchronizing words

Definition 1. *Let Σ be a finite alphabet, and AP be a finite set of atomic propositions. An* AP-labelled automaton over Σ *is a tuple $\mathcal{A} = \langle S, T, \ell\rangle$ where S is a finite set of states, $T \subseteq S \times \Sigma \times S$ is a set of transitions, and $\ell\colon S \to 2^{\mathsf{AP}}$ is a labelling function.*

The size of such an automaton is defined as the cardinality of its state set S. In some situations, the AP-labelling of those automata will be useless; we will simply call them *automata* in that case; such automata will have no labelling function ℓ.

Pick a labelled automaton $\mathcal{A} = \langle S, T, \ell\rangle$, and a state $s \in S$. A (finite) path from s in $\mathcal{A}$ is a sequence $\pi = (t_i)_{1 \leq i \leq n}$ of transitions from T such that, writing $t_i = \langle s_i, \sigma_i, s'_i\rangle$ for all $1 \leq i \leq n$, it holds $s_1 = s$ and $s'_i = s_{i+1}$ for all $1 \leq i < n$. We write $\mathsf{first}(\pi)$ and $\mathsf{last}(\pi)$ for the states s_1 and s'_n when $n > 0$, with the special case $\mathsf{first}(\pi) = \mathsf{last}(\pi) = s$ when $n = 0$ (i.e., when π is the empty sequence). The *word* associated with π is the finite sequence $\mathsf{word}(\pi) = (\sigma_i)_{1 \leq i \leq n} \in \Sigma^n$.

An *acceptor automaton* over Σ is an automaton $\mathcal{A} = \langle S, T\rangle$ over Σ equipped with two subsets I and F of S, respectively containing *initial* and *final* states. A word $\sigma \in \Sigma^n$ is accepted by the acceptor automaton $\mathcal{B} = \langle\langle S, T\rangle, I, F)\rangle$

whenever there exists a path π with $\mathsf{first}(\pi) \in I$, $\mathsf{last}(\pi) \in F$, and $\mathsf{word}(\pi) = \sigma$. The language of $\mathcal{B}$ is the set of words it accepts.

Given a state $s \in S$ and a finite word w in Σ^*, we define the set $s \otimes w$ of paths generated by reading w from s inductively as follows: for the empty word ϵ, the set $s \otimes \epsilon$ only contains the empty path from s; for any non-empty word w, writing $w = w' \cdot \sigma$ with $\sigma \in \Sigma$, we have

$$s \otimes (w' \cdot \sigma) = \{\pi \cdot \langle r, \sigma, r' \rangle \mid \pi \in s \otimes w' \text{ and } r = \mathsf{last}(\pi) \text{ and } \langle r, \sigma, r' \rangle \in T\}.$$

Finally, we let $s \odot w = \{\mathsf{last}(\pi) \mid \pi \in s \otimes w\}$ for the set of states reached from s by reading w, and for any subset $Q \subseteq S$, we define $Q \otimes w = \bigcup_{s \in Q} s \otimes w$ and $Q \odot w = \bigcup_{s \in Q} s \odot w$.

A labelled automaton is said *deterministic* (resp. *complete*) whenever for any $s \in S$ and any $\sigma \in \Sigma$, the cardinality of $s \odot \sigma$ is at most (resp. at least) 1.

A set of states $Q \subseteq S$ of a complete deterministic labelled automaton $\mathcal{A}$ is said to admit a synchronizing word if there exists a word w such that $Q \odot w$ is a singleton. A complete deterministic automaton is said *synchronizable* if its set of states S has a synchronizing word [Čer64].

Example. *Consider the automaton of Fig. 2 (borrowed from [Wol20]), over the 2-letter alphabet* $\{\mathtt{a}, \mathtt{b}\}$. *This automaton admits* `bab` *as a synchronizing word.*

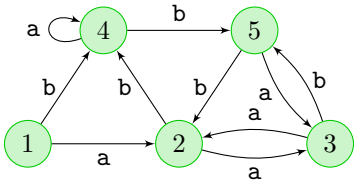

Figure 2: A synchronizable automaton with $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ and $\mathsf{AP} = \{1, 2, 3, 4, 5\}$.

With a path $\pi = (t_i)_{1 \leq i \leq n}$ in a labelled automaton $\mathcal{A}$, writing $t_i = \langle s_i, \sigma_i, s_i' \rangle$ for all $1 \leq i \leq n$, we associate the sequence of states $(q_i)_{0 \leq i \leq n}$ defined as $q_0 = s_1$ and $q_i = s_i'$ for all $1 \leq i \leq n$. We then let $\mathsf{trace}(\pi) = (\ell(q_i))_{0 \leq i \leq n}$. For a formula $\phi \in \mathsf{LTL}$, we write that $\pi \models \phi$ when $\mathsf{trace}(\pi), 0 \models \phi$.

In the sequel, we focus on the following problem:

Definition 2. *Subset-synchronizability under* LTL *constraints is the problem of deciding, given a complete, deterministic* AP*-labelled automaton* $\mathcal{A} = \langle S, T, \ell \rangle$ *over* Σ*, a subset* $Q \subseteq S$*, and an* LTL *formula* ϕ *over* AP*, whether there exists a finite word* $w \in \Sigma^*$ *such that* $Q \odot w$ *is a singleton and for all* $\pi \in Q \otimes w$*, it holds* $\pi \models \phi$*. Synchronizability under* LTL *constraints is the special case of subset-synchronizability where* $Q = S$*.*

Example. *Consider again the automaton of Fig. 2, and the property that the transition from state* 5 *to state* 2 *is never taken; this condition can be expressed as* ϕ: $\mathsf{G}(5 \Rightarrow \mathsf{X}\neg 2)$ *(assuming that the names of the states are the atomic*

propositions). Obviously, `bab` *is not a good candidate; actually, because of the path starting from state* 5*, any synchronizing word under* LTL *constraint* ϕ *has to start with an* `a`*; it turns out that* `abab` *is a valid solution. On the other hand, it can be checked that no synchronizing word can avoid the transition from state* 2 *to state* 4*.*

3. Related work: synchronizing words under dynamic constraints

Our work can be seen as an extension of some of the results of [Wol20]. In that paper, Wolf considered the problem of synchronizing labelled automata under *dynamic constraints*, enforcing different kinds of constraints on the first or last occurrence of states of the automaton. Formally, with a (non-necessarily synchronizing) finite word, we can associate different orders on the states of the (deterministic complete) automaton: for instance, the order $\propto_{p@w}^{l<f}$ is defined as follows: for any two states p and q in S, it holds $p \propto_{p@w}^{l<f} q$ if, and only if, for all $s \in S$, along the path $s \otimes w$, either p never occurs, or q never occurs, or the last occurrence of p occurs before the first occurrence of q. Synchronizability under $\propto_{p@w}^{l<f}$ then asks, given an automaton and a binary relation $R \subseteq S^2$, whether there is a synchronizing word w for which $R \subseteq \propto_{p@w}^{l<f}$; in other terms, for all $\langle p, q\rangle \in R$, the last occurrence of state p (if any) must occur before the first occurrence of state q (if any) along any path following w in $\mathcal{A}$. This is easily expressed in LTL, e.g. as $\mathsf{G}(q \Rightarrow \mathsf{G}\neg p)$, assuming that $\mathsf{AP} = S$ and each state is labelled with (only) its name. It is proved in [Wol20] that synchronizability under $\propto_{p@w}^{l<f}$ is PSPACE-complete.

Similarly, the order $\propto_{p@w}^{l\leq l}$ requires that the last occurrence of the left-hand-side state should occur before or at the same time as the last occurrence of the right-hand-side state. Constraint $p \propto_{p@w}^{l\leq l} q$ can be expressed in LTL as $\mathsf{G}(p \Rightarrow \mathsf{F}q)$. The associated synchronizability problem is shown NP-hard (and in PSPACE) in [Wol20]; it is PSPACE-complete when the constraint does not take into account the starting state of each path (equivalently, when the LTL constraint is prefixed by X).

4. Synchronizability under LTL constraints

Our main result can be stated as follows:

Theorem 3. *Synchronizability under* LTL *constraints is* PSPACE*-complete, even for* fixed *formulas in* $L^+(\mathsf{F})$ *or* $L^+(\mathsf{G})$*.*

Proof. We first prove membership in PSPACE, and start with an overview of the proof. Intuitively, given a formula ϕ over AP and an AP-labelled automaton $\mathcal{A}$, the first step of the algorithm consists in building an acceptor automaton $\mathcal{B}_\phi$ over 2^{AP} accepting exactly the finite traces satisfying ϕ (following the classical transformation of LTL formula into automata [DV13]). Identifying traces over AP with words over the alphabet 2^{AP}, we then construct the synchronized

product $\mathcal{C}$ of $\mathcal{A}$ and $\mathcal{B}_\phi$, which accepts the traces generated by $\mathcal{A}$ and satisfying ϕ. The existence of a synchronizing word for $\mathcal{A}$ with constraint ϕ is then equivalent to the existence of a word w which, when read from a set of initial states of $\mathcal{C}$ covering all states of $\mathcal{A}$, reaches final states of $\mathcal{C}$ involving only one single state of $\mathcal{A}$. Checking the existence of such a word can be performed by a non-deterministic algorithm, which guesses w letter by letter, and computes the set of states reached by reading the current prefix of w from each of the initial states considered, until reaching a final configuration as described above. While a naive implementation of this algorithm requires exponential space, we notice that we do not need to explicitely build the automata $\mathcal{B}_\phi$ and $\mathcal{C}$, but only need to compute the successor of a state by a letter. Following this remark, we get a non-deterministic algorithm running in polynomial space; by Savitch's theorem [Sav70], this proves that synchronizability under LTL constraints is in PSPACE.

We now develop the details of this proof. For the sake of self-containment, we briefly sketch the construction of an acceptor automaton $\mathcal{B}_\phi = \langle\langle V, R\rangle, I, F\rangle$ over 2^{AP} associated with an LTL formula [BK08, DV13].

Fix a formula $\phi \in \mathsf{LTL}$; its set of subformulas $\mathsf{subf}(\phi)$ is the smallest set of LTL formulas containing ϕ and true, and such that, for any two formulas α and β of LTL,

- if $\alpha \vee \beta$ or $\alpha \,\mathsf{U}\, \beta$ appears in $\mathsf{subf}(\phi)$, then also α and β are in $\mathsf{subf}(\phi)$;
- if $\mathsf{X}\alpha$ or $\neg\alpha$ appears in $\mathsf{subf}(\phi)$, then $\alpha \in \mathsf{subf}(\phi)$;
- if $\alpha \in \mathsf{subf}(\phi)$ and α is not of the form $\neg\alpha'$, then $\neg\alpha \in \mathsf{subf}(\phi)$.

Clearly enough, $\mathsf{subf}(\phi)$ contains at most $2|\phi|$ formulas. A subset C of $\mathsf{subf}(\phi)$ is said to be maximally-consistent whenever it contains true and for all α and β in $\mathsf{subf}(\phi)$,

- if $\neg\alpha \in \mathsf{subf}(\phi)$, then $\neg\alpha \in C$ if, and only if, $\alpha \notin C$;
- if $\alpha \vee \beta \in \mathsf{subf}(\phi)$, then $\alpha \vee \beta \in C$ if, and only if, $\alpha \in C$ or $\beta \in C$;
- if $\alpha \,\mathsf{U}\, \beta \in \mathsf{subf}(\phi)$, then $\alpha \,\mathsf{U}\, \beta \in C$ implies that $\alpha \in C$ or $\beta \in C$.

The number of maximally-consistent subsets of $\mathsf{subf}(\phi)$ is at most $2^{2|\phi|}$.

The states of automaton $\mathcal{B}_\phi$ are the maximally-consistent sets of subformulas of ϕ, together with an extra state f; there is a transition $\langle v, l, v'\rangle$ from state $v \neq f$ to state $v' \neq f$ labelled with l if, and only if, for any $\psi \in \mathsf{subf}(\phi)$,

- if $\psi = \mathsf{X}\alpha$, then $\psi \in v$ if, and only if, $\alpha \in v'$;
- if $\psi = \alpha \,\mathsf{U}\, \beta$, then $\psi \in v$ if, and only if, $\beta \in v$ or $\alpha \in v$ and $\psi \in v'$;
- $l = v \cap \mathsf{AP}$.

Additionally, for any $v \neq f$, there is a transition from v to f labelled with $v \cap \mathsf{AP}$ whenever for all $\alpha \,\mathsf{U}\, \beta \in \mathsf{subf}(\phi)$, if $\alpha \,\mathsf{U}\, \beta \in v$, then $\beta \in v$ and for all $\mathsf{X}\alpha \in \mathsf{subf}(\phi), \mathsf{X}\alpha \notin v$. Finally, all states containing ϕ are initial, and f is the unique final state. It can be proved inductively that the language accepted by $\mathcal{B}_\phi$ contains exactly the finite traces over AP that satisfy ϕ [BK08, DV13].

We consider the synchronized product of $\mathcal{A} = \langle S, T, \ell \rangle$ with $\mathcal{B}_\phi = \langle \langle V, R \rangle, I, F \rangle$, denoted with $\mathcal{A} \ltimes \mathcal{B}_\phi$, which is the automaton $\mathcal{C} = \langle Q, R \rangle$ over Σ defined as:

- $Q = \{\langle s, v \rangle \in S \times V \mid \ell(s) = v \cap \mathsf{AP}\} \uplus \{\langle s, f \rangle \mid s \in S\}$;
- R is the set of triples $\langle \langle s, v \rangle, \sigma, \langle s', v' \rangle \rangle \in Q \times \Sigma \times Q$ such that there are transitions $\langle s, \sigma, s' \rangle$ in $\mathcal{A}$ and $\langle v, v \cap \mathsf{AP}, v' \rangle$ in $\mathcal{B}_\phi$.

By construction, for any path π in $\mathcal{A} \ltimes \mathcal{B}_\phi$ starting from some state $\langle s, v \rangle$ with $v \in I$ and ending in some state $\langle s', f \rangle$, it holds $\pi \models \phi$.

In order to decide synchronizability of $\mathcal{A} = \langle S, T, \ell \rangle$ under LTL constraint ϕ, we consider the cross-product $\mathcal{P}$ of $|S|$ copies of $\mathcal{A} \ltimes \mathcal{B}_\phi$, and look for a path π from some state of the form $(\langle s_i, v_i \rangle)_{1 \leq i \leq |S|}$ with $\{s_i \mid 1 \leq i \leq |S|\} = S$ and $v_i \in I$ for all $1 \leq i \leq |S|$, to a state of the form $(\langle s'_i, f \rangle)_{1 \leq i \leq |S|}$ where $\{s'_i \mid 1 \leq i \leq |S|\}$ is a singleton. Clearly enough, any such path π is such that for all $s \in S$, the path $s \otimes \mathsf{word}(\pi)$ in $\mathcal{A}$ satisfies ϕ, since it corresponds to some path from $\langle s_i, v_i \rangle$ to $\langle s'_i, f \rangle$ in the product automaton $\mathcal{A} \ltimes \mathcal{B}_\phi$. Since the size of $\mathcal{P}$ is at most $(|S| \times 2^{|\phi|})^{|S|}$, if such a path π exists, there must be one that never visits the same state twice, thus there is one of size at most $(|S| \times 2^{|\phi|})^{|S|}$. As a consequence, synchronizability of $\mathcal{A}$ under LTL constraint ϕ can be decided by non-deterministically building such a path step-by-step, using space polynomial in both $|\phi|$ and $|S|$ to store the current state of $\mathcal{P}$ being visited and a counter to stop the computation if no solution is found after $(|S| \times 2^{|\phi|})^{|S|}$ steps. Since $\mathsf{PSPACE} = \mathsf{NPSPACE}$ [Sav70], this proves that synchronizability under LTL constraints is in PSPACE.

Hardness in PSPACE is easily derived from the results of [Wol20], even for a fixed LTL formula in $\mathsf{L}^+(\mathsf{G})$ (see Section 3). We propose an alternative PSPACE-hardness proof for fixed formulas *without nesting of modalities*, and involving only modality G (i.e., in $\mathsf{L}^+(\mathsf{G})$), or only modality F (i.e., in $\mathsf{L}^+(\mathsf{F})$). Our proof is based on a reduction from the subset-synchronizability problem (with no LTL constraints), which is known to be PSPACE-complete [Rys83].

Consider an automaton $\mathcal{A} = \langle S, T \rangle$ over alphabet Σ, and a non-empty subset $Q \subseteq S$ of states that we want to synchronize. We define a labelled automaton $\mathcal{A}' = \langle S', T', \ell' \rangle$ over Σ' by letting $S' = S \uplus \{s_\#, s_0, s'_0, s_f, s'_f\}$, $\Sigma' = \Sigma \uplus \{\#\} \uplus S$, and $T' = T \uplus U$ where U contains the following transitions:

- $\langle s_0, \#, s_\# \rangle$, $\langle s'_0, \#, s_\# \rangle$, and $\langle s_0, \sigma, s'_0 \rangle$ and $\langle s'_0, \sigma, s'_0 \rangle$ for all $\sigma \in \Sigma \uplus S$. Since s_0 will have no incoming transitions, the automaton cannot synchronize in s_0, so that any synchronizing word must contain $\#$. Using an LTL constraint, we will enforce that any synchronizing word begins with $\#$;
- all transitions from s_f and s'_f go to s'_f; this way, if synchronization is possible, the synchronized runs will end in s'_f;

- $\langle s, \#, s_\# \rangle$ for all $s \in \{s_\#\} \cup S \setminus Q$, and $\langle q, \#, q \rangle$ for all $q \in Q$. This way, when reading the first letter # of a synchronizing word, all states in Q take a self-loop, states s_f and s'_f move to s'_f, while the other states (in particular those in $S \setminus Q$) all go to $s_\#$;
- $\langle s, s, s_f \rangle$ for all $s \in S$, and $\langle s, s', s_\# \rangle$ for all s and s' in S with $s' \neq s$. Additionally, U also contains $\langle s_\#, s, s'_f \rangle$ for all $s \in S$. An LTL constraint will forbid the runs originating from Q to ever visit $s_\#$, so that these runs can only leave S by reading a letter from S, and they must have synchronized before doing so;
- finally, $\langle s_\#, \sigma, s_\# \rangle$ for all $\sigma \in \Sigma$.

The construction is depicted on Fig. 3. In that figure, $\star$-transitions represent all labels that do not already appear on other transitions leaving the same state (so that the automaton is deterministic and complete).

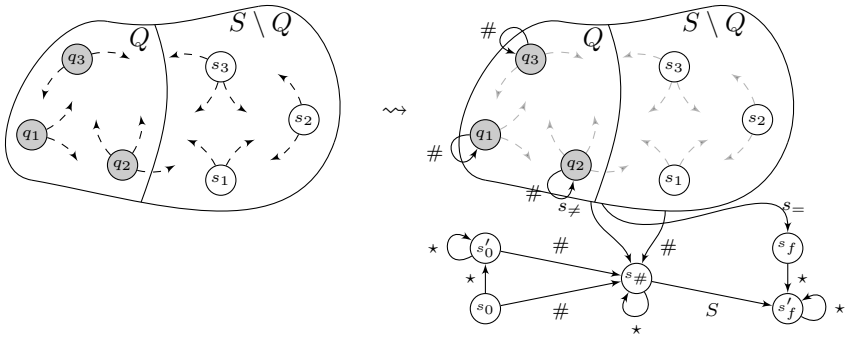

Figure 3: Reduction of subset-synchronizability to LTL-synchronizability

We let $\mathsf{AP}' = \{Q, s_\#, s_0, s'_0\}$, with $\ell'(q) = \{Q\}$ for all $q \in Q$, and $\ell'(s) = s$ for $s \in \{s_\#, s_0, s'_0\}$. Finally, we let $\phi = [s_0 \Rightarrow \mathsf{G}(\neg s'_0)] \wedge [Q \Rightarrow \mathsf{G}(\neg s_\#)]$; the first conjunct forces the synchronizing word (if any) to start with a #-transition, while the second one forbids the states in Q to ever visit $s_\#$, as required above.

We prove that Q can be synchronized in $\mathcal{A}$ if, and only if, S' can be synchronized in $\mathcal{A}'$ under LTL constraint ϕ. First, if w is a synchronizing word for Q in $\mathcal{A}$, then $w' = \# \cdot w \cdot q_w \cdot \#$ is a synchronizing word in $\mathcal{A}'$ satisfying ϕ, where q_w is such that $Q \odot w = \{q_w\}$: reading w' from states in Q first loops on the states in Q, then synchronizes to q_w, and finally goes to s_f and then s'_f, thereby satisfying ϕ; reading w' from s_f or s'_f eventually loops in s'_f, trivially satisfying ϕ; reading w' from any other state first goes to $s_\#$, loops there while reading w, and finally moves to s'_f, thereby also satisfying ϕ.

Conversely, consider a synchronizing word w' in $\mathcal{A}'$ under constraint ϕ. As argued above, it must start with # as otherwise reading w' from s_0 would visit s'_0 and falsify ϕ; states in Q then take their self-loops, while s_f and s'_f both go to s'_f, and all other states go to $s_\#$. As argued above, synchronization

can only happen in s'_f, and since the runs originating from Q cannot visit $s_\#$, they cannot leave S by reading $\#$; hence some letter in S must eventually occur in order to achieve synchronization. But this may only occur when all paths originating from the states in Q have synchronized in S, as otherwise some of those runs would visit $s_\#$. This proves our result.

This proof easily extends to fixed formulas in $\mathsf{L}^+(\mathsf{F})$: the construction of the labelled automaton is almost unchanged, except that we swap the transitions from s_0: we have $\langle s_0, \#, s'_0\rangle$ and $\langle s_0, \sigma, s_\#\rangle$ for all $\sigma \in \Sigma' \setminus \{\#\}$. Formula $\phi = [s_0 \Rightarrow \mathsf{F}s'_0] \wedge [Q \Rightarrow \mathsf{F}s_f]$ achieves the same constraints as above: the first conjunct forces to start with $\#$, and the second one prevents the synchronizing runs originating from Q to visit $s_\#$. □

Remark 1. *Notice that our algorithm is readily adapted to subset-synchronizability with LTL constraints, which is thus also in PSPACE (hence PSPACE-complete).*

Remark 2. *The last part of the hardness proof (for $\mathsf{L}^+(\mathsf{F})$) implies that the problem SYNC-UNDER-0-$\propto^{l\leq l}_{w@p}$, which is proven to be NP-hard in [Wol20, Theorem 16], is actually PSPACE-hard (hence PSPACE-complete). Indeed, the $\mathsf{L}^+(\mathsf{F})$ constraint can be encoded by requiring that the last occurrence of s_0 precedes the last occurrence of s'_0, and that the last occurrence of each state in Q precedes the last occurrence of s_f.*

Theorem 4. *Synchronizability under $\mathsf{L}^+(\mathsf{X})$-constraints is NP-complete. It is NLOGSPACE-complete when the $\mathsf{L}^+(\mathsf{X})$ formula is fixed.*

Proof. Membership in NP can be proven in two steps: first, we have to check that all states can be synchronized, using the pairwise-synchronization algorithm. If so, in order to prove that there is a synchronizing word whose traces satisfy ϕ, we guess a prefix of size d of this synchronizing word, where d is the maximal number of nested X modality in ϕ; since the whole automaton can be synchronized, the set of states reached after reading this prefix can also be synchronized.

When the formula is fixed, the second part of the algorithm runs in NLOGSPACE: it guesses and stores the prefix (using space $O(1)$), and checks from each state of $\mathcal{A}$ that the resulting path satisfies ϕ, which can be achieved in LOGSPACE. Since the classical synchronizability problem is NLOGSPACE-complete, the first part of the algorithm can be performed in NLOGSPACE (and hardness for fixed $\mathsf{L}^+(\mathsf{X})$ formulas follows).

NP-hardness for non-fixed formulas follows from the NP-hardness of model checking for $\mathsf{L}^+(\mathsf{X})$ [Mar02]: given an AP-labelled automaton $\mathcal{A}$, a state s_0 of $\mathcal{A}$, and a formula $\psi \in \mathsf{L}^+(\mathsf{X})$, deciding whether there is a path from s_0 satisfying ψ is NP-complete. This problem is easily reduced to our synchronizability problem with constraints in $\mathsf{L}^+(\mathsf{X})$: we add two extra states t and t', and two extra letters $\#$ and \$ in Σ; all $\#$-transitions go to t, except for a $\#$-self-loop on s_0, and all non-$\#$-transitions from t and t' go to t'; all \$-transitions go to t'. Finally, we label s_0, t and t' with their names, seen as two extra atomic propositions.

We consider the formula ϕ defined as

$$(\mathsf{X}t \wedge \mathsf{X}^{2..d+1}\neg t) \vee (\mathsf{X}s_0 \wedge \mathsf{X}\psi \wedge \mathsf{X}^{1..d+1}(\neg t \wedge \neg t'))$$

where again d is the maximal number of nested X in ψ, and $X^{a..b}\phi$ is a shorthand for the conjunction of all $\mathsf{X}^j\phi$ for j ranging from a to b.

Assume that there is a synchronizing word satisfying ψ. From t, the resulting path cannot satisfy $\mathsf{X}s_0$, hence it has to satisfy the first disjunct, so that this word has to start with #, and cannot involve # in the next d steps. From s_0, $\mathsf{X}t$ cannot be fulfilled, so that the second conjunct has to be satisfied, and the synchronizing word gives rise to a path satisfying ψ and visiting only states of $\mathcal{A}$ for the first d steps; since the truth value of ψ only depends on the first d steps, this shows the existence of a path in $\mathcal{A}$ satisfying ψ.

Conversely, if there is a path from s_0 satisfying ψ, then we can build a synchronizing word satisfying ϕ by prepending # and appending \$ to the corresponding word. The resulting word is easily checked to satisfy ϕ and, thanks to the last \$, to synchronize all paths to t'. □

Acknowledgement

We thank the anonymous reviewers for their very careful reading of earlier versions of this article, which helped us improve its writing.

References

[BK08] Ch. Baier and J.-P. Katoen. *Principles of Model-Checking*. MIT Press, 2008.

[Čer64] J. Černý. Poznámka k homogénnym experimentom s konečnými automatmi. *Matematicko-fyzikálny časopis*, 14(3):208–216, 1964.

[DJL+14] L. Doyen, L. Juhl, K. G. Larsen, N. Markey, and M. Shirmohammadi. Synchronizing words for weighted and timed automata. In FSTTCS'14, LIPIcs 29, p. 121–132. Leibniz-Zentrum für Informatik, 2014.

[DMS11] L. Doyen, Th. Massart, and M. Shirmohammadi. Infinite synchronizing words for probabilistic automata. In MFCS'11, LNCS 6907, p. 278–289. Springer, 2011.

[DMS12] L. Doyen, Th. Massart, and M. Shirmohammadi. Infinite synchronizing words for probabilistic automata (erratum). Research Report 1206.0995, arXiv, 2012.

[DMS19] L. Doyen, Th. Massart, and M. Shirmohammadi. The complexity of synchronizing Markov decision processes. *Journal of Computer and System Sciences*, 100:96–129, 2019.

[DV13] G. De Giacomo and M. Y. Vardi. Linear temporal logic and linear dynamic logic on finite traces. In IJCAI'13, p. 854–860. IJCAI organization, 2013.

[Fra82] P. Frankl. An extremal problem for two families of sets. *European Journal of Combinatorics*, 3(2):125–127, 1982.

[Gin58] S. Ginsburg. On the length of the smallest uniform experiment which distinguishes the terminal states of a machine. *Journal of the ACM*, 5(3):266–280, 1958.

[Mar02] N. Markey. Past is for free: On the complexity of verifying linear temporal properties with past. In EXPRESS'02, ENTCS 68, p. 89–106. Elsevier, 2002.

[Moo56] E. F. Moore. Gedanken-experiments on sequential machines. In *Automata Studies*, Annals of Mathematics Studies 34, p. 129–153. Princeton University Press, 1956.

[Nic19] C. Nicaud. The Černý conjecture holds with high probability. *Journal of Automata, Languages and Combinatorics*, 24(2-4):343–365, 2019.

[Pin83] J.-É. Pin. On two combinatorial problems arising from automata theory. In ICGTC'81, North-Holland Mathematics Studies 75, p. 535–548. Elsevier, 1983.

[Pnu77] A. Pnueli. The temporal logic of programs. In FOCS'77, p. 46–57. IEEE Comp. Soc. Press, 1977.

[QS19] K. Quaas and M. Shirmohammadi. Synchronizing data words for register automata. *ACM Transactions on Computational Logic*, 20(2):11:1–11:27, 2019.

[Rys83] I. K. Rystsov. Polynomial-complete problems in automata theory. *Information Processing Letters*, 16(1):147–151, 1983.

[San05] S. Sandberg. Homing and synchronizing sequences. In MBTRS'04, LNCS 3472, p. 5–33. Springer, 2005.

[Sav70] W. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4(2):177–192, 1970.

[SC85] A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logics. *Journal of the ACM*, 32(3):733–749, 1985.

[Shi19] Y. Shitov. An improvement to a recent upper bound for synchronizing words of finite automata. *Journal of Automata, Languages and Combinatorics*, 24(2-4):367–373, 2019.

[Szy18] M. Szykuła. Improving the upper bound on the length of the shortest reset word. In STACS'18, LIPIcs 96, p. 56:1–56:13. Leibniz-Zentrum für Informatik, 2018.

[Vol08] M. V. Volkov. Synchronizing automata and the Černý conjecture. In LATA'08, LNCS 5196, p. 11–27. Springer, 2008.

[Wol20] P. Wolf. Synchronization under dynamic constraints. In FSTTCS'20, LIPIcs 182, p. 9:1–9:18. Leibniz-Zentrum für Informatik, 2020.